프레임워크

AWS Well-Architected 프레임워크

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Well-Architected 프레임워크: 프레임워크

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

Table of Contents

요약 및 소개 1
소개 1
정의 2
아키텍처 4
일반 설계 원칙 5
프레임워크의 원칙 7
운영 우수성 7
설계 원칙 8
정의 9
모범 사례 9
리소스 17
보안 18
설계 원칙 18
정의 19
모범 사례 20
리소스 28
신뢰성 28
설계 원칙 29
정의 30
모범 사례 30
리소스 35
성능 효율성 35
설계 원칙 35
정의 36
모범 사례 37
리소스 41
비용 최적화 42
설계 원칙 42
정의 43
모범 사례 44
리소스 49
지속 가능성 50
설계 원칙 50
정의 51

- 모범 사례 ... 51
- 리소스 ... 57

검토 프로세스 ... 58

결론 ... 60

기여자 ... 61

참조 자료 ... 62

문서 수정 ... 63

부록: 질문 및 모범 사례 ... 67

- 운영 우수성 ... 67
  - Organization ... 67
  - 준비 ... 119
  - 운영 ... 183
  - 개선 ... 224
- 보안 ... 242
  - 보안 기초 ... 242
  - 자격 증명 및 액세스 관리 ... 264
  - 탐지 ... 317
  - 인프라 보호 ... 331
  - 데이터 보호 ... 355
  - 인시던트 대응 ... 387
  - 애플리케이션 보안 ... 408
- 신뢰성 ... 429
  - 기본 ... 429
  - 워크로드 아키텍처 ... 468
  - 변경 관리 ... 514
  - 장애 관리 ... 557
- 성능 효율성 ... 650
  - 아키텍처 선택 ... 651
  - 컴퓨팅 및 하드웨어 ... 665
  - 데이터 관리 ... 681
  - 네트워킹 및 콘텐츠 전송 ... 704
  - 프로세스 및 문화 ... 731
- 비용 최적화 ... 746
  - 클라우드 재무 관리 시행 ... 746
  - 지출 및 사용량 인식 ... 767
  - 비용 효율적인 리소스 ... 807

수요 관리 및 리소스 공급 844
시간 경과에 따른 최적화 856
지속 가능성 863
리전 선택 864
수요에 맞춘 조정 866
소프트웨어 및 아키텍처 879
Data 891
하드웨어 및 서비스 909
프로세스 및 문화 918
고지 사항 929
AWS 용어집 930

AWS Well-Architected 프레임워크

발행일: 2024년 11월 6일(문서 수정)

AWS Well-Architected Framework를 활용하면 AWS에서 시스템을 구축하면서 내리게 되는 결정의 장단점을 이해할 수 있습니다. 이 프레임워크를 사용하면 클라우드에서 신뢰할 수 있고 안전하며 효율적이고 경제적이면서 지속 가능한 시스템을 설계하고 운영하기 위한 아키텍처 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 신뢰할 수 있으며 효율적이고 경제적이면서 지속 가능한 워크로드를 설계하고 운영하기 위한 아키텍처 모범 사례를 알아볼 수 있습니다. 또한 모범 사례와 비교하여 아키텍처를 지속적으로 측정하고 개선할 영역을 파악할 수 있습니다. 아키텍처를 검토하는 프로세스는 아키텍처 관련 결정사항에 대한 건설적인 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. AWS는 수천 고객의 자사 아키텍처를 설계하고 검토해 왔습니다. 그리고 이러한 경험을 바탕으로 클라우드 시스템 설계의 모범 사례와 핵심 전략을 밝혀냈습니다.

AWS Well-Architected Framework에는 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. AWS가 진화를 거듭하고 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 클라우드의 서비스입니다. AWS WA Tool은 워크로드의 신뢰성 및 보안, 효율성, 경제성을 높일 수 있는 권장 사항을 제공합니다.

AWS Well-Architected Labs는 모범 사례 구현과 관련된 실무 경험이 포함된 설명서 및 코드 리포지토리를 제공하여 모범 사례 적용을 지원합니다. 또한 AWS Well-Architected 파트너 프로그램의 일원인 엄선된 AWS 파트너 네트워크(APN) 파트너와 함께 협업하고 있습니다. 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.

자사는 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.

표 1. AWS Well-Architected Framework 원칙

이름	설명
운영 우수성	효과적인 개발 및 워크로드 실행을 지원하고, 작업에 대한 인사이트를 얻으며, 지원 프로세스 및 절차를 지속적으로 개선하여 비즈니스 가치를 제공할 수 있는 능력입니다.
보안	보안 원칙은 보안 태세를 개선할 수 있는 방식으로 클라우드 기술을 활용하여 데이터, 시스템 및 자산을 보호하는 방법을 설명합니다.
신뢰성	신뢰성 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

이름	설명
	워크로드를 구현하기 위한 세부적인 모범 사례 지침을 제공합니다.
성능 효율성	컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력입니다.
비용 최적화	가장 낮은 가격으로 비즈니스 가치를 제공하도록 시스템을 실행하는 능력입니다.
지속 가능성	프로비저닝된 리소스의 이점을 극대화하고 필요한 총 리소스를 최소화하여 워크로드의 모든 구성 요소에서 에너지 소비를 줄이고 효율성을 높임으로써 지속 가능성에 미치는 영향을 지속적으로 개선할 수 있습니다.

AWS Well-Architected Framework에서는 다음 용어를 사용합니다.

- 구성 요소는 요구 사항을 충족하는 코드, 구성 및 AWS 리소스입니다. 구성 요소는 대개 기술 소유권의 단위이며 각기 분리되어 있습니다.
- 워크로드라는 용어는 비즈니스 가치를 제공하는 일련의 구성 요소를 가리키는 데 사용됩니다. 워크로드는 일반적으로 비즈니스 및 기술 책임자가 언급하는 수준의 디테일에 해당합니다.
- 아키텍처는 워크로드에서 구성 요소가 연동되는 방식이라고 볼 수 있습니다. 아키텍처 다이어그램에는 구성 요소의 통신 및 상호 작용 방식에 초점이 맞춰집니다.
- 마일스톤은 설계, 구현, 테스트, 출시 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 있습니다. 여러 스토리, 릴리스 및 작업으로 나눌 수 있습니다.
  - 중간: 이 작업을 완료하는 데는 며칠 또는 몇 주가 걸릴 수 있습니다. 여러 릴리스 및 작업으로 나눌 수 있습니다.

- 낮음: 이 작업을 완료하는 데는 몇 시간 또는 며칠이 걸릴 수 있습니다. 여러 작업으로 나눌 수 있습니다.

워크로드를 설계할 때는 업무 상황에 따라 이러한 원칙들을 절충해야 합니다. 이러한 비즈니스 의사결정에 따라 엔지니어링 우선 순위가 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 다른 제품 또는 기능 팀이 모범 사례를 충실히 따르도록 전반적으로 관리하는 기술 아키텍처 중앙 팀을 두기도 합니다. 일반적으로 기술 아키텍처 팀에는 테크니컬 아키텍트(인프라), 솔루션스 아키텍트(소프트웨어), 데이터 아키텍트, 네트워킹 아키텍트 및 보안 아키텍트와 같은 일련의 역할이 포함됩니다. 이러한 팀은 종종 TOGAF 또는 Zachman Framework를 엔터프라이즈 아키텍처 기능의 일부로 사용합니다.

AWS는 해당 기능을 중앙 팀을 두어 집중화하지 않고, 각각의 개별 팀에 나눠 주는 것을 선호합니다. 의사 결정 권한을 분산하게 될 경우, 예를 들어 팀들이 내부 기준을 충족하도록 확인하는 데 여러 위험 요소가 생기게 됩니다. 저희는 두 가지 방법으로 이러한 위험을 완화합니다. 먼저, 각 팀의 역량 확보를 위한 관행(업무 방법, 프로세스, 표준, 수락된 기준 등)을 통해 팀이 충족해야 할 표준에 대한 기준을 높일 수 있도록 전문가를 배치합니다 둘째, 표준에 부합하는지 확인하기 위해 자동화된 검사를 수행하는 메커니즘을 구현합니다.

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

즉, 사람의 노력을 규칙이나 프로세스 준수 여부를 확인하는 메커니즘(종종 자동화됨)으로 대체하는 것을 의미합니다. 이러한 분산된 접근 방식은 Amazon 리더십 원칙에 의해뒷받침되며 고객에서부터 역방향으로 작업하는 모든 역할에 걸쳐 문화를 수립합니다. 역방향 작업은 혁신 프로세스의 기본 요소입니다. 저희는 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. 새로운 팀이 이러한 역량을 확보하거나 기존 팀이 기준을 높일 수 있도록 설계를 검토하고

AWS 모범 사례를 이해하는 데 도움이 되는 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. 그리고 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 Framework는 내부 검토 프로세스를 고객 중심으로 구현한 결과이며, 주요 현장에서 활동하는 솔루션스 아키텍처 및 내부 엔지니어링 팀에서 엔지니어링 사고를 체계화한 것입니다. Well-Architected Framework는 이러한 결과를 활용할 수 있는 확장 가능한 메커니즘입니다.

자사는 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 책임 엔지니어가 특정 영역에 대한 사고 방식을 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.

- 용량 요구 사항 추적 중지: 워크로드를 배포할 때 용량을 잘못 결정하면 결국 고가의 유휴 리소스를 방치하게 되거나 제한된 용량으로 인한 성능 문제를 처리해야 합니다. 하지만 클라우드 컴퓨팅에서는 이러한 문제가 사라집니다. 필요한 만큼 용량을 많이 또는 적게 사용하다가 자동으로 스케일 인 및 스케일 아웃할 수 있기 때문입니다.
- 프로덕션 규모에서 시스템 테스트: 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 몇분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.
- 아키텍처 실험을 염두에 둔 자동화: 자동화를 통해 저렴한 비용으로 워크로드를 생성, 복제하고 수작업의 수고를 덜 수 있습니다. 자동화 과정의 변경 사항을 추적하고, 그 효과를 감사하고, 필요하면 이전의 파라미터로 되돌릴 수 있습니다.

- 진화하는 아키텍처 고려: 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, 이러한 초기의 결정 때문에 변경된 비즈니스 요구 사항을 시스템이 만족시키지 못할 수도 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 예약하여 아키텍처 및 프로세스가 어떻게 작동하는지 테스트합니다. 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, 신뢰성, 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
- 신뢰성
- 성능 효율성
- 비용 최적화
- 지속 가능성

운영 우수성

운영 우수성(OE)은 소프트웨어를 올바르게 구축하는 동시에 지속적으로 우수한 고객 경험을 제공하기 위한 노력입니다. 운영 우수성 원칙에는 팀 구성, 워크로드 설계, 규모에 따른 운영, 장기적 발전에 관한 모범 사례가 포함되어 있습니다.

운영 우수성 원칙에서는 설계 원리 개요, 모범 사례, 질문 사항을 제공합니다. 구현에 대한 권장 가이드는 운영 우수성 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에서 운영 우수성을 달성하기 위한 설계 원칙은 다음과 같습니다.

- 비즈니스 성과를 중심으로 팀 구성: 비즈니스 성과를 달성하는 팀의 역량은 리더십 비전, 효과적인 운영, 비즈니스에 맞는 운영 모델에서 비롯됩니다. 경영진은 팀이 가장 효율적인 방식으로 운영하고 비즈니스 성과를 달성하도록 장려하는 적절한 클라우드 운영 모델을 활용하여 CloudOps 혁신에 전적으로 투자하고 전념해야 합니다. 적절한 운영 모델은 규모 조정 및 최적화로 생산성을 높이고 민첩성, 대응성, 적응을 통한 차별화를 위해 인력, 프로세스 및 기술 역량을 사용합니다. 조직의 장기적 비전은 목표에 반영되고 목표는 기업 전반의 이해관계자 및 클라우드 서비스 소비자에게 전달됩니다. 목표와 운영 KPI는 모든 수준에서 연계됩니다. 이러한 관행은 다음과 같은 설계 원칙을 구현함으로써 얻을 수 있는 장기적 가치를 뒷받침합니다.
- 실행 가능한 인사이트를 위한 관찰성 구현: 워크로드 동작, 성능, 신뢰성, 비용 및 상태를 포괄적으로 이해할 수 있습니다. 핵심 성과 지표(KPI)를 설정하고 관찰성 원격 측정을 활용하여 정보에 입각한 결정을 내리고 비즈니스 성과가 위험에 처했을 때 즉각적인 조치를 취합니다. 실행 가능한 관찰성 데이터를 기반으로 성능, 신뢰성, 비용을 선제적으로 개선합니다.
- 가능한 경우 안전하게 자동화: 애플리케이션 코드를 위해 사용하였던 엔지니어링 원칙을 클라우드에서 인프라를 포함한 환경에 적용할 수 있습니다. 전체 워크로드와 해당 작업(애플리케이션, 인프라, 구성, 프로시저)을 코드로 정의하고 업데이트할 수 있습니다. 그런 다음 이벤트에 대한 응답으로 워크로드 작업을 시작하여 워크로드 작업을 자동화할 수 있습니다. 클라우드에서는 속도 제어, 오류 임곗값, 승인을 비롯한 가드레일을 구성하여 자동화 안전을 실현할 수 있습니다. 효과적인 자동화를 통해 이벤트에 일관되게 대응하고, 인적 오류를 제한하며, 작업자 수고를 줄일 수 있습니다.
- 되돌릴 수 있는 소규모 변경 자주 적용: 구성 요소를 정기적으로 업데이트할 수 있도록 확장 가능하고 느슨하게 결합된 워크로드를 설계합니다. 자동화된 배포 기법과 소규모의 점진적인 변경을 함께 사용하면 영향 반경을 줄이고 장애 발생 시 더 빠르게 되돌릴 수 있습니다. 이를 통해 품질을 유지하고 시장 상황의 변화에 신속하게 적응하면서 워크로드에 유익한 변화를 가져올 수 있다는 자신감이 높아집니다.
- 수시로 운영 절차 개선: 워크로드가 발전함에 따라 운영도 적절하게 개선합니다. 운영 절차를 사용할 때 개선할 여지가 있는지 확인합니다. 정기적으로 검토하여 모든 절차가 효과적이며 팀이 이러한 절차에 익숙한지 확인하고 검증합니다. 격차가 확인되면 그에 따라 절차를 업데이트합니다. 절차 업데이트를 모든 이해관계자와 팀에 전달합니다. 운영을 게임화하여 모범 사례를 공유하고 팀을 교육합니다.
- 장애 예측: 워크로드의 위험 프로필 및 비즈니스 성과에 미치는 영향을 이해하기 위해 실패 시나리오를 유도하여 운영 성공을 극대화합니다. 시뮬레이션에서 확인한 장애에 대한 절차의 효과와 팀의 대응을 테스트합니다. 테스트를 통해 확인된 미해결 위험을 관리하기 위해 정보에 입각한 결정을 내립니다.

- 모든 운영 이벤트 및 지표에서 학습: 모든 운영상 이벤트 및 실패로부터 파악한 내용을 통해 개선합니다. 파악한 내용을 팀 전반과 조직 전체에 공유합니다. 파악한 내용에서 운영이 비즈니스 성과에 어떻게 기여하는지에 대한 데이터와 일화를 강조해야 합니다.
- 관리형 서비스 사용: 가능한 경우 AWS 관리형 서비스를 사용하여 운영 부담을 줄입니다. 해당 서비스와의 상호 작용을 중심으로 운영 절차를 구축합니다.

정의

클라우드의 운영 우수성에는 4가지 모범 사례 영역이 있습니다.

- 조직
- 준비
- 운영
- 개선

조직의 리더십이 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, 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 비즈니스와 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

> **Note**
> 운영 우수성과 관련된 모든 질문에는 이 원칙의 약어인 OPS가 맨 앞에 표시됩니다.

주제

- 조직

- 준비
- 운영
- 개선

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 운영을 개선하는 과정에서 최대한의 이점을 얻을 수 있습니다. 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 내외부 고객 요구 사항을 평가하여 주력할 영역을 결정합니다. 고객 요구 사항을 평가하면 비즈니스 성과를 달성하는데 어떤 지원이 필요한지 철저하게 파악할 수 있습니다. 특정 초점을 의무화하거나 강조할 수 있는 규제 준수 요구 사항과 업계 표준과 같은 외부 요인과 조직의 거버넌스로 정해진 지침 또는 의무를 알고 있는지 확인합니다. 내부 거버넌스와 외부 규정 준수 요구 사항의 변경 사항을 식별할 수 있는 메커니즘이 있는지 확인합니다. 요구 사항이 식별되지 않았다는 결론을 내릴 때는 신중하게 판단하여 내린 결론인지 재차 확인해야 합니다. 정기적으로 우선순위를 검토하여 요구 사항의 변화에 따라 순위를 변경합니다.

비즈니스에 대한 위협 요소(예: 비즈니스상의 위험 및 법적 책임, 정보 보안 위협)를 평가하고 위험 목록에서 이 정보를 관리합니다. 위험의 영향과 상충하는 이해관계나 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 있습니다. 하지만 감수할 수 없는 경우에는 이를 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, 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, 인프라 구성 요소마다 소유자가 명시되어 있고, 각 프로세스와 절차를 정의하고 실행하는 소유자가 각각 명시되어 있는지 확인합니다.

각 구성 요소, 프로세스, 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. 참여하는 고위 리더십이 기대치를 설정하고 성공 여부를 측정해야 합니다. 고위 리더십은 조직이 발전하고 모범 사례를 도입하도록 하는 감독이자 후원자이며 지지자입니다. 성과를 달성할 수 없는 위험한 상태일 때 팀원이 그 영향을 최소화할 수 있도록 조치를 취하게 하고 위험하다고 판단될 때는 문제를 해결하고 사고를 방지할 수 있도록 의사 결정권자와 이해관계자에게 에스컬레이션하도록 합니다. 팀원이 적시에 적절한 조치를 취할 수 있도록 알려진 위험과 계획된 이벤트와 관련하여 시기 적절하고 명확하게 대화하며 실행 가능한 부분을 알려줍니다.

실험을 권장하여 학습을 가속화하고 팀원의 관심과 참여를 유지합니다. 팀은 새로운 기술을 도입하고 요구 사항과 책임이 변했을 때 이를 지원할 수 있도록 기술을 발전시켜야 합니다. 학습을 위한 시간을 따로 지정하여 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, 접근성을 높여 유익한 관점을 확보합니다.

조직에 적용되는 외부 규제 또는 규정 준수 요구 사항이 있다면 팀원이 우선순위에 대한 영향을 확인할 수 있도록 AWS 클라우드 규정 준수에서 제공하는 리소스를 사용하여 관련 정보를 제공해야 합니다. Well-Architected Framework에서는 학습, 평가, 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 규모를 조정 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 Advisor은 우선순위 결정에 도움이 될 수 있는 최적화 방안을 알려 주는 핵심 검사 세트를 이용할 수 있는 도구입니다. Business 및 Enterprise Support 고객에게는 우선순위를 더욱 세세히 결정하는 데 사용할 수 있는 검사 기능이 추가로 제공됩니다. 이러한 기능을 사용하면 보안, 신뢰성, 성능, 비용 최적화, 지속 가능성 영역을 중점적으로 확인할 수 있습니다.

AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS와 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 교육 및 자격증에서

는 AWS 기초에 관한 자습형 디지털 과정을 통해 일부 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 제공업체(예: AWS Managed Services, AWS Managed Services 파트너 또는 AWS 파트너 네트워크의 관리형 서비스 제공업체)를 통해 클라우드 환경을 전문적으로 구현할 수 있으며, 보안 및 규정 준수 요구 사항과 비즈니스 목표를 지원받을 수 있습니다. 관리형 서비스를 운영 모델에 추가하면 시간과 리소스를 절약할 수 있으며, 새로운 기술과 기능을 개발하는 대신 내부 팀이 비즈니스를 차별화하는 전략적 결과에 집중할 수 있습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다. (운영 우수성 질문 및 모범 사례 목록은 부록을 참조하세요.)

OPS 1: 회사에서 우선순위를 어떻게 결정하나요?

모든 사람이 비즈니스 성공을 달성하는 데 있어 자신의 역할을 이해해야 합니다. 리소스 우선순위 설정을 위한 공동의 목표가 있어야 합니다. 그러면 운영을 개선하려는 노력의 이점을 극대화할 수 있습니다.

OPS 2: 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성하나요?

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, 의사 결정권자를 파악하면 역량을 집중하고 팀의 이점을 극대화할 수 있습니다.

OPS 3: 조직 문화는 비즈니스 성과를 어떻게 지원하나요?

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 그에 따라 변경합니다. 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 실험에 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 얻을 수 있도록 워크로드를 설계합니다. 관찰성은 단순한 모니터링을 넘어서서 외부 출력을 기반으로 시스템의 내부 작동을 포괄적으로 이해할 수 있게 합니다. 지표, 로그, 추적에 기반을 둔 관찰성을 통해 시스템 동작과 역학에 대한 심층적인 인사이트를 얻을 수 있습니다. 효과적인 관찰성을 통해 팀은 패턴, 이상 및 추세를 식별하여 잠재적 문제를 사전에 해결하고 최적의 시스템 상태를 유지할 수 있습니다. 모니터링 활동과 비즈니스 목표를 일치시키기 위해서는 핵심 성과 지표(KPI)를 식별하는 것이 매우 중요합니다. 이러한 조정을 통해 팀은 진정으로 중요한 지표를 사용하여 데이터를 기반으로 결정을 내리고 시스템 성능과 비즈니스 결과를 모두 최적화할 수 있습니다. 또한 관찰성을 통해 기업은 사후 대응이 아닌 사전 대응이 가능합니다. 팀은 단순히 대응하는 데 그치지 않고 시스템 내의 인과 관계를 이해하여 문제를 예측하고 예방할 수 있습니다. 워크로드가 진화함에 따라 관찰성 전략을 재검토하고 개선하여 관련성과 효율성을 유지하는 것이 중요합니다.

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질에 대한 빠른 피드백, 버그 수정을 지원해야 합니다. 이러한 방식을 도입하면 유용한 변경 사항을 프로덕션 환경으로 빠르게 전달할 수 있고 문제가 퍼질 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 변경 사항을 적용한 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 테스트하고 확인할 수 있도록 부적절한 변경 사항을 처리할 계획을 세웁니다. 계획된 활동에 변경 사항이 미치는 위험을 제어할 수 있도록 환경 내에서 일어날 활동을 알고 있어야 합니다. 되돌릴 수 있도록 조금씩 자주 변경 사항을 적용하도록 변경 범위를 제한합니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변

경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다. 또한 중요한 변경 사항의 이점을 더 자주 누릴 수 있기도 합니다.

워크로드, 프로세스, 절차, 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다. 수동 또는 자동화된 체크리스트 등 일관된 프로세스를 사용하여 워크로드 또는 변경에 대응할 수 있는 준비가 되었는지 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 코드를 통해 운영하면 안전하게 실험하여 워크로드와 운영 절차를 개발하고 실패를 연습할 수 있습니다. AWS CloudFormation을 사용하면 운영 제어 수준을 점점 향상할 수 있는 일관된 템플릿 형식의 샌드박스 개발, 테스트, 생산 환경을 갖출 수 있습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 4: 워크로드에 어떻게 관찰성을 구현하나요?

워크로드에 관찰성을 구현하여 상태를 파악하고 비즈니스 요구 사항에 따라 데이터 기반 결정을 내릴 수 있습니다.

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

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식으로 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정이 가능합니다. 이렇게 하면 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 최소화하나요?

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 있나요?

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

코드를 통해 운영 활동을 구현하여 운영 인력의 생산성을 최대화하고 오류율을 최소화하며 대응을 자동화할 수 있습니다. 해당하는 경우에는 '사전 분석' 기능을 사용하여 장애를 예측하고 절차를 생성합니다. 리소스 태그와 AWS Resource Groups를 사용하여 메타데이터를 적용하고 일관된 태그 지정 전략을 시행하면 리소스를 식별할 수 있습니다. 조직, 비용 회계, 액세스 제어의 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 빠르게 구현합니다. 워크로드를 평가하는 데 사용하는 체크리스트를 변경할 때는 해당 변경으로 인해 더 이상 규정을 준수하지 못하게 되는 사용 중인 시스템은 어떻게 할 것인지 계획합니다.

운영

관찰성을 통해 의미 있는 데이터에 집중하고 워크로드의 상호 작용과 결과를 이해할 수 있습니다. 필수 인사이트에 집중하고 불필요한 데이터를 제거함으로써 워크로드 성능을 이해할 수 있는 간단한 접근 방식을 유지할 수 있습니다. 데이터를 수집하는 것뿐만 아니라 데이터를 올바르게 해석하는 것도 중요합니다. 명확한 기준을 정의하고, 적절한 경고 임곗값을 설정하며, 편차를 적극적으로 모니터링합니다. 특히 다른 데이터와 관계가 있는 경우 주요 지표의 변화로 특정 문제 영역을 정확히 찾아낼 수 있습니다. 관찰성을 사용하면 잠재적 문제를 더 잘 예측하고 해결하여 워크로드를 원활하게 운영하고 비즈니스 요구 사항을 충족할 수 있습니다.

워크로드 운영의 성공은 비즈니스 및 고객 성과 달성에 따라 측정됩니다. 예상 결과를 정의하고 성공을 측정하는 방법을 결정하며 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. 비즈니스와 고객에게 미치는 영향을 기반으로 이벤트 대응의 우선순위를 지정합니다. 이벤트 대응에 경고가 발생하는지 연결된 실행 프로세스가 있는지를 담당자와 함께 확인합니다. 이벤트를 해결하는 데 필요한 인력을 미리 정하고 에스컬레이션 프로세스를 포함하여 필요할 경우 긴급성과 영향을 기반으로 추가

인력을 배치합니다. 권한이 있는 개인을 식별하고 참여시켜 이전에 해결되지 않은 이벤트 대응에 대해 대응 과정이 비즈니스에 영향을 미쳤는지 확인합니다.

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

AWS에서는 AWS의 기본 지표와 워크로드에서 수집된 지표가 나와 있는 대시보드 보기를 생성할 수 있습니다. CloudWatch 또는 서드파티 애플리케이션을 활용하여 운영 활동의 비즈니스, 워크로드, 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 수정을 지원합니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 8: 워크로드 관찰성을 어떻게 활용하나요?
관찰성을 활용하여 워크로드 상태를 최적화합니다. 관련 지표, 로그, 추적을 활용하여 워크로드 성능을 종합적으로 파악하고 문제를 효율적으로 해결합니다.

OPS 9 운영 상태를 어떻게 파악하나요?
운영 지표를 정의, 캡처 및 분석하면 운영 이벤트에 대한 가시성을 확보하여 적절한 조치를 취할 수 있습니다.

OPS 10: 워크로드 및 운영 이벤트를 어떻게 관리하나요?
이벤트로 인해 워크로드가 중단될 가능성을 최소화할 수 있도록 이벤트 대응을 위한 절차를 준비하고 검증합니다.

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 공유하며 지속적으로 개선해야 합니다. 거의 연속적이고 서서히 개선을 이뤄내는 데에 주력하여 작업 주기를 조절합니다. 고객에게 영향을 미치는 모든 이벤트의 사후 분석을 수행합니다. 재발 제한 또는 방지를 위한 기여 요인과 예방 조치를 파악합니다. 영향을 받는 커뮤니티와 함께 기여 요소를 적절히 알립니다. 워크로드와 운영 절차 모두를 포함하여 개선할 부분(예: 기능 요청, 문제 해결, 규정 준수 요구 사항)을 정기적으로 평가하고 우선순위를 조정합니다.

절차 내에 피드백 루프를 포함시켜 개선할 영역을 빠르게 식별하고 실행을 통해 학습한 교훈을 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 장기 보관을 위해 Amazon S3로 로그를 직접 전송할 수 있습니다. AWS Glue를 사용하면 분석을 위해 Amazon S3의 로그 데이터를 검색 및 준비하여 AWS Glue Data Catalog에 관련된 메타데이터를 저장할 수 있습니다. 그리고 Amazon Athena에서 AWS Glue와의 기본 통합을 통해 로그 데이터를 분석하고 표준 SQL을 사용해 쿼리할 수 있습니다. Amazon QuickSight와 같은 비즈니스 인텔리전스 도구를 사용하면 데이터를 시각화하고 탐색하며 분석할 수 있습니다. 개선을 이끌 추세와 관심 이벤트를 찾습니다.

다음은 운영 우수성 고려 사항에 중점을 둔 질문입니다.

OPS 11: 운영을 어떻게 지속적으로 개선하나요?
시간과 리소스를 할애하여 점진적 개선을 거의 지속적으로 수행하면 운영의 효과와 효율성을 높일 수 있습니다.

성공적인 운영 개선은 잦은 소규모 개선, 안전한 환경 제공, 실험과 개발, 테스트 개선을 위한 시간 제공 그리고 실패로부터 학습을 독려하는 환경을 통해 이루어집니다. 샌드박스, 개발, 테스트, 생산 환경에 대한 운영 지원을 통해 운영 제어 수준을 점점 높아지도록 하고 개발을 촉진하며 생산 단계에 배포된 변경에서 성공적인 결과를 예측할 수 있도록 합니다.

리소스

운영 우수성 모범 사례에 대한 자세한 내용은 다음 리소스를 참조하세요.

설명서

- DevOps 및 AWS

백서

- 운영 우수성 원칙

비디오

- Amazon의 DevOps

보안

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

보안 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현에 대한 권장 가이드는 보안 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 워크로드 보안을 강화할 수 있는 여러 가지 원칙이 존재합니다.

- 강력한 자격 증명 기반 구현: 권한을 최소화한 보안 주체를 구현하고 AWS 리소스와의 각 상호 작용에 대한 적절한 권한을 부여하여 업무를 분리합니다. 자격 증명 관리를 중앙 집중화하고 장기적인 정적 자격 증명에 대한 의존도를 해소하는 것을 목표로 합니다.
- 추적 기능 유지 관리: 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 제어와 함께 심층 방어 접근 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 안전한 확장 능력을 빠르고 비용 효율적으로 향상시킵니다. 버전 제어가 가능한 템플릿에서 코드로 정의되고 관리되는 제어 기능의 구현을 비롯한 보안 아키텍처를 생성합니다.
- 전송 및 보관 중인 데이터 보호: 데이터를 민감도 수준에 따라 분류하고 적절한 경우 암호화, 토큰화 및 액세스 제어와 같은 메커니즘을 사용합니다.
- 사람들이 데이터에 쉽게 접근할 수 없도록 유지: 데이터에 대한 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 통해 사고에 대비합니다. 인시던트 대응 시뮬레이션을 실행하고 자동화된 도구를 사용하여 감지, 조사 및 복구 속도를 높입니다.

정의

클라우드의 보안에는 7가지 모범 사례 영역이 있습니다.

- 보안 기초
- 자격 증명 및 액세스 관리
- 탐지
- 인프라 보호
- 데이터 보호
- 인시던트 대응
- 애플리케이션 보안

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 고객의 보안 및 규정 준수 목표를 이루는데 도움을 줍니다. 클라우드 서비스를 뒷받침하는 인프라를 AWS가 물리적으로 보호해 주기 때문에 AWS 고객들은 서비

스를 이용하여 목표를 달성하는 데 집중할 수 있습니다. 또한 AWS 클라우드에서는 보안 데이터에 더 폭넓게 액세스할 수 있으며 보안 이벤트에 대한 응답도 자동화되어 있습니다.

모범 사례

주제

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

보안

다음은 보안 고려 사항에 중점을 둔 질문입니다. (보안 질문 및 모범 사례 목록은 부록을 참조하세요.)

SEC 1: 워크로드를 안전하게 운영하려면 어떻게 해야 하나요?

워크로드를 안전하게 운영하려면 모든 보안 영역에 중요한 모범 사례를 적용해야 합니다. 운영 우수성에 대해 조직 및 워크로드 수준에서 정의한 요구 사항 및 프로세스를 모든 영역에 적용하세요.

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 자동화하면 보안 운영을 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 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, 다중 인증(MFA) 등 강력한 암호를 요구할 수 있는 기능도 있습니다. 기존의 디렉터리 서비스와 연동되도록 할 수도 있습니다. 시스템이 AWS에 액세스해야 하는 워크로드의 경우 IAM이 인스턴스 프로필, 자격 증명 연동, 임시 자격 증명을 통해 보안 액세스를 보장합니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 2: 사람과 시스템에 대한 자격 증명은 어떻게 관리하나요?

보안 AWS 워크로드를 운영할 때는 두 가지 유형의 ID를 관리해야 합니다. 액세스 권한을 관리하고 부여하는 데 필요한 자격 증명의 유형을 이해하면 적절한 자격 증명이 적절한 조건에서 적절한 리소스에 액세스할 수 있도록 보장할 수 있습니다.

사람 ID: 관리자, 개발자, 운영자 및 최종 사용자가 AWS 환경 및 애플리케이션에 액세스하려면 ID가 필요합니다. 조직의 구성원 또는 웹 브라우저, 클라이언트 애플리케이션 또는 대화형 명령줄 도구를 통해 AWS 리소스와 상호 작용하는 외부 사용자입니다.

시스템 자격 증명: 서비스 애플리케이션, 운영 도구 및 워크로드에서 AWS 서비스에 요청하려면(예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 ID에는 AWS 환경에서 실행되는 시스템이 포함됩니다(예: Amazon EC2 인스턴스 또는 AWS Lambda 함수). 또한 액세스 권한이 필요한 외부 당사자를 위해 시스템 ID를 관리할 수도 있습니다. 또한 AWS 외부에 AWS 환경에 대한 액세스 권한이 필요한 시스템이 있을 수도 있습니다.

SEC 3: 사람과 시스템에 대한 권한은 어떻게 관리하나요?

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 직접 호출을 포함한 프로그래밍 방식의 액세스는 AWS Security Token Service에서 발행한 것과 같은 임시 및 제한된 권한 자격 증명을 사용하여 수행해야 합니다.

사용자가 AWS Management Console 외부에서 AWS와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. 프로그래밍 방식으로 액세스를 부여하는 방법은 AWS에 액세스하는 사용자 유형에 따라 다릅니다.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 AWS IAM Identity Center을 사용하도록 AWS CLI 구성을 참조하세요. • AWS SDK, 도구,AWS API에 대해서는 AWS SDK 및 도구 참조 가이드에서 IAM Identity Center 인증을 참조하세요.
IAM	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	IAM 사용자 설명서의 AWS 리소스와 함께 임시 보안 인증 정보 사용에 나와 있는 지침을 따르세요.
IAM	(권장되지 않음) 장기 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 IAM 사용자 보안 인증 정보를 사용한 인증을 참조하세요.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		• AWS SDK와 도구에 대해서는 AWS SDK 및 도구 참조 가이드에서 장기 보안 인증 정보를 사용한 인증을 참조하세요. • AWS API에 대해서는 IAM 사용자 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.

AWS는 Identity and Access Management를 사용하여 도울 수 있는 리소스를 제공합니다. 모범 사례를 배우려면 자격 증명 및 인증 관리, 사용자 액세스 제어, 프로그래밍 방식 액세스 제어에 대한 실습을 살펴보세요.

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 과정을 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 및 경보)을 처리하여 탐지 제어를 구현할 수 있습니다. CloudTrail 로그, AWS API 직접 호출 및 CloudWatch는 경보와 함께 측정치 모니터링을 제공하며 AWS Config는 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(S3)를 사용하여 액세스 요청을 기록할 수 있습니다.

다음은 보안 고려 사항에 중점을 둔 질문입니다.

SEC 4: 보안 관련 이벤트를 어떻게 감지하고 조사하나요?

로그와 지표에서 이벤트를 캡처하고 분석하여 가시성을 확보합니다. 보안 이벤트 및 잠재적 위협에 대해 조치를 취해 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 대응하는 것이 매우 중요합니다. AWS는 데이터 보존 기간 또는 데이터 보존, 아카이브 또는 삭제 위치를 정의하는 기능을 고객에게 부여함으로써 로그 관리를 보다 쉽게 구현할 수 있도록 합니다. 이렇게 하면 더 단순하고 경제적인 방식으로, 예측 가능하고 신뢰할 수 있도록 데이터를 처리할 수 있습니다.

인프라 보호

모범 사례와 업계 규정 또는 규제 의무를 준수하기 위해서는 인프라 보호가 필요하며, 여기에는 심층 방어와 같은 제어 방법이 포함됩니다. 지속적으로 클라우드 또는 온프레미스에서 작업을 성공적으로 수행하려면 반드시 이러한 방법을 사용해야 합니다.

AWS에서는 AWS 기본 기술을 사용하거나 AWS Marketplace에서 제공되는 파트너 제품 및 서비스를 사용하여 상태 저장(Stateful) 및 상태 비저장(Stateless) 방식의 패킷 검사를 구현할 수 있습니다. Amazon Virtual Private Cloud(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 보호하나요?

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 리소스는 어떻게 보호하나요?

워크로드의 컴퓨팅 리소스를 외부 및 내부 위협으로부터 보호할 수 있는 다중 방어 계층이 필요합니다. 컴퓨팅 리소스에는 EC2 인스턴스, 컨테이너, AWS Lambda 함수, 데이터베이스 서비스, IoT 디바이스 등이 포함됩니다.

어떤 환경이든 다중 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)로 이러한 구성을 유지할 수 있습니다. 이렇게 하면 Auto Scaling에 의한 시작되거나 수동으로 시작된 모든 경우에 이 AMI로 시작되는 모든 새 가상 서버(인스턴스)가 이 강화된 구성을 얻게 됩니다.

데이터 보호

시스템을 설계하려면 먼저 보안과 관련된 기본적인 관행부터 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 관행으로 데이터 보호를 실현합니다.

- AWS 고객은 데이터에 대한 완전한 통제력을 유지합니다.
- AWS는 정기적인 키 교체 등 키 관리 및 데이터 암호화를 더 간편하게 처리하도록 만듭니다. AWS 서비스를 이용하거나 사용자가 직접 관리하여 손쉽게 자동화할 수 있습니다.
- 파일 액세스, 변경 사항 등 중요한 콘텐츠가 수록된 상세 로그를 확인할 수 있습니다.
- AWS는 탁월한 복원성을 목표로 스토리지 시스템을 설계했습니다. 예를 들어 Amazon S3 Standard, S3 Standard–IA, S3 One Zone-IA 및 Amazon Glacier는 지정된 기간에 객체에 대해 99.999999999%의 내구성을 제공할 수 있도록 설계되었습니다. 이 내구성은 연평균 0.000000001%의 객체 손실 수준으로 예측됩니다.
- 광범위한 데이터 수명 주기 관리 프로세스에 포함될 수 있는 버전 관리는 우발적인 덮어쓰기나 삭제 및 그와 유사한 손해를 방지할 수 있습니다.
- AWS는 절대로 지역 간 데이터 이동을 하지 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 분류하나요?

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

SEC 8: 저장 데이터는 어떻게 보호하나요?

여러 제어를 구현하여 무단 액세스 또는 처리 오류의 위험을 줄여 저장 데이터를 보호합니다.

SEC 9: 전송 중 데이터는 어떻게 보호하나요?

여러 제어를 구현하여 무단 액세스 또는 손실의 위험을 줄여 전송 중인 데이터를 보호합니다.

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

인시던트 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 게임 데이를 통해 인시던트 대응을 정기적으로 연습한다면 아키텍처가 적기에 조사 및 복구를 수용할 수 있게 할 수 있습니다.

AWS에서는 다음과 같은 관행으로 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 시작합니다.
- AWS CloudFormation을 사용하여 도구 및 '클린 룸'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 복구하나요?

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.

애플리케이션 보안

애플리케이션 보안(AppSec)은 개발하는 워크로드의 보안 속성을 설계, 구축 및 테스트하는 전반적인 프로세스를 설명합니다. 조직에 적절한 교육을 받은 직원이 있어야 하며, 빌드 및 릴리스 인프라의 보안 속성을 이해하고, 자동화를 사용하여 보안 문제를 식별해야 합니다.

소프트웨어 개발 수명 주기(SDLC) 및 릴리스 후 프로세스에 정기적으로 애플리케이션 보안 테스트를 수행하면 프로덕션 환경에 유입되는 애플리케이션 보안 문제를 식별, 수정 및 방지할 수 있는 구조화된 메커니즘을 갖출 수 있습니다.

애플리케이션 개발 방법에는 워크로드를 설계, 구축, 배포 및 운영할 때의 보안 제어 기능이 포함되어야 합니다. 이와 동시에 지속적으로 결함을 줄이고 기술 부채를 최소화하도록 프로세스를 조정하세요. 예를 들어 설계 단계에서 위협 모델링을 사용하면 설계 결함을 조기에 발견할 수 있으므로 기다렸다가 나중에 문제를 완화하는 것보다 수정이 더 쉽고 비용이 적게 듭니다.

SDLC에서 결함은 보통 일찍 해결해야 비용과 복잡성이 줄어듭니다. 문제를 해결하는 가장 쉬운 방법은 애초에 문제가 발생하지 않게 하는 것입니다. 따라서 위협 모델로 시작하면 설계 단계부터 올바른 결과에 집중하는 데 도움이 됩니다. AppSec 프로그램이 발전을 거듭하면서 자동화를 사용하여 수행되는 테스트의 양을 늘리고, 빌더에 대한 피드백의 충실도를 개선하며, 보안 검토에 필요한 시간을 줄일 수 있습니다. 이러한 모든 작업은 구축하는 소프트웨어의 품질을 개선하고, 프로덕션에 기능을 도입하는 속도를 높입니다.

이 구현 지침은 조직 및 문화, 파이프라인 자체 보안, 파이프라인 내부 보안, 종속성 관리라는 네 가지 영역에 중점을 둡니다. 각 영역은 구현할 수 있는 일련의 원칙을 제공하며, 워크로드를 설계, 개발, 구축, 배포 및 운영하는 방법을 아우르는 전체적인 관점을 제공합니다.

AWS에는 애플리케이션 보안 프로그램을 다룰 때 사용할 수 있는 여러 방법이 있습니다. 이러한 접근 방식 중 일부는 기술에 의존하며, 일부는 애플리케이션 보안 프로그램의 인력 및 조직 측면에 중점을 두고 있습니다.

다음은 애플리케이션 보안 고려 사항에 중점을 둔 질문입니다.

SEC 11: 설계, 개발 및 배포 수명 주기 전반에 걸쳐 애플리케이션의 보안 속성을 어떻게 통합하고 검증하나요?

인력 교육, 자동화를 사용한 테스트, 종속성 이해, 도구 및 애플리케이션의 보안 속성 검증은 프로덕션 워크로드에서 발생할 수 있는 보안 문제를 줄이는 데 도움이 됩니다.

리소스

보안 관련 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

설명서

- AWS 클라우드 보안
- AWS 규정 준수
- AWS 보안 블로그
- AWS Security Maturity Model

백서

- 보안 요소
- AWS Security Overview
- AWS Risk and Compliance

비디오

- AWS 보안: 연방 정부
- Shared Responsibility Overview

신뢰성

신뢰성 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.

신뢰성 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현에 대한 권장 가이드는 신뢰성 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 5가지 신뢰성 설계 원칙이 있습니다.

- 장애 자동 복구: 워크로드의 핵심 성과 지표(KPI) 모니터링을 통해 임곗값을 위반하면 자동화를 시작할 수 있습니다. 이러한 KPI는 서비스의 기술적 측면이 아닌 비즈니스 가치에 기반한 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 실제 장애 시나리오가 발생하기 전에 테스트하고 수정할 수 있는 장애 경로를 노출하여 위험을 줄여 줍니다.
- 수평적 스케일링을 통해 전체 워크로드 가용성 증대: 단일의 큰 리소스를 다수의 작은 리소스로 대체하여 단일 장애가 전체 워크로드에 미치는 영향을 축소합니다. 요청을 더 작은 리소스 여러 개로 분산시키면 공통의 장애 지점이 공유되지 않습니다.
- 용량 추정 중지: 워크로드에 대한 수요가 해당 워크로드의 용량을 넘어서는 리소스 부족 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 보다 효율적인 수준으로 수요를 충족할 수 있습니다. 클라우드에도 제한은 있지만 할당량을 어느 정도 제어하고 관리하는 것이 가능합니다(Service Quotas 및 제약 조건 관리 참조).
- 자동화를 통한 변경 관리: 인프라 변경은 자동화를 통해 수행되어야 합니다. 관리가 필요한 변경에는 자동화 변경이 포함되며, 이후에 이러한 변경을 추적하고 검토할 수 있습니다.

정의

클라우드의 신뢰성에는 4가지 모범 사례 영역이 있습니다.

- 기본
- 워크로드 아키텍처
- 변경 관리
- 장애 관리

신뢰성을 달성하려면 기반부터 시작해야 합니다. 이러한 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

- 기본
- 워크로드 아키텍처
- 변경 관리
- 장애 관리

기본

기반에 관한 요구 사항은 그 범위가 단일 워크로드 또는 프로젝트 이상으로 확장됩니다. 시스템을 설계할 때는 먼저 신뢰성을 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 변경할 수 있습니다.

다음은 신뢰성 고려 사항에 중점을 둔 질문입니다. (신뢰성 질문 및 모범 사례 목록은 부록을 참조하세요.)

REL 1: 서비스 할당량과 제약 조건은 어떻게 관리하나요?

클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 비율을 제한하여 서비스가 남용되지 않도록 하기 위해 존재합니다. 또한 리소스에도 제약이 따릅니다. 예를 들면, 광섬유 케이블을 통해 비트를 전송할 수 있는 속도나 물리적 디스크의 스토리지 용량 등이 있습니다.

REL 2: 네트워크 토폴로지는 어떻게 계획하나요?

워크로드는 여러 환경에 존재하는 경우가 많습니다. 여기에는 여러 클라우드 환경(퍼블릭 액세스 가능 및 프라이빗)과 기존 데이터 센터 인프라가 포함됩니다. 계획에는 시스템 내부 및 시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 IP 주소 관리 및 도메인 이름 확인과 같은 네트워크 고려 사항이 포함되어야 합니다.

워크로드 아키텍처

신뢰할 수 있는 워크로드는 소프트웨어와 인프라에 대한 사전 설계 결정에서 시작됩니다. 아키텍처 선택은 모든 Well-Architected 원칙에서 워크로드 동작에 영향을 미칩니다. 신뢰성을 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 신뢰성 모범 사례를 구현할 수 있습니다. 또한 개발자는 Amazon Builders' Library에서 Amazon의 소프트웨어 구축 및 운영 방법에 대해 자세히 알아볼 수 있습니다.

다음은 신뢰성 고려 사항에 중점을 둔 질문입니다.

REL 3: 워크로드 서비스 아키텍처는 어떻게 설계하나요?

서비스 지향 아키텍처(SOA) 또는 마이크로서비스 아키텍처를 사용하여 확장성과 신뢰성이 뛰어난 워크로드를 구축합니다. 서비스 지향 아키텍처(SOA)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 설계하나요?

분산 시스템은 통신 네트워크를 사용하여 서버 또는 서비스와 같은 구성 요소를 상호 연결합니다. 이러한 네트워크에서 데이터 손실이나 지연 시간이 발생하더라도 워크로드는 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 이러한 모범 사례는 장애를 예방하고 평균 고장 간격(MTBF)을 개선합니다.

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

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 평균 복구 시간(MTTR)이 개선됩니다.

변경 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 신뢰성 고려 사항에 중점을 둔 질문입니다.

REL 6: 워크로드 리소스는 어떻게 모니터링하나요?

로그와 지표는 워크로드 상태를 파악할 수 있는 강력한 도구입니다. 로그와 지표를 모니터링하고 임계값을 초과하거나 중요한 이벤트가 발생할 경우 알림을 보내도록 워크로드를 구성할 수 있습니다. 모니터링을 수행하면 워크로드가 저성능 임곗값을 초과하거나 장애가 발생할 때를 인식하고 이에 대응하여 자동으로 복구할 수 있습니다.

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

확장 가능한 워크로드는 리소스를 자동으로 추가 또는 제거할 수 있는 탄력성을 제공하여 리소스 공급이 특정 시점의 수요와 거의 일치하도록 합니다.

REL 8: 변경 사항은 어떻게 적용하나요?

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 해결하는 것이 어려워집니다.

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

장애 관리

통상적인 수준의 복잡한 시스템에는 장애가 발생하기 마련입니다. 신뢰성을 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 임곗값을 넘어서면 자동화된 작업을 시작하여 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 신뢰성 고려 사항에 중점을 둔 질문입니다.

REL 9: 데이터는 어떻게 백업하나요?

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

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

장애 격리는 구성 요소 또는 시스템 장애의 영향을 정의된 경계로 제한합니다. 올바르게 격리하면 경계 외부의 구성 요소는 장애의 영향을 받지 않습니다. 여러 장애 격리 경계에서 워크로드를 실행하면 장애에 대한 복원력이 향상될 수 있습니다.

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

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

REL 12: 신뢰성은 어떻게 테스트하나요?

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

REL 13: 재해 복구(DR)는 어떻게 계획하나요?

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.

정기적으로 데이터를 백업하고 백업 파일을 테스트하여 논리적 오류와 물리적 오류를 모두 복구할 수 있는지 확인합니다. 빈번한 워크로드 자동 테스트를 통해 장애 원인을 파악하고 복구 방식을 살펴보는 것이 장애 관리의 핵심입니다. 정기 일정에 따라 이 테스트를 수행하고, 중요한 워크로드 변경 이후에도 이 테스트가 시작되는지 확인해야 합니다. Recovery Time Objective(RTO), Recovery Point Objective(RPO) 및 KPI를 적극적으로 추적하여 특히 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

신뢰성 관련 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

설명서

- AWS 설명서
- AWS 글로벌 인프라
- AWS Auto Scaling: How Scaling Plans Work
- AWS Backup란 무엇입니까?

백서

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

성능 효율성

성능 효율성 원칙에는 클라우드 리소스를 성능 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력이 포함됩니다.

성능 효율성 원칙에서 설계 원칙 개요, 모범 사례, 질문 사항을 제공합니다. 구현에 대한 권장 가이드는 성능 효율성 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 5가지 성능 효율성 설계 원칙이 있습니다.

- 고급 기술의 대중화: 팀에서 고급 기술을 원활하게 구현할 수 있도록 복잡한 작업을 클라우드 공급업체에 위임합니다. IT 팀에 새로운 기술의 호스팅 및 실행에 대해 알아볼 것을 요청하는 대신 기술을

서비스 형태로 사용하는 것을 고려하세요. 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 프로비저닝과 관리가 아닌 제품 개발에 집중할 수 있습니다.

- 몇 분 안에 전 세계에 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.
- 테스트 횟수 증가: 자동화할 수 있는 가상 리소스를 활용하며 여러 가지 인스턴스, 스토리지 또는 구성에 대한 비교 테스트를 신속하게 수행할 수 있습니다.
- 기계에 대한 공감 고려: 클라우드 서비스가 어떻게 사용되는지 파악하고 항상 워크로드 목표에 부합하는 기술 접근 방식을 사용합니다. 예를 들어 데이터베이스 또는 스토리지 접근 방식을 선택할 때는 데이터 접근 패턴을 고려합니다.

정의

클라우드의 성능 효율성에는 5가지 모범 사례 영역이 있습니다.

- 아키텍처 선택
- 컴퓨팅 및 하드웨어
- 데이터 관리
- 네트워킹 및 콘텐츠 전송
- 프로세스 및 문화

고성능 아키텍처를 구축할 때는 데이터 기반 접근 방식을 취합니다. 개괄적 설계부터 리소스 유형 선택과 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 성능 개선을 위해 아키텍처에서 절충안을 구성합니다.

모범 사례

주제

- 아키텍처 선택
- 컴퓨팅 및 하드웨어
- 데이터 관리
- 네트워킹 및 콘텐츠 전송
- 프로세스 및 문화

아키텍처 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, 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 질문 및 모범 사례 목록은 부록을 참조하세요.)

PERF 1: 워크로드에 적합한 클라우드 리소스 및 아키텍처를 어떻게 선택하나요?
워크로드에서 보다 효과적인 성능을 얻으려면 종종 여러 접근 방식을 취해야 합니다. Well-Architected 시스템은 다수의 솔루션 및 기능을 사용하여 성능을 개선합니다.

컴퓨팅 및 하드웨어

특정 워크로드에 대한 최적의 컴퓨팅 선택은 애플리케이션 설계, 사용량 패턴 및 구성 설정에 따라 다를 수 있습니다. 아키텍처는 다양한 컴포넌트에 대해 서로 다른 컴퓨팅 옵션을 사용하고 다양한 기능을 활성화하여 성능을 개선할 수 있습니다. 아키텍처에 대해 잘못된 컴퓨팅 옵션을 선택하면 성능 효율성 저하로 이어질 수 있습니다.

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

- 인스턴스는 가상화된 서버이기 때문에 버튼을 누르거나 API를 호출하는 방법으로 기능을 변경할 수 있습니다. 클라우드에서는 리소스를 한번 결정하면 그대로 고정되는 것이 아니므로 다양한 서버 유형을 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.
- 컨테이너는 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.
- 함수는 실행하려는 코드에서 실행 환경을 추상화합니다. 예를 들어, AWS Lambda에서는 인스턴스를 실행하지 않고 코드를 실행할 수 있습니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 2: 워크로드에서 컴퓨팅 리소스를 선택하고 사용하는 방법은 무엇인가요?

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

데이터 관리

특정 시스템에 대한 최적의 스토리지 솔루션은 데이터 유형의 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 아카이브), 업데이트 빈도(WORM, 동적) 및 가용성과 내구성 제약 사항에 따라 다릅니다. Well-Architected 워크로드는 용도에 맞게 구축된 데이터 저장소를 사용하므로 다양한 기능을 통해 성능을 개선할 수 있습니다.

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

- 객체 스토리지는 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 스토리지는 일관되고 지연 시간이 짧은 고가용성 블록 스토리지를 각 가상 호스트에 제공하며, Direct-Attached Storage(DAS) 또는 Storage Area Network(SAN)와 유사합니다. 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 스토리지를 사용하면 여러 시스템에서 공유 파일 시스템에 액세스할 수 있습니다. Amazon Elastic File System(Amazon 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, 능률적으로 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: 워크로드의 데이터를 어떻게 저장, 관리, 액세스하나요?

시스템에 대한 보다 효율적인 스토리지 솔루션은 액세스 작업 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적), 가용성과 내구성 제약 사항에 따라 다릅니다. Well-Architected 시스템은 여러 스토리지 솔루션을 사용하며 다양한 기능을 설정하여 성능을 개선하고 리소스를 효율적으로 사용합니다.

네트워킹 및 콘텐츠 전송

워크로드에 대한 최적의 네트워킹 솔루션은 지연 시간, 처리량 요구 사항, 지터, 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.

AWS에서 네트워킹은 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 네트워킹 요구 사항에 보다 쉽게 부합할 수 있습니다. AWS에서는 제품 기능(예: 강화된 네트워킹, Amazon EC2 네트워킹 최적화 인스턴스, Amazon S3 Transfer Acceleration, 동적 Amazon CloudFront)을 제공하여 네트워크 트래픽을 최적화합니다. 또한 AWS Global Accelerator에서는 네트워킹 기능(예: Amazon Route 53 지연 속도 기반 라우팅, Amazon VPC 엔드포인트, AWS, AWS Direct Connect)도 제공하여 네트워크 거리 또는 지터를 줄입니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 4: 워크로드에서 네트워킹 리소스를 어떻게 선택하고 구성하나요?

이 질문에는 클라우드에서 효율적인 네트워킹 및 콘텐츠 전송 솔루션을 설계, 구성 및 운영하기 위한 지침과 모범 사례가 포함됩니다.

프로세스 및 문화

워크로드를 설계할 때는 효율적인 고성능 클라우드 워크로드를 더 잘 실행하는 데 도움이 되도록 채택할 수 있는 원칙과 관행이 있습니다. 클라우드 워크로드의 성능 효율성을 촉진하는 문화를 도입하려면 다음과 같은 주요 원칙과 관행을 고려하세요.

이러한 문화를 구축하기 위해 다음과 같은 핵심 원칙을 고려하세요.

- 코드형 인프라: AWS CloudFormation 템플릿 등의 접근 방식을 사용하여 코드형 인프라를 정의합니다. 템플릿을 사용하면 애플리케이션 코드와 구성과 함께 인프라를 소스 제어에 포함할 수 있습니다. 이렇게 하면 소프트웨어를 개발하는 데 사용하는 것과 동일한 사례를 인프라에 적용할 수 있으므로 검토를 빠르게 반복할 수 있습니다.
- 배포 파이프라인: 소스 코드 리포지토리, 빌드 시스템, 배포, 테스트 자동화 등의 지속적 통합 및 지속적 전달(CI/CD) 파이프라인을 사용하여 인프라를 배포합니다. 이렇게 하면 반복 가능하며 일관성 있는 저렴한 방식으로 배포를 반복해서 진행할 수 있습니다.
- 잘 정의된 지표: 핵심 성과 지표(KPI)를 캡처하기 위해 지표를 설정하고 모니터링합니다. 기술과 비즈니스 지표를 모두 사용하는 것이 좋습니다. 웹 사이트 또는 모바일 앱의 경우 주요 지표는 첫 번째 바이트가 수신되거나 첫 번째 렌더링이 완료될 때까지의 시간을 측정합니다. 그 외에 일반적으로 적용되는 지표에는 스레드 수, 가비지 수집 속도, 대기 상태 등이 있습니다. 요청당 누적 비용 집계액 등의 비즈니스 지표에서는 비용 절감 방법을 파악할 수 있습니다. 지표를 해석할 방법을 신중하게 고려합니다. 예를 들어 평균이 아닌 최대값이나 99번째 백분위수를 선택할 수 있습니다.
- 자동 성능 테스트: 배포 프로세스의 일환으로 더 빠르게 실행되는 테스트가 정상적으로 완료되고 나면 성능 테스트를 자동으로 시작합니다. 이 자동 테스트에서는 새 환경을 설정하고, 테스트 데이터 등의 초기 조건을 설정한 다음 일련의 벤치마크와 로드 테스트를 실행해야 합니다. 시간별 성능 변화를 추적할 수 있도록 이러한 테스트의 결과를 빌드에 다시 연결해야 합니다. 오래 실행되는 테스트의 경우에는 테스트를 빌드의 다른 부분과 비동기식으로 파이프라인에 포함할 수 있습니다. Amazon EC2 스팟 인스턴스를 사용하여 야간에 성능을 테스트할 수도 있습니다.
- 로드 생성: 가상 또는 사전 녹화 방식의 사용자 여정을 복제하는 일련의 테스트 스크립트를 생성해야 합니다. 이러한 스크립트는 항상 동일한 결과를 반환하고 결합되지 않아야 합니다. 유효한 결과를 얻기 위해 사전 준비 스크립트를 포함해야 할 수도 있습니다. 따라서 스크립트를 최대한 많이 테스트

하여 프로덕션 환경의 사용 동작을 복제하는 것이 좋습니다. 소프트웨어 또는 서비스형 소프트웨어(SaaS) 솔루션을 사용하여 로드를 생성할 수 있습니다. 비용 효율적인 방식으로 로드를 생성할 수 있도록 AWS Marketplace 솔루션 및 스팟 인스턴스를 사용하는 방법을 고려하세요.

- 성능 확인: 팀이 주요 지표(특히 각 빌드 버전 관련 지표)를 확인할 수 있도록 제공해야 합니다. 이렇게 하면 시간 경과에 따른 긍정적이거나 부정적인 주요 추세를 확인할 수 있습니다. 또한 오류나 예외 수 관련 지표도 표시하여 작동 중인 시스템을 테스트하고 있는지를 확인해야 합니다.
- 시각화: 성능 문제, 핫스팟, 대기 상태, 낮은 사용률 등이 확인되는 위치를 명확하게 표시하는 시각화 기술을 사용합니다. 아키텍처 다이어그램 위에 성과 지표를 겹쳐서 표시합니다. 콜 그래프나 코드는 문제를 빠르게 확인하는 데 도움을 줍니다.
- 정기적인 검토 프로세스: 아키텍처의 성능 저하는 성능 검토 프로세스가 없거나 효과적이지 않은 경우 주로 발생합니다. 아키텍처의 성능이 좋지 않은 경우 성능 검토 프로세스를 구현하면 반복적으로 개선을 주도할 수 있습니다.
- 지속적 최적화: 클라우드 워크로드의 성능 효율성을 지속적으로 최적화할 수 있는 문화를 채택합니다.

다음은 성능 효율성 고려 사항에 중점을 둔 질문입니다.

PERF 5: 워크로드의 성능 효율성을 높이기 위해 어떤 프로세스를 사용하나요?

워크로드를 설계할 때는 효율적인 고성능 클라우드 워크로드를 더 잘 실행하는 데 도움이 되도록 채택할 수 있는 원칙과 관행이 있습니다. 클라우드 워크로드의 성능 효율성을 촉진하는 문화를 도입하려면 다음과 같은 주요 원칙과 관행을 고려하세요.

리소스

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

설명서

- Amazon S3 성능 최적화
- Amazon EBS 볼륨 성능

백서

- 성능 효율성 원칙

비디오

- 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 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현에 대한 권장 가이드는 비용 최적화 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 5가지 비용 최적화 설계 원칙이 있습니다.

- 클라우드 재무 관리 구현: 클라우드에서 재무적 성공을 달성하고 비즈니스 가치 실현을 앞당기려면 클라우드 재무 관리 및 비용 최적화에 투자합니다. 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 사용할 필요 없이, 필요한 컴퓨팅 리소스에만 비용을 지불하고 비즈니스 요구 사항에 따라 사용량을 늘리거나 줄입니다. 예를 들어 개발 및 테스트 환경은 주로 주

중 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 결과와 워크로드 제공과 관련된 비용을 측정합니다. 이렇게 측정하여 성과 개선과 비용 절감으로 얻을 수 있는 이익을 확인하시기 바랍니다.
- 획일적인 업무 부담에 대한 비용 지출 중단: 랙 및 스택 설치와 서버 전원 공급 등 데이터 센터 운영의 힘든 작업을 AWS가 처리합니다. 또한 관리형 서비스를 통해 운영 체제 및 애플리케이션을 관리하는 운영 부담을 덜어줍니다. 따라서 IT 인프라가 아니라 고객과 비즈니스 프로젝트에 집중할 수 있습니다.
- 지출 분석 및 기여도 확인: 클라우드를 사용하면 손쉽게 시스템의 사용량과 비용을 정확하게 식별할 수 있어 개별 워크로드 소유자의 IT 비용 기여도를 투명하게 확인할 수 있습니다. 그 결과 투자 대비 수익률(ROI)을 측정할 수 있어 워크로드 소유자에게는 리소스를 최적화하고 비용을 절감하는 기회가 됩니다.

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. 경우에 따라서는 선결제 비용 최적화에 투자하는 것보다 출시 시간을 단축하거나, 새로운 기능을 배포하거나, 단순히 납기를 준수하는 등 속도를 기준으로 최적화하는 것이 가장 좋습니다. 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. 그러나 온프레미스 환경의 리소스를 클라우드로 '리프트 앤 시프트'한 후 최적화해야 하는 경우에는 이 선택이 적합합니다. 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
- 지출 및 사용량 인식
- 비용 효율적인 리소스
- 수요 관리 및 리소스 공급
- 시간 경과에 따른 최적화

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 방식이 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 구축하고 혁신하며, 더 민첩하게 대응하고, 내부 또는 외부 요인에 적응합니다.

AWS에서 비용 및 사용 보고서(CUR)와 함께 Cost Explorer와 Amazon Athena 및 Amazon QuickSight(선택 사항)를 사용하여 비용과 사용량에 대한 조직 전체의 인식을 높일 수 있습니다. AWS 예산에서는 비용과 사용량에 대한 사전 알림을 제공합니다. AWS 블로그는 새로운 서비스 릴리스를 숙지할 수 있도록 신규 서비스와 기능에 대한 정보를 제공합니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다. (비용 최적화 질문 및 모범 사례 목록은 부록을 참조하세요.)

COST 1: 클라우드 재무 관리를 어떻게 구현하나요?
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 최적화의 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 고려합니다.

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 협상하며, 주문 번호를 관리하고, 배송을 예약한 후 리소스를 배포하는 등의 온프레미스 인프라 프로비저닝과 연관된 시간 및 수동 프로세스를 줄입니다. 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 줄여줍니다. 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 만듭니다. 이를 통해 비용과 사용량의 분리와 할당이 가능합니다. 리소스 태그 지정을 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 관리하나요?

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: 비용과 사용량을 어떻게 모니터링하나요?

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

COST 4: 리소스를 어떻게 폐기하나요?

프로젝트 시작부터 마지막까지의 전체 과정에서 변경 제어 및 리소스 관리를 구현합니다. 이를 통해 미사용 리소스를 차단하여 낭비를 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. 심층적인 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 요구 사항을 가장 효과적으로 충족하는 Amazon EC2 및 다른 서비스의 인스턴스를 획득하도록 유연하고 비용 효율적인 요금 옵션을 매우 다양하게 제공합니다. 온디맨드 인스턴스의 경우 최소 약

정이 없으며 시간 단위로 컴퓨팅 파워 비용을 지불합니다. 절감형 플랜 및 예약형 인스턴스는 온디맨드 요금에 비해 최대 75% 할인된 요금을 제공합니다. 스팟 인스턴스를 사용하면 미사용 Amazon EC2 용량을 활용할 수 있으며 온디맨드 요금보다 최대 90% 할인된 요금을 제공합니다. 스팟 인스턴스는 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 줄일 수 있으며, Amazon Aurora on Amazon RDS를 활용하면 값비싼 데이터베이스 라이선싱 비용을 해소할 수 있습니다.

다음은 비용 최적화 고려 사항에 중점을 둔 질문입니다.

COST 5: 서비스를 선택할 때 비용을 어떻게 평가하나요?

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, 애플리케이션 및 비즈니스 관련 활동에 집중할 수 있습니다.

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

진행 중인 작업에 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 요금 모델을 어떻게 사용하나요?

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

COST 8: 데이터 전송 요금을 위한 계획은 어떻게 되나요?

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 사용하여 AWS 사용량을 정기적으로 검토함으로써 사용률을 적극적으로 모니터링하고 그에 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 줄어듭니다. 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 관리하나요?

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 고려하세요. 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 변경되면 더 이상 필요하지 않은 리소스, 전체 서비스 및 시스템을 과감하게 폐기하세요.

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 기

술을 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 평가하나요?

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 Amazon RDS를 사용하면 관계형 데이터베이스의 비용을 줄일 수 있습니다. Lambda와 같은 서버리스를 사용하면 코드를 실행하기 위해 인스턴스를 운영하고 관리할 필요가 없습니다.

COST 11: 작업 비용을 어떻게 평가하나요?

클라우드에서의 운영 비용을 평가하고, 시간이 많이 걸리는 클라우드 운영을 검토하며, 관련 AWS 서비스, 서드파티 제품 또는 사용자 지정 도구를 채택하여 인적 노력과 비용을 줄이도록 이를 자동화하합니다.

리소스

비용 최적화 관련 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

설명서

- AWS 설명서

백서

- 비용 최적화 요소

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. 구현에 대한 권장 가이드는 지속 가능성 원칙 백서에서 확인할 수 있습니다.

주제

- 설계 원칙
- 정의
- 모범 사례
- 리소스

설계 원칙

클라우드에는 6가지 지속 가능성 설계 원칙이 있습니다.

- 영향 파악: 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 설정: 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 단위에 대해 측정된 영향 강도를 줄입니다. 목표를 통해 비즈니스 또는 조직의 보다 광범위한 지속 가능성 목표를 지원하고, 회귀를 식별하며, 잠재적 개선 영역의 우선순위를 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, 처리 및 스토리지를 줄이거나 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 운영하고 효율적인 운영을 책임지는 서버리스 컨테이너용 AWS Fargate와 같은 관리형 서비스를 도입함으로써 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 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
- 수요에 맞춘 조정
- 소프트웨어 및 아키텍처
- 데이터
- 하드웨어 및 서비스
- 프로세스 및 문화

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 채택하며, 능률적인 데이터 스토리지 기술을 사용하여 적절한 크기의 효과적인 컴퓨팅 인프라를 배포하고 고성능 최종 사용자 하드웨어 요구 사항을 최소화하는 등 다양하게 나타날 수 있습니다.

모범 사례

주제

- 리전 선택
- 수요에 맞춘 조정
- 소프트웨어 및 아키텍처

- 데이터 관리
- 하드웨어 및 서비스
- 프로세스 및 문화

리전 선택

워크로드 리전의 선택은 성능, 비용 및 탄소 배출량을 포함한 KPI에 큰 영향을 미칩니다. 이러한 KPI를 개선하려면 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 워크로드의 리전을 선택해야 합니다.

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

SUS 1: 워크로드에 적합한 리전을 선택하려면 어떻게 해야 하나요?

워크로드 리전의 선택은 성능, 비용 및 탄소 배출량을 포함한 KPI에 큰 영향을 미칩니다. 이러한 KPI를 개선하려면 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 워크로드의 리전을 선택해야 합니다.

수요에 맞춘 조정

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. 사용자가 리소스를 소비하는 데 필요한 네트워크를 제한하도록 리소스를 배치합니다. 사용되지 않는 자산을 제거합니다. 팀원에게 지속 가능성에 미치는 영향을 최소화하면서 요구 사항을 지원하는 디바이스를 제공합니다.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 2: 클라우드 리소스를 비즈니스 요구에 어떻게 맞추나요?

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. 사용자가 리소스를 소비하는 데 필요한 네트워크를 제한하도록 리소스를 배치합니다. 사용되지 않는 자산을 제거합니다. 팀원에게 지속 가능성에 미치는 영향을 최소화하면서 요구 사항을 지원하는 디바이스를 제공합니다.

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

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

사용되지 않는 자산 생성 및 유지 관리 감소: 애플리케이션 자산(예: 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 줄입니다. 사용되지 않는 자산(예: 더 이상 판매되지 않는 제품 이미지)을 폐기하여 사용된 리소스를 해제하고 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

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 하나요?

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

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

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 소프트웨어 및 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 시간에 따른 개별 구성 요소의 사용률 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 소비한 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 사용하는 디바이스와 장비, 예상 수명 주기, 이러한 구성 요소 교체가 재정 및 지속 가능성에 미치는 영향을 이해합니다. 소프트웨어 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 소비하며, 전송 및 저장되는 방식을 이해합니다. 데이터 처리 및 스토리지 요구 사항을 최소화하는 기술을 선택합니다.

데이터 관리

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 4: 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 하나요?

데이터 관리 원칙을 구현하여 워크로드를 지원하는 데 필요한 프로비저닝된 스토리지와 이를 사용하는 데 필요한 리소스를 줄입니다. 데이터를 이해하고 데이터의 비즈니스 가치와 데이터 사용 방식

SUS 4: 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 하나요?

을 가장 효과적으로 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 생성합니다. 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.

필요하지 않은 데이터 또는 중복 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. 미사용 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 백업합니다. 백업 정책을 검토하고 복구 시나리오에서 가치를 제공하지 않는 임시 스토리지는 제외합니다.

하드웨어 및 서비스

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.

다음은 지속 가능성 고려 사항에 중점을 둔 질문입니다.

SUS 5: 지속 가능성 목표를 지원하기 위해 아키텍처에서 클라우드 하드웨어 및 서비스를 어떻게 선택하고 사용하나요?

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.

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, 동영상 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 디바이스(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.

프로세스 및 문화

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: 조직의 프로세스가 지속 가능성 목표를 어떻게 지원하고 있나요?

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

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

워크로드를 최신 상태로 유지: 최신 운영 체제, 라이브러리 및 애플리케이션을 통해 워크로드 효율성을 개선하고 보다 효율적인 기술을 더 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.

빌드 환경의 사용률 증가: 자동화와 코드형 인프라를 사용하여 필요 시 프로덕션 전 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.

테스트를 위해 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 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

백서

- 지속 가능성 원칙

비디오

- The Climate Pledge

검토 프로세스

아키텍처 검토는 일관적인 방식으로 수행하고 잘잘못을 따지지 않는 접근 방식을 취해야 심층적으로 분석할 수 있습니다. 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. 거의 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 RCA를 검토하여 사용 중인 검토 프로세스를 개선할 수 있는지 확인합니다.

변경이 어려운 단방향 도어 결정을 피하기 위해 실행 이전 설계 단계 초반에 제품 수명 주기의 주요 마일스톤에서 검토를 적용해야 합니다. (대부분의 의사 결정은 번복할 수 있는 양방향 방식으로 이루어집니다. 이러한 결정 과정에서는 간단한 프로세스를 사용할 수 있습니다. 단방향 도어는 번복하기 어렵거나 불가능하기 때문에 실행하기 전에 더 많은 검사가 필요합니다.) 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 활성화했나요?')

검토를 마친 후에는 비즈니스 상황에 따라 우선순위를 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 언급됨)
  - 대규모 출시를 준비하는 경우 원활하게 진행되길 바랄 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 언급됩니다.)
  - 이 이벤트는 이동할 수 없습니다. 아키텍처에 대한 위험을 파악하지 않은 채 그대로 진행하고 싶으신가요? 이러한 모든 문제를 해결하지 못하더라도, 구체화하는 경우 해당 문제를 처리하기 위한 지침서를 가질 수 있습니다.
- '다른 업체가 자사의 솔루션 구현 방식에 대한 비밀을 알게 되길 바라진 않습니다!'
  - Well-Architected Framework의 질문을 팀에게 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

AWS의 Well-Architected 프레임워크는 클라우드에서 신뢰할 수 있고 안전하며 효율적이고 경제적이면서 지속 가능한 시스템을 설계하고 운영하기 위한 여섯 가지 원칙의 아키텍처 모범 사례를 제공합니다. 이 프레임워크에서는 기존 아키텍처 또는 제안된 아키텍처를 검토할 수 있는 몇 가지 질문을 제공합니다. 그리고 각 원칙에 대한 AWS 모범 사례도 제시합니다. 아키텍처에 이 프레임워크를 사용하면 기능적 요구 사항에 초점을 둔 안정적이고 효율적인 시스템을 구축할 수 있습니다.

기여자

다음 개인과 조직이 이 문서에 기여했습니다.

- Brian Carlson, Amazon Web Services의 Operations Lead Well-Architected
- Ben Potter, Amazon Web Services의 Security Lead Well-Architected
- Seth Eliot, Amazon Web Services의 Reliability Lead Well-Architected
- Eric Pullen, Amazon Web Services의 Sr. Solutions Architect
- Rodney Lester, Amazon Web Services의 Principal Solutions Architect
- Jon Steele, Amazon Web Services의 Sr. Technical Account Manager
- Max Ramsay, Amazon Web Services의 Principal Security Solutions Architect
- Callum Hughes, Amazon Web Services의 Solutions Architect
- Ben Mergen, Amazon Web Services의 Senior Cost Lead Solutions Architect
- Chris Kozlowski, Amazon Web Services의 Enterprise Support, Senior Specialist Technical Account Manager
- Alex Livingstone, Amazon Web Services의 Cloud Operations, Principal Specialist Solutions Architect
- Paul Moran, Amazon Web Services의 Enterprise Support, Principal Technologist
- Peter Mullen, Amazon Web Services의 Professional Services, Advisory Consultant
- Chris Pates, Amazon Web Services의 Enterprise Support, Senior Specialist Technical Account Manager
- Arvind Raghunathan, Amazon Web Services의 Enterprise Support, Principal Specialist Technical Account Manager
- Sam Mokhtari, Amazon Web Services의 Senior Efficiency Lead Solutions Architect

참조 자료

AWS 아키텍처 센터

AWS 클라우드 규정 준수

AWS Well-Architected 파트너 프로그램

AWS Well-Architected Tool

AWS Well-Architected 홈페이지

운영 우수성 원칙 백서

보안 원칙 백서

신뢰성 원칙 백서

성능 효율성 원칙 백서

비용 최적화 원칙 백서

지속 가능성 원칙 백서

Amazon Builders' Library

문서 수정

이 백서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
메이저 업데이트	신뢰성, 보안, 운영 우수성, 지속 가능성 및 성능 효율성에 대한 새로운 지침으로 모범 사례가 업데이트되었습니다. 신뢰성 원칙의 많은 모범 사례가 대규모로 갱신되고 업데이트되었습니다. 새로운 서비스 및 생성형 AI 권장 사항으로 보안 및 운영 우수성에 대한 지침이 업데이트되고 개선되었습니다. 지속 가능성은 AWS 서비스와 새로운 모범 사례를 기반으로 여러 내용이 업데이트되었습니다.	2024년 11월 6일
메이저 업데이트	원칙 전반에 걸쳐 모범 사례가 대규모로 업데이트되었습니다. 보안과 비용 모두 새로운 모범 사례를 도입했습니다.	2024년 6월 27일
메이저 업데이트	주요 원칙 업데이트.	2023년 10월 3일
메이저 업데이트	권장 가이드 및 새로운 모범 사례를 추가하여 모범 사례를 업데이트했습니다. 보안 및 비용 최적화 원칙에 새로운 질문이 추가되었습니다.	2023년 4월 10일
마이너 업데이트	작업 수준에 대한 정의를 추가하고 부록에 모범 사례를 업데이트했습니다.	2022년 10월 20일

메이저 업데이트	지속 가능성 원칙을 추가하고 링크를 업데이트했습니다.	2021년 12월 2일
메이저 업데이트	지속 가능성 원칙이 프레임워크에 추가되었습니다.	2021년 11월 20일
마이너 업데이트	포용적이지 않은 표현을 삭제했습니다.	2021년 4월 22일
마이너 업데이트	여러 링크를 수정했습니다.	2021년 3월 10일
마이너 업데이트	문서 전반에 편집상 변경 사항을 적용했습니다.	2020년 7월 15일
메이저 업데이트	대부분의 질문과 답변을 검토하고 다시 작성합니다.	2020년 7월 8일
백서 업데이트	AWS Well-Architected Tool, AWS Well-Architected Labs 및 AWS Well-Architected 파트너 링크를 추가하고, 여러 언어 버전의 프레임워크를 지원하도록 마이너한 사항 몇 가지를 수정했습니다.	2019년 7월 1일
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.	2018년 11월 1일

백서 업데이트	업데이트를 통해 질문 텍스트를 더 단순하게 작성하고 답변을 표준화했으며 가독성을 개선했습니다.	2018년 6월 1일
백서 업데이트	운영 우수성이 원칙 맨 앞으로 이전되어 다시 작성되었으며, 이에 따라 다른 원칙의 구성도 바뀌었습니다. AWS의 발전을 반영하기 위해 다른 원칙이 수정되었습니다.	2017년 11월 1일
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.	2016년 11월 1일
마이너 업데이트	최신 Amazon CloudWatch Logs 정보로 부록을 업데이트했습니다.	2015년 11월 1일
최초 게시	AWS Well-Architected Framework를 게시했습니다.	2015년 10월 1일

> **Note**
> RSS 업데이트를 구독하려면 사용 중인 브라우저에서 RSS 플러그인을 활성화해야 합니다.

프레임워크 버전:

- 2024-06-27
- 2023-10-03
- 2023-04-10

- 2022-03-31

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(OE)은 소프트웨어를 올바르게 구축하는 동시에 지속적으로 우수한 고객 경험을 제공하기 위한 노력입니다. 운영 우수성 원칙에는 팀 구성, 워크로드 설계, 규모에 따른 운영, 장기적 발전에 관한 모범 사례가 포함되어 있습니다. 구현에 대한 권장 가이드는 운영 우수성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- Organization
- 준비
- 운영
- 개선

Organization

Questions

- OPS 1. 귀사의 운영 우선순위를 결정하는 요인은 무엇인가요?
- OPS 2. 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합하나요?
- OPS 3. 조직 문화는 비즈니스 성과를 어떻게 지원하나요?

OPS 1. 귀사의 운영 우선순위를 결정하는 요인은 무엇인가요?

모든 직원이 효율적인 업무 수행에서 맡은 역할을 파악해야 합니다. 리소스 우선순위 설정을 위한 공동의 목표가 있어야 합니다. 그러면 운영을 개선하려는 노력의 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 이점과 위험을 관리하면서 장단점 평가

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.

원하는 성과:

- 고객 성과에서 시작하여 역방향으로 작업합니다.
- 운영 관행이 비즈니스 성과 및 목표를 어떻게 지원하는지 이해합니다.
- 모든 관련 당사자를 참여시킵니다.
- 외부 고객의 요구를 파악할 수 있는 메커니즘이 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 확립의 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 사례가 확립되지 않을 경우 노출되는 위험 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 이해관계 수립: 워크로드의 비즈니스 기능과 워크로드 작동 과정에서 각 팀의 역할을 비롯하여 이러한 요소가 내외부 고객의 공동 비즈니스 목표를 지원하는 방식에 대한 공유된 이해관계를 정립합니다.

리소스

관련 모범 사례:

- OPS11-BP03 피드백 루프 구현

OPS01-BP02 내부 고객 요구 평가

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

원하는 성과:

- 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 우선순위를 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 확립의 이점: 내부 고객 요구를 평가하고 이해하면 비즈니스 가치를 제공하기 위해 작업의 우선순위를 정하는 방법을 알 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 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 이해관계자와 함께 내부 고객의 목표, 요구 사항 및 우선순위를 논의합니다. 이렇게 하면 비즈니스 및 고객 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
- 공유된 이해관계 수립: 워크로드의 비즈니스 기능과 워크로드 작동 과정에서 각 팀의 역할을 비롯하여 이러한 요소가 내외부 고객의 공동 비즈니스 목표를 지원하는 방식에 대한 공유된 이해관계를 정립합니다.

리소스

관련 모범 사례:

- OPS11-BP03 피드백 루프 구현

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 성과:

- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 비슷하지만 조직 외부에서 이루어집니다.

관련 문서:

- AWS Management and Governance Cloud Environment Guide
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- Governance in the AWS 클라우드: The Right Balance Between Agility and Safety
- 거버넌스, 위험 및 규정 준수(GRC)란 무엇인가요?

관련 비디오:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)

관련 예제:

- AWS Config Conformance Pack Samples

관련 서비스:

- AWS Config
- AWS Organizations - Service Control Policies

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 성과:

- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 및 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. 규정 준수 검증 및 감사 보고서를 위해 AWS Audit Manager와 같은 서비스를 사용합니다.
   b. AWS Artifact에서 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 전체 규정 준수의 중요한 부분입니다.
- SEC01-BP06 파이프라인에서 보안 제어 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 정책을 기반으로 합니다.

- SEC10-BP03 포렌식 역량 확보 - 포렌식 역량은 규정 준수 감사에 사용할 수도 있습니다.

관련 문서:

- AWS 규정 준수 센터
- AWS 규정 준수 리소스
- AWS 위험 및 규정 준수 백서
- AWS Shared Responsibility Model
- 규정 준수 프로그램 제공 AWS 범위 내 서비스

관련 비디오:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202)

관련 예제:

- AWS 기반 PCI DSS 및 AWS Foundational Security Best Practices

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.

Well-Architected Framework에서는 학습, 평가, 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 규모를 조정 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. 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 평가할 수 있는 미션 크리티컬 워크로드의 Well-Architected Review 안내 서비스를 이용할 수 있습니다. Enterprise Support 고객은 클라우드 운영 방식의 격차를 파악하는 데 사용할 수 있는 Operations Review 서비스를 이용할 수 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 우선순위를 결정할 수 있습니다.

AWS Trusted Advisor은 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, 신뢰성, 성능 및 비용 최적화 영역을 중점적으로 확인할 수 있습니다.

원하는 성과:

- Well-Architected 및 Trusted Advisor 결과물을 정기적으로 검토하고 이에 따라 조치를 취합니다.
- 서비스의 최신 패치 상태를 알고 있습니다.
- 알려진 위협의 위험과 영향을 이해하고 그에 따라 조치를 취합니다.
- 필요에 따라 완화 조치를 실행합니다.
- 조치와 상황에 대해 알립니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

- 위협 환경 평가: 주력할 영역을 결정할 때 해당 영향을 포함할 수 있도록 업무상의 위협 요소(예: 경쟁, 업무상의 위험/책임, 운영상의 위험, 정보 보안 위협)를 평가합니다.
  - AWS 최신 보안 공지
  - AWS Trusted Advisor
- 위협 모델 유지 관리: 잠재적 위협, 계획되고 배치된 완화 및 해당 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 모범 사례:

- SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정

관련 문서:

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

관련 비디오:

- AWS re:Inforce 2023 - A tool to help improve your threat modeling

OPS01-BP06 이점과 위험을 관리하면서 장단점 평가

여러 당사자의 이해관계가 상충하면 노력의 우선순위를 정하고 역량을 구축하며 비즈니스 전략에 맞는 결과를 제공하는 것이 어려울 수 있습니다. 예를 들어 IT 인프라 비용을 최적화하는 것보다 새로운 기능의 시장 출시를 앞당기는 것에 중점을 두라는 요청을 받을 수 있습니다. 이로 인해 두 당사자 간에 이해가 상충할 수 있습니다. 이러한 상황에서는 분쟁 해결을 위해 상위 기관에 결정을 맡겨야 합니다. 의사 결정 프로세스에서 정서적 애착에 좌우되지 않도록 하려면 데이터가 필요합니다.

전술적 차원에서도 같은 문제가 발생할 수 있습니다. 예를 들어, 관계형 데이터베이스 기술을 사용할지, 비관계형 데이터베이스 기술을 사용할지 선택하는 것은 애플리케이션 운영에 상당한 영향을 미칠 수 있습니다. 다양한 선택의 예측 가능한 결과를 이해하는 것이 중요합니다.

AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS와 해당 서비스 관련 정보를 제공할 수 있습니다. 지원(AWS 지식 센터, AWS 토론 포럼, 지원 센터) 및 AWS 설명서에 나와 있는 리소스를 사용하여 팀을 교육합니다. 더 궁금한 점이 있으면 지원로 문의하세요.

AWS는 또한 Amazon Builders' Library의 운영 모범 사례와 패턴을 공유합니다. AWS 블로그 및 공식 AWS 팟캐스트에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다.

원하는 성과: 명확하게 정의된 의사 결정 거버넌스 프레임워크를 구축하여 클라우드 전송 조직 내 모든 수준에서 중요한 결정을 촉진할 수 있습니다. 이 프레임워크에는 위험 관리 대장, 의사 결정 권한이 부여된 정의된 역할, 내릴 수 있는 각 의사 결정 수준에 대한 정의된 모델 등의 기능이 포함됩니다. 이 프레임워크는 갈등 해결 방법, 제시해야 하는 데이터, 옵션의 우선순위 지정 방법을 미리 정의하므로 일단 결정을 내리면 지체 없이 실행할 수 있습니다. 의사 결정 프레임워크에는 모든 결정의 이점과 위험을 검토하고 평가하여 장단점을 이해하기 위한 표준화된 접근 방식이 포함됩니다. 여기에는 규정 준수 요구 사항 충족과 같은 외부 요인이 포함될 수 있습니다.

일반적인 안티 패턴:

- 투자자는 결제 카드 산업 데이터 보안 표준(PCI-DSS)을 준수함을 입증할 것을 요청합니다. 요청을 충족하는 것과 현재 개발 작업을 계속하는 것 사이의 장단점을 고려하지 않습니다. 이렇게 하는 대신, 규정 준수를 입증하지 않고 개발 작업을 계속 진행합니다. 투자자는 플랫폼 보안과 투자에 대한 우려 사항으로 인해 회사의 지원을 중단합니다.
- 개발자 중 한 명이 인터넷에서 찾은 라이브러리를 포함하기로 결정했습니다. 출처를 알 수 없는 이 라이브러리의 도입에 따른 위험을 평가하지 않았으며 취약성 또는 악성 코드가 포함되어 있는지 알 수 없습니다.
- 마이그레이션에 대한 최초의 사업성 근거는 애플리케이션 워크로드의 60%를 현대화한다는 것이었습니다. 그러나 기술적인 문제로 인해 20%만 현대화하기로 결정하여 장기적으로 계획된 이익이 줄고, 인프라 팀이 레거시 시스템을 수동으로 지원해야 하므로 운영자의 수고가 증가했으며, 이러한 변경을 계획하지 않은 인프라 팀에서 새로운 기술 역량 개발에 대한 의존도가 높아졌습니다.

이 모범 사례 확립의 이점: 이사회 수준의 비즈니스 우선순위를 완벽하게 조정 및 지원하고, 성공을 위해 수반되는 위험을 이해하며, 정보에 입각한 의사 결정을 내리고, 위험이 성공 가능성을 저해하는 경우 적절한 조치를 취합니다. 의사 결정의 영향과 결과를 이해하면 선택의 우선순위를 정하고 리더가 더 빨리 합의에 도달하여 비즈니스 성과를 개선할 수 있습니다. 선택을 통해 얻을 수 있는 이점을 파악하

고 조직에 미칠 수 있는 위험을 인식하면 사례에 의존하지 않고 데이터에 기반한 결정을 내리는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

이점과 위험 관리는 주요 의사 결정의 요구 사항을 주도하는 관리 기관에 의해 정의되어야 합니다. 관련된 위험을 이해하면서 조직에 어떤 이점이 있는지를 기반으로 결정을 내리고 우선순위를 정하기를 원합니다. 정확한 정보는 조직이 결정을 내리는 데 매우 중요합니다. 이는 견고한 측정을 기반으로 하고 비용 이익 분석의 일반적인 업계 관행에 따라 정의되어야 합니다. 이러한 유형의 결정을 내리려면 중앙 집중식 권한과 분산형 권한 간의 균형을 유지해야 합니다. 항상 장단점이 있기 때문에 각 선택이 정의된 전략과 원하는 비즈니스 성과에 어떤 영향을 미치는지 이해하는 것이 중요합니다.

구현 단계

1. 종합적인 클라우드 거버넌스 프레임워크 내에서 이점 측정 사례를 공식화하세요.
   a. 의사 결정의 중앙 통제와 일부 의사 결정에 대한 분산된 권한 간의 균형을 유지합니다.
   b. 모든 의사 결정에 부과되는 부담스러운 의사 결정 프로세스로 인해 속도가 느려질 수 있다는 점을 이해하세요.
   c. 의사 결정 프로세스에 외부 요인(예: 규정 준수 요구 사항)을 통합하세요.
2. 이해 상충의 대상이 되는 의사 결정을 진행해야 하는 사람을 포함하여 다양한 수준의 의사 결정에 대해 합의된 의사 결정 프레임워크를 수립합니다.
   a. 되돌릴 수 없는 단방향 결정을 중앙 집중화합니다.
   b. 하위 조직 리더가 양방향 결정을 내릴 수 있도록 합니다.
3. 이점과 위험을 이해하고 관리합니다. 결정을 통해 제공되는 이점과 관련 위험을 적절하게 절충합니다.
   a. 이점 식별: 비즈니스 목표, 필요 사항 및 우선순위에 따른 이점을 파악합니다. 비즈니스 사례에 미치는 영향, 출시 시간, 보안, 신뢰성, 성능, 비용 등을 예로 들 수 있습니다.
   b. 위험 식별: 비즈니스 목표, 필요 사항 및 우선순위에 따른 위험을 파악합니다. 출시 시간, 보안, 신뢰성, 성능 및 비용 등을 예로 들 수 있습니다.
   c. 위험 대비 이점 평가 및 정보에 입각한 의사 결정: 실무,개발, 운영을 비롯한 주요 이해관계자의 목표, 필요 사항 및 우선순위를 기준으로 하여 이점과 위험의 영향을 확인합니다. 위험 발생 가능성 및 해당 위험이 주는 영향의 비용 대비 이점의 가치를 평가합니다. 예를 들어, 신뢰성보다 시장 진입 속도를 강조하면 경쟁 우위를 제공할 수 있습니다. 하지만 신뢰성 문제가 있는 경우 가동 시간이 단축될 수 있습니다.

4. 규정 준수 요구 사항 준수를 자동화하는 주요 결정을 프로그래밍 방식으로 적용합니다.
5. 가치 흐름 분석 및 LEAN과 같은 알려진 업계 프레임워크와 기능을 활용하여 현재 상태 성과와 비즈니스 지표의 기준을 정하고 이러한 지표의 개선을 위한 진행 상황을 정의합니다.

구현 계획의 작업 수준: 중간~높음

리소스

관련 모범 사례:

- OPS01-BP05 위협 환경 평가

관련 문서:

- Amazon의 Day 1 문화 요소 | 고품질의 빠른 의사 결정
- 클라우드 거버넌스
- Management & Governance Cloud Environment
- Governance in the Cloud and in the Digital Age: Parts One & Two

관련 비디오:

- 팟캐스트 | Jeff Bezos | On how to make decisions

관련 예제:

- Make informed decisions using data (The DevOps Sagas)
- Using development value stream mapping to identify constraints to DevOps outcomes

OPS 2. 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합하나요?

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, 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 책임과 소유권을 관리하는 메커니즘 확보
- OPS02-BP05 추가, 변경 및 예외를 요청하는 메커니즘
- OPS02-BP06 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 성과:

- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 관리합니다. 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. 계정이 AWS Organizations에서 관리되지 않는 경우 대체 계정 연락처를 사용하면 AWS가 필요에 따라 적절한 담당자와 연락할 수 있습니다. 계정의 대체 연락처가 개인이 아닌 그룹으로 설정되도록 구성합니다.
3. 태그를 사용하여 AWS 리소스 소유자를 식별합니다. 소유자와 해당 연락처 정보를 별도의 태그로 지정할 수 있습니다.
   a. AWS Config 규칙을 사용하여 리소스에 필수 소유권 태그가 포함되도록 할 수 있습니다.
   b. 조직의 태그 지정 전략을 개발하는 방법에 대한 자세한 지침은 AWS Tagging Best Practices 백서를 참조하세요.
4. 생성형 AI를 사용하여 기업 시스템의 정보를 기반으로 직원 생산성을 높이고, 질문에 답하며, 작업을 완료하는 대화형 어시스턴트인 Amazon Q Business를 사용합니다.
   a. Amazon Q Business를 회사의 데이터 소스에 연결합니다. Amazon Q Business는 Amazon Simple Storage Service(S3), Microsoft SharePoint, Salesforce, Atlassian Confluence를 포함하

여 40개가 넘는 지원되는 데이터 소스에 대한 사전 구축된 커넥터를 제공합니다. 자자세한 내용은 Amazon Q Business 커넥터를 참조하세요.

5. 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
- OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보

관련 문서:

- AWS Account Management - Updating contact information
- AWS Organizations - Updating alternative contacts in your organization
- AWS Tagging Best Practices 백서
- Build private and secure enterprise generative AI apps with Amazon Q Business and AWS IAM Identity Center
- Amazon Q Business, now generally available, helps boost workforce productivity with generative AI
- AWS 클라우드 Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

관련 워크숍:

- AWS 워크숍 - Tagging

관련 예제:

- AWS Config 규칙 - Amazon EC2 with required tags and valid values

관련 서비스:

- AWS Config 규칙 - required-tags
- 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.

원하는 성과: 운영 작업에 대한 일련의 프로세스와 절차가 효율적으로 정의 및 관리되고 있습니다. 프로세스와 절차는 중앙 위치에 저장되며 팀원이 사용할 수 있습니다. 프로세스 및 절차는 소유권을 명확하게 할당하여 자주 업데이트됩니다. 가능한 경우 스크립트, 템플릿 및 자동화 문서가 코드로 구현됩니다.

일반적인 안티 패턴:

- 프로세스를 문서화하지 않습니다. 단편화된 스크립트가 격리된 운영자 워크스테이션에 존재할 수 있습니다.
- 스크립트 사용 방법에 대한 지식은 소수의 개인이 보유하거나 팀 지식으로 비공식적으로 유지됩니다.
- 업데이트에는 기존 프로세스가 필요하지만 업데이트의 소유권이 불분명하고 원래 작성자가 더 이상 조직의 일원이 아닙니다.
- 프로세스와 스크립트를 검색할 수 없어 필요할 때(예: 인시던트 대응) 즉시 사용할 수 없습니다.

이 모범 사례 확립의 이점:

- 프로세스와 절차를 통해 워크로드 운영 작업을 강화할 수 있습니다.
- 새로운 팀원들은 더 빠르게 역량을 발휘할 수 있습니다.
- 인시던트 완화 시간이 단축됩니다.
- 여러 팀원(및 팀)이 동일한 프로세스와 절차를 일관되게 사용할 수 있습니다.
- 팀이 반복 가능한 프로세스를 통해 프로세스 규모를 조정할 수 있습니다.
- 표준화된 프로세스와 절차는 팀 간에 워크로드 책임을 이전하는 데 따른 영향을 완화하는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

- 프로세스 및 절차에서 정의를 담당하는 소유자를 식별했습니다.
  - 워크로드 지원을 위해 수행되는 운영 활동을 식별합니다. 검색 가능한 위치에 이러한 활동을 문서화합니다.
  - 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.
  - AWS Systems Manager와 같은 서비스, 문서, AWS Lambda를 통해 활동 아티팩트의 메타데이터에 대한 소유권을 파악합니다. 태그 또는 리소스 그룹을 사용하여 리소스 소유권을 캡처하고 소유권 및 연락처 정보를 지정합니다. AWS Organizations를 사용하여 태그 지정 정책을 생성하고 소유권 및 연락처 정보를 파악합니다.
- 시간이 지남에 따라 이러한 절차를 코드로 실행할 수 있도록 발전시켜 사람이 개입할 필요가 줄어들어야 합니다.
  - AWS Lambda 함수, CloudFormation 템플릿 또는 AWS Systems Manager Automation 문서를 예로 들어 보겠습니다.
  - 적절한 리포지토리에서 버전 관리를 수행합니다.
  - 소유자와 문서를 쉽게 식별할 수 있도록 적절한 리소스 태그 지정을 포함합니다.

고객 사례

AnyCompany Retail은 소유권을 애플리케이션 또는 애플리케이션 그룹(공통 아키텍처 관행과 기술을 공유)에 대한 프로세스를 소유하는 팀 또는 개인으로 정의합니다. 처음에는 프로세스 및 절차가 문서 관리 시스템에 단계별 가이드로 문서화되며, 애플리케이션을 호스팅하는 AWS 계정과 계정 내 특정 리소스 그룹에서 태그를 사용하여 검색할 수 있습니다. 이들은 AWS Organizations를 사용하여 AWS 계정을 관리합니다. 시간이 지남에 따라 이러한 프로세스는 코드로 변환되고 리소스는 코드형 인프라(예: CloudFormation 또는 AWS 클라우드 개발 키트 (AWS CDK) 템플릿)를 사용하여 정의됩니다. 운영 프로세스는 AWS Systems Manager 또는 AWS Lambda 함수의 자동화 문서가 됩니다. 그러면 AWS CloudWatch 경보 또는 AWS EventBridge 이벤트와 같은 이벤트에 대응하여 예약된 작업으로 시작되거나 IT 서비스 관리(ITSM) 플랫폼 내의 요청에 의해 시작될 수 있습니다. 모든 프로세스에는 소유권을 식별하는 태그가 있습니다. 자동화 및 프로세스에 대한 문서는 프로세스의 코드 저장소에서 생성한 Wiki 페이지 내에서 유지 관리됩니다.

구현 단계

1. 기존 프로세스 및 절차를 문서화합니다.
   a. 이를 검토하고 최신 상태로 유지합니다.
   b. 각 프로세스 또는 절차의 소유자를 식별합니다.
   c. 이에 대한 버전을 관리합니다.
   d. 가능하면 아키텍처 설계를 공유하는 워크로드 및 환경 전반에서 프로세스와 절차를 공유합니다.
2. 피드백 및 개선을 위한 메커니즘을 수립합니다.
   a. 프로세스를 검토해야 하는 빈도에 대한 정책을 정의합니다.
   b. 검토자와 승인자를 위한 프로세스를 정의합니다.
   c. 피드백을 제공하고 추적할 수 있도록 문제 또는 티켓팅 대기열을 구현합니다.
   d. 가능하면 프로세스 및 절차에는 변경 승인 위원회(CAB)의 사전 승인 및 위험 분류가 있어야 합니다.
3. 프로세스와 절차를 실행해야 하는 담당자가 프로세스와 절차에 액세스하고 검색할 수 있는지 확인합니다.
   a. 태그를 사용하여 워크로드의 프로세스 및 절차에 액세스할 수 있는 위치를 지정합니다.
   b. 의미 있는 오류 및 이벤트 메시지를 사용하여 문제를 해결하기 위한 적절한 프로세스나 절차를 지정합니다.
   c. Wiki 및 문서 관리를 사용하여 프로세스 및 절차를 조직 전체에서 일관되게 검색할 수 있도록 합니다.
4. 생성형 AI를 사용하여 기업 시스템의 정보를 기반으로 직원 생산성을 높이고, 질문에 답하며, 작업을 완료하는 대화형 어시스턴트인 Amazon Q Business를 사용합니다.
   a. Amazon Q Business를 회사의 데이터 소스에 연결합니다. Amazon Q Business는 Amazon S3, Microsoft SharePoint, Salesforce, Atlassian Confluence를 포함하여 40개가 넘는 지원되는 데이터 소스에 대한 사전 구축된 커넥터를 제공합니다. 자세한 내용은 Amazon Q 커넥터를 참조하세요.
5. 필요한 경우 자동화합니다.
   a. 서비스 및 기술 팀에서 API를 제공할 경우 자동화를 개발해야 합니다.
   b. 프로세스에 대해 적절하게 교육합니다. 사용자 스토리와 요구 사항을 개발하여 이러한 프로세스를 자동화합니다.
   c. 프로세스와 절차의 사용을 성공적으로 측정하고, 지속적인 개선을 지원하기 위한 이슈 또는 티켓을 생성합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS02-BP01 리소스 소유자 식별
- OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보
- OPS11-BP04 지식 관리 수행

관련 문서:

- AWS 백서 - AWS의 DevOps 소개
- AWS 백서 - Best Practices for Tagging AWS Resources
- AWS 백서 - Organizing Your AWS Environment Using Multiple Accounts
- AWS 클라우드 Operations and Migrations Blog - Using Amazon Q Business to streamline your operations
- AWS 클라우드 Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS 클라우드 Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

관련 워크숍:

- AWS Well-Architected Operational Excellence 워크숍
- AWS 워크숍 - Tagging

관련 비디오:

- How to automate IT Operations on AWS
- AWS re:Invent 2020 - Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 - Automating patch management and compliance using AWS (NIS306)

- 지원s You - Diving Deep into AWS Systems Manager

관련 서비스:

- AWS Systems Manager - 자동화
- AWS Service Management Connector

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.

원하는 성과:

조직은 정의된 워크로드에서 특정 활동을 수행하고 워크로드에서 생성된 이벤트에 대응해야 할 책임을 명확하게 정의합니다. 조직은 프로세스 및 이행의 소유권을 문서화하고 이 정보를 검색할 수 있게 합니다. 조직 변경이 발생하면 책임을 검토 및 업데이트하고 팀은 결함 및 비효율성 식별 활동의 성과를 추적하고 측정합니다. 피드백 메커니즘을 구현하여 결함과 개선 사항을 추적하고 지속적인 개선을 지원합니다.

일반적인 안티 패턴:

- 책임을 문서화하지 않습니다.
- 단편화된 스크립트가 격리된 운영자 워크스테이션에 존재합니다. 이를 사용하는 방법을 알고 있거나 비공식적으로 팀 지식이라고 부르는 사람은 소수에 불과합니다.
- 기존 프로세스를 업데이트할 예정이지만 프로세스 담당자가 누구인지 알 수 없으며 원래 작성자도 더 이상 조직의 일원이 아닙니다.
- 프로세스와 스크립트를 검색할 수 없어 필요할 때(예: 인시던트 대응) 즉시 사용할 수 없습니다.

이 모범 사례 확립의 이점:

- 누가 활동 수행을 담당하는지, 조치가 필요한 경우 누구에게 알릴 것인지, 누가 작업을 수행하고 결과를 검증하고 활동 소유자에게 피드백을 제공하는지 이해할 수 있습니다.
- 프로세스와 절차를 통해 워크로드 운영 작업을 강화할 수 있습니다.
- 새로운 팀원들은 더 빠르게 역량을 발휘할 수 있습니다.

- 인시던트를 완화하는 데 걸리는 시간을 줄일 수 있습니다.
- 팀마다 동일한 프로세스와 절차를 사용하여 일관된 방식으로 작업을 수행합니다.
- 팀이 반복 가능한 프로세스를 통해 프로세스 규모를 조정할 수 있습니다.
- 표준화된 프로세스와 절차는 팀 간에 워크로드 책임을 이전하는 데 따르는 영향을 완화하는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

책임을 정의하려면 먼저 책임 매트릭스, 프로세스 및 절차, 역할 및 책임, 도구 및 자동화와 같은 기존 문서부터 시작해야 합니다. 문서화된 프로세스의 책임을 검토하고 토론을 주최하세요. 팀과 함께 검토하여 문서의 책임과 프로세스 간의 불일치를 파악합니다. 팀 간의 기대치 차이를 파악하기 위해 해당 팀의 내부 고객과 함께 제공되는 서비스에 대해 논의하세요.

불일치를 분석하고 해결합니다. 개선 기회를 파악하고, 자주 요청되고 리소스를 많이 사용하는 활동(일반적으로 개선이 필요한 활동)을 찾아보세요. 개선을 간소화하고 표준화하기 위한 모범 사례, 패턴 및 권장 가이드를 살펴보세요. 개선 기회를 기록하고 개선이 완료될 때까지 추적하세요.

시간이 지남에 따라 이러한 절차를 코드로 실행할 수 있도록 발전시켜 사람이 개입할 필요가 줄어들어야 합니다. 예를 들어 프로시저는 AWS Lambda 함수, AWS CloudFormation 템플릿 또는 AWS Systems Manager Automation 문서로 시작할 수 있습니다. 이러한 절차가 적절한 리포지토리에서 버전 관리되는지 확인하고 팀에서 소유자와 문서를 쉽게 식별할 수 있도록 적절한 리소스 태그 지정을 포함하세요. 활동 수행에 대한 책임을 문서화한 다음 성공적인 시작 및 운영과 원하는 성과의 성능을 위한 자동화를 모니터링합니다.

고객 사례

AnyCompany Retail은 소유권을 공통 아키텍처 관행과 기술을 공유하는 애플리케이션 또는 애플리케이션 그룹에 대한 프로세스를 소유하는 팀 또는 개인으로 정의합니다. 처음에 회사는 문서 관리 시스템의 단계별 지침으로 프로세스와 절차를 문서화합니다. AWS 계정을 관리하는 AWS Organizations를 통해, 애플리케이션을 호스팅하는 AWS 계정 태그와 계정 내 특정 리소스 그룹의 태그를 사용하여 프로시저를 검색할 수 있도록 합니다. AnyCompany Retail은 시간이 지남에 따라 이러한 프로세스를 코드로 변환하고 CloudFormation 또는 AWS 클라우드 개발 키트 (AWS CDK) 템플릿과 같은 서비스를 통해 코드형 인프라를 사용하여 리소스를 정의합니다. 운영 프로세스는 AWS Systems Manager 또는 AWS Lambda 함수의 자동화 문서가 됩니다. 그러면 Amazon CloudWatch 경보 또는 Amazon EventBridge 이벤트와 같은 이벤트에 대응하여 예약된 작업으로 시작되거나 IT 서비스 관리(ITSM) 플랫폼 내의 요청에 의해 시작될 수 있습니다. 모든 프로세스에는 소유자를 식별하는 태그가 있습니다.

팀은 자동화 및 프로세스에 대한 문서를 프로세스의 코드 저장소에서 생성한 Wiki 페이지 내에서 관리합니다.

구현 단계

1. 기존 프로세스 및 절차를 문서화합니다.
   a. 최신 상태인지 검토하고 확인합니다.
   b. 각 프로세스 또는 프로시저에 소유자가 있는지 확인합니다.
   c. 프로시저의 버전을 관리하세요.
   d. 가능하면 아키텍처 설계를 공유하는 워크로드 및 환경 전반에서 프로세스와 절차를 공유합니다.
2. 피드백 및 개선을 위한 메커니즘을 수립합니다.
   a. 프로세스를 검토해야 하는 빈도에 대한 정책을 정의합니다.
   b. 검토자와 승인자를 위한 프로세스를 정의합니다.
   c. 문제 또는 티켓팅 대기열을 구현하여 피드백을 제공하고 추적합니다.
   d. 가능하면 프로세스 및 절차에는 변경 승인 위원회(CAB)의 사전 승인 및 위험 분류가 있어야 합니다.
3. 프로세스 및 프로시저를 실행해야 하는 사용자가 이를 액세스하고 검색할 수 있도록 합니다.
   a. 태그를 사용하여 워크로드의 프로세스 및 절차에 액세스할 수 있는 위치를 지정합니다.
   b. 의미 있는 오류 및 이벤트 메시지를 사용하여 문제를 해결하기 위한 적절한 프로세스나 절차를 지정합니다.
   c. Wiki 또는 문서 관리를 사용하여 조직 전체에서 프로세스와 절차를 일관되게 검색할 수 있습니다.
4. 필요할 때 자동화하세요.
   a. 서비스와 기술이 API를 제공하는 경우 자동화를 개발합니다.
   b. 프로세스를 제대로 이해하고 있는지 확인하고 해당 프로세스를 자동화하기 위한 사용자 스토리와 요구 사항을 개발합니다.
   c. 반복적 개선을 지원하는 문제 추적을 통해 프로세스 및 절차의 성공적인 사용을 측정합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS02-BP01 리소스 소유자 식별
- OPS02-BP02 프로세스 및 절차의 소유자 식별

- OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보
- OPS02-BP05 책임과 소유권을 식별하는 메커니즘
- OPS11-BP04 지식 관리 수행

관련 문서:

- AWS 백서 | AWS의 DevOps 소개
- AWS 백서 | Best Practices for Tagging AWS Resources
- AWS 백서 | Organizing Your AWS Environment Using Multiple Accounts
- AWS 클라우드 Operations & Migrations Blog | Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS 워크숍 - Tagging
- AWS Service Management Connector

관련 비디오:

- AWS Knowledge Center Live | Tagging AWS Resources
- AWS re:Invent 2020 | Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 | Automating patch management and compliance using AWS (NIS306)
- 지원s You | Diving Deep into AWS Systems Manager

OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보

역할의 책임과 비즈니스 성과에 기여하는 방법을 파악합니다. 이를 통해 작업의 우선순위와 역할이 중요한 이유를 알 수 있습니다. 그 결과 팀원들은 요구 사항을 인식하고 적절하게 대응할 수 있습니다. 팀원이 자신의 역할을 알게 되면 소유권을 확립하고 개선 기회를 식별하며 영향을 미치거나 적절한 변경을 수행하는 방법을 이해할 수 있습니다.

경우에 따라 책임의 소유자가 명확하지 않을 수 있습니다. 이러한 상황에서는 격차를 해소하는 메커니즘을 설계해야 합니다. 소유권을 할당하거나 필요 사항을 해결할 계획을 세울 권한이 있는 사람에게 이를 알릴 수 있도록 잘 정의된 에스컬레이션 경로를 만드세요.

원하는 성과: 조직 내 여러 팀이 리소스, 수행할 작업, 프로세스 및 절차와 관련된 방식을 포함하여 명확하게 정의된 책임을 갖고 있습니다. 이러한 책임은 팀의 책임과 목표는 물론 다른 팀의 책임과도 일치합니다. 일관되고 검색 가능한 방식으로 에스컬레이션 경로를 문서화하고 이러한 결정을 책임 매트릭스, 팀 정의 또는 Wiki 페이지와 같은 문서 아티팩트에 입력합니다.

일반적인 안티 패턴:

- 팀의 책임은 모호하거나 잘못 정의되어 있습니다.
- 팀은 역할과 책임을 연계하지 않습니다.
- 팀은 목표와 목적, 책임을 조정하지 않으므로 성공을 측정하기가 어렵습니다.
- 팀원의 책임이 팀 및 상위 조직에 적합하지 않습니다.
- 팀은 책임을 최신 상태로 유지하지 않으므로 팀에서 수행하는 작업과 일치하지 않습니다.
- 책임 결정을 위한 에스컬레이션 경로가 정의되어 있지 않거나 명확하지 않습니다.
- 에스컬레이션 경로에는 시기적절한 응답을 보장하는 단일 스레드 소유자가 없습니다.
- 역할, 책임 및 에스컬레이션 경로는 검색할 수 없으며 필요할 때(예: 인시던트 대응) 즉시 사용할 수 없습니다.

이 모범 사례 확립의 이점:

- 책임이나 소유권을 가진 사람이 누구인지 알게 되면 적절한 팀이나 팀원에게 연락하여 요청을 하거나 작업을 전환할 수 있습니다.
- 조치를 취하지 않거나 해결되지 않은 요구 사항이 발생할 위험을 줄이기 위해 책임 또는 소유권을 할당할 권한이 있는 사람을 지정하게 됩니다.
- 책임 범위를 명확하게 정의하면 팀원이 자율성과 소유권을 얻게 됩니다.
- 책임 범위 정의를 통해 사용자가 내리는 결정, 취하는 조치, 적절한 소유자에 대한 활동 전달을 알 수 있습니다.
- 팀의 책임 범위를 벗어나는 것이 무엇인지 명확하게 이해할 수 있기 때문에 포기한 책임을 쉽게 식별할 수 있으며, 이를 통해 명확하게 에스컬레이션할 수 있습니다.
- 팀은 혼란과 긴장을 피하고 워크로드와 리소스를 더 적절하게 관리할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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. 조직이 AWS에서 마이그레이션 및 현대화 기회를 활용하고자 함에 따라 역할과 책임도 달라질 수 있습니다. 팀과 팀원이 각자의 책임을 인식하도록 하고 이러한 변경 기간에 작업을 수행할 수 있도록 적절하게 교육하세요.

에스컬레이션을 받아야 하는 역할 또는 팀을 결정하여 책임과 소유권을 식별합니다. 이 팀은 다양한 이해관계자와 협력하여 결정을 내릴 수 있습니다. 그러나 이들은 의사 결정 프로세스의 관리를 담당해야 합니다.

조직 구성원에게 소유권과 책임을 발견하고 식별할 수 있는 액세스 가능한 메커니즘을 제공합니다. 이러한 메커니즘은 특정 요구 사항에 대해 누구에게 연락해야 하는지 알려줍니다.

고객 사례

AnyCompany Retail은 최근 리프트 앤 시프트 방식으로 온프레미스 환경에서 AWS 랜딩 존으로 워크로드를 마이그레이션하는 작업을 완료했습니다. 이들은 운영 검토를 수행하여 일반적인 운영 작업을 어떻게 수행하는지 되돌아보고 기존의 책임 매트릭스가 새로운 환경에서의 운영을 반영하는지 확인했습니다. 온프레미스에서 AWS로 마이그레이션하면서 하드웨어 및 물리적 인프라와 관련된 인프라 팀의 책임이 줄어들었습니다. 또한 이에 따라 워크로드의 운영 모델을 발전시킬 수 있는 새로운 기회가 드러났습니다.

이들은 대부분의 책임을 식별, 해결 및 문서화하는 동시에 운영 관행이 발전함에 따라 놓쳤거나 변경해야 할 수 있는 책임에 대한 에스컬레이션 경로도 정의했습니다. 워크로드 전반을 표준화하고 효율성을 개선할 수 있는 새로운 기회를 모색하려면 AWS Systems Manager와 같은 운영 도구와 AWS Security Hub 및 Amazon GuardDuty와 같은 보안 도구에 대한 액세스를 제공하세요. AnyCompany Retail은 가장 먼저 해결하고자 하는 개선 사항을 바탕으로 책임과 전략을 검토합니다. 회사는 새로운 업무 방식과 기술 패턴을 도입함에 따라 이에 맞게 책임 매트릭스를 업데이트합니다.

구현 단계

1. 기존 문서부터 시작하세요. 일반적인 소스 문서에는 다음이 포함될 수 있습니다.
   a. 책임 또는 Responsible, Accountable, Consulted, and Informed(RACI) 매트릭스
   b. 팀 정의 또는 Wiki 페이지
   c. 서비스 정의 및 오퍼링
   d. 역할 또는 직무 설명
2. 문서화된 책임을 검토하고 토론을 주최하세요.
   a. 팀과 함께 검토하여 문서화된 책임과 팀이 일반적으로 수행하는 책임 간의 불일치 사항을 파악하세요.
   b. 내부 고객이 제공하는 잠재적 서비스에 대해 논의하여 팀 간의 기대치 차이를 파악하세요.
3. 불일치를 분석하고 해결합니다.
4. 몇 가지 개선 기회를 파악합니다.
   a. 자주 요구되고 리소스를 많이 사용하는 요청(일반적으로 강력한 개선 대상)을 식별합니다.

   b. 모범 사례를 찾아보고, 패턴을 파악하고, 권장 가이드를 따라 개선을 간소화하고 표준화합니다.
   c. 개선 기회를 기록하고 완료될 때까지 추적합니다.
5. 팀에 아직 책임 할당을 관리하고 추적할 책임이 없다면 팀에서 이 책임을 맡을 사람을 정합니다.
6. 팀의 책임 명시 요청 프로세스를 정의합니다.
   a. 프로세스를 검토하고 명확하고 사용하기 쉬운지 확인합니다.
   b. 누군가가 에스컬레이션을 담당하고 추적하여 결론에 도달하도록 하세요.
   c. 운영 지표를 수립하여 효과를 측정합니다.
   d. 피드백 메커니즘을 만들어 팀이 개선 기회를 강조할 수 있는지 확인하세요.
   e. 주기적 검토를 위한 메커니즘을 구현합니다.
7. 검색 가능하고 액세스 가능한 위치에 문서화하세요.
   a. 일반적으로는 Wiki 또는 문서 포털을 사용합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS01-BP06 장단점 평가
- OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여
- OPS03-BP03 에스컬레이션 장려
- OPS03-BP07 팀에 적절한 리소스 제공
- OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정
- OPS09-BP03 운영 지표 검토 및 개선 우선순위 지정
- OPS11-BP01 지속적인 개선을 위한 프로세스 마련

관련 문서:

- AWS 백서 - AWS의 DevOps 소개
- AWS 백서 - AWS 클라우드 Adoption Framework: Operations Perspective
- AWS Well-Architected Framework 운영 우수성 - 워크로드 수준 운영 모델 토폴로지
- AWS Prescriptive Guidance - Building your Cloud Operating Model

- AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model
- AWS 클라우드 Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams
- AWS 클라우드 Operations & Migrations Blog - Why a Cloud Operating Model?
- AWS DevOps Blog - How organizations are modernizing for cloud operations

관련 비디오:

- AWS Summit Online - Cloud Operating Models for Accelerated Transformation
- AWS re:Invent 2023 - Future-proofing cloud security: A new operating model

OPS02-BP05 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 성과:

- 할당된 소유권에 따라 프로세스, 절차 및 리소스 변경을 요청할 수 있습니다.
- 변경은 이점과 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 시스템에서 문서화합니다.
   a. OPS02-BP01 리소스 소유자 식별, OPS02-BP02 프로세스 및 절차의 소유자 식별 또는 OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별 작업을 구현하지 않았다면 해당 작업부터 시작합니다.
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 시스템에서 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 구축하기 전에 리소스에 식별된 소유자가 있어야 합니다.
- OPS02-BP02 프로세스 및 절차의 소유자 식별 - 변경 관리 프로세스를 구축하기 전에 프로세스에 식별된 소유자가 있어야 합니다.
- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별 - 변경 관리 프로세스를 구축하기 전에 운영 활동에 식별된 소유자가 있어야 합니다.

관련 문서:

- AWS Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices
- Change Management in the Cloud Whitepaper

관련 서비스:

- AWS Systems Manager Change Manager

OPS02-BP06 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 성과:

- 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 사례가 확립되지 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 계약을 설정하기 전에 프로세스 소유권을 식별해야 합니다.
- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별 - 팀 간 계약을 설정하기 전에 운영 활동 소유권을 식별해야 합니다.

관련 문서:

- AWS Executive Insights - 2-피자 팀을 통해 더 많이 빠르게 혁신
- AWS에서 DevOps 소개 - 피자 두 판 팀

OPS 3. 조직 문화는 비즈니스 성과를 어떻게 지원하나요?

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

모범 사례

- OPS03-BP01 경영진의 후원 제공
- 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 역량 유지와 강화 장려
- OPS03-BP07 팀에 적절한 리소스 제공

OPS03-BP01 경영진의 후원 제공

최고 수준에서 고위 경영진은 총괄 후원자 역할을 하여 조직의 성공에 대한 평가를 포함하여 조직의 성과에 대한 기대치와 방향을 명확하게 설정합니다. 후원자는 조직의 모범 사례 채택과 발전을 지지하고 추진합니다.

원하는 성과: 클라우드 운영을 채택, 전환 및 최적화하기 위해 노력하는 조직은 원하는 성과에 대한 명확한 리더십과 책임을 확립합니다. 조직은 조직이 새로운 성과를 달성하는 데 필요한 각 역량을 이해하고 발전을 위해 담당 팀에 소유권을 지정합니다. 리더십은 이러한 방향을 적극적으로 설정하고, 소유권을 지정하며, 책임을 지고, 업무를 정의합니다. 따라서 조직 전반에서 개인은 함께 참여하고 동기를 부여 받으며 원하는 목표를 향해 적극적으로 노력할 수 있습니다.

일반적인 안티 패턴:

- 워크로드 소유자는 명확한 후원자 및 클라우드 운영 계획 없이 AWS로 워크로드를 마이그레이션해야 합니다. 그 결과 팀은 운영 역량을 개선하고 성숙시키겠다는 목적 의식을 갖고 협력하지 않습니다. 운영 모범 사례 표준의 부재로 인해 팀이 어려움(예: 운영자의 수고, 당직, 기술 부채)을 겪고 있어 혁신에 제약이 따릅니다.
- 리더십 후원자 및 전략을 제공하지 않고 새로운 기술을 채택해야 한다는 새로운 조직 차원의 목표가 설정되었습니다. 팀은 목표를 다르게 해석하기 때문에 어디에 노력을 집중해야 하는지, 왜 중요한지, 영향을 어떻게 측정하는지에 대해 혼란이 발생합니다. 결과적으로 조직은 이 기술을 채택하는 데 추진력을 잃게 됩니다.

이 모범 사례 확립의 이점: 경영진 후원을 통해 비전, 방향 및 목표를 명확하게 전달하고 공유하면 팀원은 자신에게 기대되는 바를 알 수 있습니다. 리더가 적극적으로 참여할 때 개인과 팀은 정의된 목표를 달성하기 위해 같은 방향으로 집중적으로 노력을 기울이기 시작합니다. 결과적으로 조직은 성공을 위한 역량을 극대화합니다. 성공을 평가할 때 성공을 가로막는 장애물을 더 잘 식별하여 경영진 후원자의 개입을 통해 해결할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

- 클라우드 여정의 모든 단계(마이그레이션, 채택, 최적화)에서 성공을 거두려면 최고 수준의 리더십이 지정된 경영진 후원자와 함께 적극적으로 참여해야 합니다. 경영진 후원자는 팀의 사고 방식, 기술력, 작업 방식을 정의된 전략에 맞게 조정합니다.
  - 이유 설명: 명확하게 해명하고 비전과 전략의 근거를 설명합니다.
  - 기대치 설정: 진행 상황과 성공을 측정하는 방법을 포함하여 조직의 목표를 정의하고 게시합니다.
  - 목표 달성 추적: 목표의 점진적 성취도를 정기적으로 측정합니다(단순한 작업 완료가 아님). 결과가 위험할 경우 적절한 조치를 취할 수 있도록 결과를 공유합니다.
  - 목표 달성에 필요한 리소스 제공: 사람과 팀을 한데 모아 협업하고 정의된 성과를 지원하는 올바른 솔루션을 구축합니다. 이는 조직적 마찰을 줄이거나 없애줍니다.
  - 팀 지지: 팀과 지속적으로 협력하여 팀의 성과와 팀에 영향을 미치는 외부 요인을 파악합니다. 팀의 업무 진행을 방해하는 장애물을 파악합니다. 팀을 대신해 장애물을 해결하고 불필요한 부담을 제거하는 데 도움을 줍니다. 팀이 외부 요인의 영향을 받는 경우 목표를 재평가하고 적절하게 타겟을 조정합니다.
  - 모범 사례 채택 유도: 정량화할 수 있는 이점을 제공하고 만든 사람과 도입한 사람을 명시하는 모범 사례를 높이 평가합니다. 추가적인 채택을 장려하여 달성된 이점을 확대합니다.
  - 팀의 발전 장려: 지속적인 개선의 문화를 조성하고, 성공과 실패를 통해 능동적으로 학습합니다. 개인 및 조직의 성장과 개발을 장려합니다. 데이터와 사례를 통해 비전과 전략을 발전시킵니다.

고객 사례

AnyCompany Retail은 생성형 AI를 사용한 고객 경험의 빠른 혁신, 생산성 향상, 성장 가속화를 통해 비즈니스 혁신을 진행 중입니다.

구현 단계

1. 단일 스레드 리더십을 구축하고 혁신을 주도하고 추진할 주 경영진 후원자를 지정합니다.

2. 혁신의 명확한 비즈니스 성과를 정의하고 소유권과 책임을 지정합니다. 주요 경영진에 중요한 결정을 이끌고 결론지을 수 있는 권한을 부여합니다.
3. 혁신 전략이 매우 명확하고 경영진 후원자가 조직의 모든 수준에 이를 폭넓게 전달하도록 확인합니다.
   a. IT 및 클라우드 이니셔티브에 대해 명확하게 정의된 비즈니스 목표를 설정합니다.
   b. IT 및 클라우드 혁신을 추진하기 위한 주요 비즈니스 지표를 문서화합니다.
   c. 전략의 일부를 담당하는 모든 팀과 개인에게 비전을 일관되게 전달합니다.
4. 특정 리더, 관리자 및 개별 기여자에게 전달해야 하는 메시지를 지정하는 커뮤니케이션 계획 매트릭스를 개발합니다. 이 메시지를 전달해야 하는 담당자나 팀을 지정합니다.
   a. 커뮤니케이션 계획을 일관되고 신뢰할 수 있는 방식으로 이행합니다.
   b. 정기적으로 대면 이벤트를 통해 기대치를 설정하고 관리합니다.
   c. 커뮤니케이션의 효과에 대한 피드백을 수용하고 그에 따라 커뮤니케이션을 조정하고 계획을 세웁니다.
   d. 커뮤니케이션 이벤트를 예약하여 팀의 문제를 사전에 파악하고 필요한 경우 과정을 수정할 수 있는 일관된 피드백 루프를 구축합니다.
5. 리더십 관점에서 각 이니셔티브에 적극적으로 참여하여 영향을 받는 모든 팀이 달성해야 할 성과를 이해하고 있는지 확인합니다.
6. 모든 현황 회의에서 경영진 후원자는 방해 요소를 찾고, 확립된 지표, 사례 또는 팀의 피드백을 검토하고, 목표를 향한 진행 상황을 측정해야 합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션
- OPS11-BP01 지속적인 개선을 위한 프로세스 마련
- OPS11-BP07 운영 지표 검토 수행

관련 문서:

- Untangling Your Organisational Hairball: Highly Aligned
- The Living Transformation: Pragmatically approaching changes

- Becoming a Future-Ready Enterprise
- 7 Pitfalls to Avoid When Building a CCOE
- Navigating the Cloud: Key Performance Indicators for Success

관련 비디오:

- AWS re:Invent 2023: A leader's guide to generative AI: Using history to shape the future (SEG204)

관련 예제:

- Prosci: Primary Sponsor's Role & Importance

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

리더십이 책임 의식을 강화하는 문화를 구축하면 모든 직원이 자신이 정의한 역할 및 책임 범위를 넘어 회사 전체를 위해 행동할 수 있는 권한이 있다고 느끼게 됩니다. 직원은 위험이 발생할 때 이를 사전에 파악하고 적절한 조치가 이루어지도록 할 수 있습니다. 이러한 문화를 통해 직원은 상황을 인식하고 가치 있는 결정을 내릴 수 있습니다.

예를 들어, Amazon은 리더십 원칙을 지침으로 삼아 직원들이 상황에 맞게 앞으로 나아가고, 문제를 해결하며, 갈등을 해소하고, 조치를 취할 수 있도록 바람직한 행동을 유도합니다.

원하는 성과: 리더십은 감사 가능한 권한 및 안전 메커니즘으로 의사 결정이 정의되는 한, 조직의 하위 수준에서도 개인과 팀이 중요한 결정을 내릴 수 있도록 하는 새로운 문화에 영향을 줍니다. 실패하더라도 괜찮습니다. 팀은 계속해서 비슷한 상황에 대처하기 위해 의사 결정과 대응을 개선하는 방법을 반복해서 배우게 됩니다. 누군가의 행동이 다른 팀에 도움이 될 수 있는 개선으로 이어지면, 이들은 이러한 행동으로 얻은 지식을 사전에 공유합니다. 리더십은 운영 개선을 측정하고 이러한 패턴을 채택한 개인과 조직에 인센티브를 제공합니다.

일반적인 안티 패턴:

- 조직 내에는 위험이 식별되었을 때 어떤 조치를 취해야 하는지에 대한 명확한 지침이나 메커니즘이 없습니다. 예를 들어 피싱 공격을 감지한 직원은 보안팀에 보고하지 않아 조직의 상당 부분이 공격을 받게 됩니다. 이로 인해 데이터 침해가 발생합니다.
- 고객은 주로 배포 실패로 인한 서비스 가용성 장애에 대해 불평합니다. SRE 팀이 배포 도구를 담당하며 배포에 대한 자동 롤백은 장기 로드맵에 포함되어 있습니다. 최근 애플리케이션 출시에서 엔지니어 중 한 명이 애플리케이션을 이전 버전으로 자동으로 롤백하는 솔루션을 고안했습니다. 이들의

솔루션이 SRE 팀의 패턴이 될 수 있지만, 다른 팀은 이러한 개선 사항을 추적할 프로세스가 없기 때문에 이를 채택하지 않게 됩니다. 조직은 고객에게 영향을 미치고 부정적인 분위기를 야기하는 배포 실패로 인해 계속 어려움을 겪습니다.

- 규정을 준수하기 위해 Infosec 팀은 Amazon EC2 Linux 인스턴스에 연결하는 운영자를 대신하여 공유 SSH 키를 정기적으로 교체하기 위해 오랫동안 확립된 프로세스를 감독합니다. Infosec 팀이 키 교체를 완료하는 데 수일이 걸리며 해당 인스턴스에 연결할 수 없게 됩니다. Infosec 내부 또는 외부의 어느 누구도 동일한 결과를 얻기 위해 AWS에서 다른 옵션을 사용할 것을 제안하지 않습니다.

이 모범 사례 확립의 이점: 의사 결정 권한을 분산하고 팀이 주요 결정을 내릴 수 있도록 권한을 강화함으로써 문제를 더 빠르게 해결하고 성공률을 높일 수 있습니다. 또한 팀은 주인 의식을 깨닫기 시작하며, 실패도 감수할 수 있게 됩니다. 실험은 기업 문화의 핵심이 됩니다. 관리자와 이사는 업무의 모든 측면에서 세세하게 관리되는 것으로 느끼지 않게 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

1. 실패가 용납되는 문화를 조성합니다.
2. 조직 내 다양한 기능 영역에 대해 명확한 소유권과 책임을 정의합니다.
3. 모든 사람에게 소유권과 책임을 알리며 개인이 개별 의사 결정을 내리는 데 도움을 줄 수 있는 사람이 누구인지 알 수 있게 합니다.
4. 단방향 및 양방향 결정을 정의하여 개인이 언제 상위 리더십에 에스컬레이션해야 하는지 알 수 있게 합니다.
5. 결과가 위험한 경우 모든 직원이 다양한 수준에서 조치를 취할 권한이 있다는 사실에 대한 조직의 인식을 제고합니다. 팀원에게 효과적으로 대응하는 데 필요한 기술을 연습할 수 있는 거버넌스, 권한 수준, 도구 및 기회에 대한 문서를 제공합니다.
6. 팀원들에게 다양한 의사 결정에 대응하는 데 필요한 기술을 연습할 기회를 줍니다. 의사 결정 수준을 정의한 후에는 게임 데이를 실시하여 모든 개별 기여자가 프로세스를 이해하고 시연할 수 있는지 확인합니다.
   a. 프로세스와 절차를 테스트하고 교육할 수 있는 안전한 대체 환경을 제공합니다.
   b. 결과에 사전 정의된 수준의 위험이 있을 때 팀원에게 조치를 취할 권한이 있음을 인정하고 인식을 제고합니다.
   c. 특히 팀원이 지원하는 워크로드 및 구성 요소에 대한 권한과 액세스를 지정하여 조치를 취할 수 있는 팀원의 권한을 정의합니다.
7. 팀이 학습한 내용(운영 성공 및 실패)을 공유할 수 있는 기능을 제공합니다.

8. 팀이 현재 상황에 도전할 수 있도록 지원하고 개선 사항뿐만 아니라 조직에 미치는 영향을 추적하고 측정할 수 있는 메커니즘을 제공합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS01-BP06 이점과 위험을 관리하면서 장단점 평가
- OPS02-BP05 책임과 소유권을 식별하는 메커니즘

관련 문서:

- AWS 블로그 게시물 | The agile enterprise
- AWS 블로그 게시물 | Measuring success : A paradox and a plan
- AWS 블로그 게시물 | Letting go : Enabling autonomy in teams
- Centralize or Decentralize?

관련 비디오:

- re:Invent 2023 | How to not sabotage your transformation (SEG201)
- re:Invent 2021 | Amazon Builders' Library: Operational Excellence at Amazon
- Centralization vs. Decentralization

관련 예제:

- 아키텍처 결정 레코드를 사용하여 소프트웨어 개발 프로젝트에 대한 기술적 의사 결정 간소화

OPS03-BP03 에스컬레이션 장려

원하는 성과가 위험하고 예상 기준이 충족되지 않는다고 생각되는 경우 경영진은 팀원이 문제와 우려 사항을 상위 의사 결정권자 및 이해관계자에게 에스컬레이션하도록 장려합니다. 이는 조직 문화의 중요한 요소이며 모든 수준에서 장려됩니다. 위험을 식별하고 인시던트를 방지할 수 있도록 에스컬레이션을 조기에 자주 수행해야 합니다. 리더십은 문제를 에스컬레이션하는 개인을 질책하지 않습니다.

원하는 성과: 조직 전체에서 개인이 높은 수준의 직속 리더십에 문제를 쉽게 에스컬레이션할 수 있습니다. 경영진은 의도적, 의식적으로 팀이 문제를 에스컬레이션해도 괜찮다고 생각해야 한다는 기대를 설정했습니다. 조직 내 각 수준에 문제를 에스컬레이션하는 메커니즘이 있습니다. 직원이 관리자에게 에스컬레이션하면 영향 수준과 문제를 에스컬레이션할지 여부를 공동으로 결정합니다. 에스컬레이션을 시작하려면 직원은 문제 해결을 위한 권장 작업 계획을 포함해야 합니다. 직속 경영진이 적시에 조치를 취하지 않을 경우, 직원들이 조직에 대한 위험으로 인해 에스컬레이션이 정당하다고 느낀다면 최고 수준의 경영진에 문제를 제기하는 것이 좋습니다.

일반적인 안티 패턴:

- 경영진은 클라우드 전환 프로그램 현황 회의에서 문제와 장애 요소가 발생하는 위치를 찾기 위한 충분한 조사를 하지 않습니다. 진행 상태에 대해 좋은 소식만 제시됩니다. CIO는 자신이 좋은 소식만 듣고 싶다는 점을 분명히 했습니다. 문제가 제기되면 CEO는 프로그램이 실패했다고 생각하기 때문입니다.
- 클라우드 운영 엔지니어로서 애플리케이션 팀에서 새로운 지식 관리 시스템을 널리 채택하지 않고 있다는 것을 알게 되었습니다. 회사는 이 새로운 지식 관리 시스템을 구현하기 위해 1년간 수백만 달러를 투자했지만 사람들은 여전히 로컬에서 런북을 작성하고 조직의 클라우드에 공유하고 있기 때문에 지원되는 워크로드와 관련된 지식을 찾기가 어렵습니다. 이 시스템을 지속적으로 사용하면 운영 효율성을 높일 수 있으므로 경영진에 이 사실을 알리세요. 지식 관리 시스템의 구현을 주도하는 책임자에게 이 내용을 전달했을 때, 투자에 의문을 제기한다는 이유로 질책을 당합니다.
- 컴퓨팅 리소스 강화를 담당하는 Infosec 팀은 컴퓨팅 팀이 리소스를 사용할 수 있도록 릴리스하기 전에 EC2 인스턴스의 철저한 보안 유지를 위해 필요한 검사를 수행하는 프로세스를 마련하기로 결정했습니다. 이로 인해 리소스를 배포하는 데 1주일이 더 지연되어 SLA 위반이 발생했습니다. 컴퓨팅 팀은 이를 클라우드 부문 VP에게 에스컬레이션하는 것을 주저합니다. 정보 보안 담당 VP에게 좋지 않기 때문입니다.

이 모범 사례 확립의 이점:

복잡하거나 중요한 문제는 비즈니스에 영향을 미치기 전에 해결됩니다. 낭비되는 시간이 줄어듭니다. 위험이 최소화됩니다. 팀은 문제를 해결할 때 보다 능동적이고 결과에 초점을 맞춥니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

조직의 모든 수준에서 자유롭게 에스컬레이션하려는 의지와 능력은 조직 전체의 모든 수준에서 강조된 교육, 리더십 커뮤니케이션, 기대치 설정, 메커니즘 배포를 통해 의식적으로 개발되어야 하는 조직적, 문화적 기반입니다.

구현 단계

1. 조직에 대한 정책, 표준 및 기대치를 정의합니다.
   a. 정책, 기대치 및 표준에 대한 폭넓은 도입과 이해를 보장합니다.
2. 표준이 충족되지 않을 때 조기에 자주 에스컬레이션할 수 있도록 직원을 격려하고 교육하고 권한을 부여합니다.
3. 빠르고 빈번한 에스컬레이션이 모범 사례임을 조직 차원에서 확인합니다. 에스컬레이션이 근거가 없는 것으로 판명될 수 있으며 에스컬레이션하지 않아 해당 기회를 놓치는 것보다 인시던트를 방지할 기회를 갖는 것이 낫다는 것을 받아들입니다.
   a. 에스컬레이션을 위한 메커니즘을 구축합니다(예: Andon 코드 시스템).
   b. 에스컬레이션이 발생하는 시점과 방법을 정의하는 문서화된 절차가 있어야 합니다.
   c. 조치를 취하거나 승인할 권한이 증가하는 일련의 조직 구성원들을 정의하고 각 이해관계자의 연락처 정보를 포함합니다.
4. 에스컬레이션이 발생하면 팀원이 리더십의 조치를 통해 위험이 완화되었다고 확신할 때까지 에스컬레이션이 계속되어야 합니다.
   a. 에스컬레이션에는 다음이 포함되어야 합니다.
      i. 상황에 대한 설명 및 위험의 특성
      ii. 상황의 중요성
      iii. 영향을 받는 대상
      iv. 영향의 규모
      v. 영향 발생 시 긴급성
      vi. 제안된 구제책 및 완화 계획
   b. 에스컬레이션하는 직원을 보호합니다. 대응 없는 의사 결정권자나 이해관계자에 대해 팀원이 에스컬레이션하는 경우 보복으로부터 팀원을 보호하는 정책을 마련합니다. 이러한 일이 발생하는지 여부를 식별하고 적절하게 대응할 수 있는 메커니즘이 마련되어 있습니다.
5. 조직이 생산하는 모든 것에 대해 지속적인 개선 피드백이 반복되는 문화를 장려합니다. 피드백 루프는 담당자에 대한 가벼운 에스컬레이션으로 작용하며 에스컬레이션이 필요하지 않은 경우에도 개선 기회를 식별합니다. 지속적인 개선 문화는 모든 사람이 보다 능동적으로 행동할 수 있도록 합니다.
6. 리더십은 정책, 표준, 메커니즘 그리고 질책 없는 개방적 에스컬레이션과 지속적인 피드백 루프에 대한 열의를 주기적으로 재강조해야 합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS02-BP05 추가, 변경 및 예외를 요청하는 메커니즘

관련 문서:

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps Guidance | Establish clear escalation paths and encourage constructive disagreement

관련 비디오:

- Jeff Bezos on how to make decisions (& increase velocity)
- Toyota Product System: Stopping Production, a Button, and an Andon Electric Board
- Andon Cord in LEAN Manufacturing

관련 예제:

- Working with escalation plans in Incident Manager

OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션

리더는 특히 조직이 새로운 전략, 기술 또는 업무 방식을 채택할 때 강력하고 효과적인 커뮤니케이션을 만들어내는 역할을 합니다. 리더는 모든 직원이 회사 목표를 향해 노력하도록 기대치를 설정해야 합니다. 리더가 자금을 지원하고 후원하는 계획의 실행을 담당하는 팀 간에 인식을 제고하고 유지할 수 있는 커뮤니케이션 메커니즘을 고안하세요. 조직 간 다양성을 활용하고 여러 가지 고유한 관점에 귀를 기울이세요. 이러한 관점을 통해 혁신을 증진하고, 기존의 추정 사항에 의문을 제기하며, 확증 편향의 위험을 줄일 수 있습니다. 팀 내에서 포용성, 다양성, 접근성을 높여 유익한 관점을 확보하세요.

원하는 성과: 조직에서 변화가 조직에 미치는 영향을 해결하기 위한 커뮤니케이션 전략을 설계합니다. 팀들은 서로 적대적으로 일하기보다는 계속해서 서로 협력할 수 있도록 계속 정보를 얻고 동기를 부여받습니다. 개인은 명시된 목표를 달성하는 데 자신의 역할이 얼마나 중요한지 이해합니다. 이메일은 커뮤니케이션을 위한 수동적인 메커니즘일 뿐이며 적절하게 사용됩니다. 경영진은 개별 기여자와 시간

을 보내어 이들이 자신의 책임, 완료해야 할 업무, 업무가 전체 사명에 기여하는 바를 이해하도록 돕습니다. 필요한 경우 리더는 규모가 작은 장소에 사람들을 직접 불러 메시지를 전달하고 이러한 메시지가 효과적으로 전달되고 있는지 확인합니다. 효과적인 커뮤니케이션 전략의 결과로 조직은 리더가 기대하는 수준 이상의 성과를 거둘 수 있습니다. 리더는 팀 내부 및 여러 팀에 걸쳐 다양한 의견을 내도록 장려하고 다양한 의견을 구합니다.

일반적인 안티 패턴:

- 조직에 모든 워크로드를 AWS로 마이그레이션하기 위한 5년 계획이 있습니다. 클라우드의 비즈니스 사례에는 서버리스 기술을 활용하기 위해 전체 워크로드의 25%를 현대화하는 것이 포함됩니다. CIO는 이 전략을 직속 부하 직원에게 전달하고 각 리더가 직접 대면 커뮤니케이션 없이 관리자, 이사, 개별 기여자에게 이 프레젠테이션을 전달할 것으로 기대합니다. CIO는 한 걸음 물러서서 조직에서 새로운 전략을 수행하기를 기대합니다.
- 리더는 피드백을 위한 메커니즘을 제공하거나 사용하지 않으며, 기대치에 대한 격차가 커져 프로젝트가 지연됩니다.
- 직원은 보안 그룹을 변경하라는 임무는 받았지만 변경이 필요한 사항, 변경이 모든 워크로드에 미칠 수 있는 영향, 변경 시기에 대한 세부 정보는 받지 못합니다. 관리자가 InfoSec 담당 VP가 보낸 이메일을 전달하고 'Make this happen.' 메시지를 추가합니다.
- 계획된 현대화 수를 25%에서 10%로 줄이도록 마이그레이션 전략이 변경되었습니다. 이는 운영 조직에 영향을 미칩니다. 운영 조직은 이러한 전략적 변화에 대해 알지 못했기 때문에 더 많은 워크로드를 AWS로 리프트 앤 시프트하기에 숙련된 인력을 충분히 갖추지 못했습니다.

이 모범 사례 확립의 이점:

- 조직은 새로운 전략이나 변경된 전략에 대해 잘 알게 되며 리더가 설정한 전체 목표와 지표를 달성하도록 서로 돕겠다는 강한 동기를 가지고 그에 따라 행동합니다.
- 알려진 위험과 예정된 이벤트를 팀원에게 적시에 알리는 데 사용되는 메커니즘이 마련됩니다.
- 조직은 필요한 기술 역량과 함께 새로운 업무 방식(사람, 조직, 프로세스 또는 기술의 변화 포함)을 더욱 효과적으로 받아들이고 비즈니스 이점을 더 빠르게 실현합니다.
- 팀원은 수신되는 커뮤니케이션의 필수 컨텍스트를 파악하여 업무를 보다 효과적으로 수행할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

이 모범 사례를 구현하려면 조직 전체의 이해관계자와 협력하여 통신 표준에 동의해야 합니다. 이러한 표준을 조직에 공개적으로 알리세요. 중요한 IT 전환이 발생할 경우, 계획 팀이 구성되어 있다면 이러한 관행을 무시하는 조직보다 변화가 직원에게 미치는 영향을 더 성공적으로 관리할 수 있습니다. 대규모 조직은 변화를 관리하기가 더 어려울 수 있습니다. 새로운 전략에 대해 모든 개별 기여자로부터 강력한 동의를 얻는 것이 중요하기 때문입니다. 이러한 전환 계획 팀이 없는 경우 효과적인 커뮤니케이션에 대한 책임은 전적으로 리더에게 있습니다. 전환 계획 팀을 구성할 때는 모든 조직 리더와 협력하여 모든 수준에서 효과적인 커뮤니케이션을 정의하고 관리할 팀을 배정하세요.

고객 사례

AnyCompany Retail은 AWS Enterprise Support에 가입했으며 클라우드 운영은 다른 서드파티 공급업체에 맡기고 있습니다. 운영 활동을 위한 주요 커뮤니케이션 매체로는 채팅과 ChatOps를 사용합니다. 알림 및 기타 정보가 특정 채널로 전송됩니다. 누군가가 조치를 취해야 할 때 원하는 성과를 명확하게 명시하며, 대부분의 경우 참조할 수 있는 런북이나 플레이북이 제공됩니다. 이들은 변경 달력을 사용하여 프로덕션 시스템에 대한 주요 변경을 예약합니다.

구현 단계

1. 조직 내의 여러 수준에서 발생하는 변화에 대한 커뮤니케이션 계획을 수립하고 시작할 책임이 있는 핵심 팀을 조직 내에 구성합니다.
2. 단일 스레드 소유권을 도입하여 감독을 확보합니다. 독립적으로 혁신할 수 있는 역량을 개별 팀에 부여하고 독립성과 메커니즘의 일관된 사용 간에 균형을 유지합니다. 이렇게 하면 검사와 방향성을 적절한 수준으로 유지할 수 있습니다.
3. 조직 전체의 이해관계자와 협력하여 커뮤니케이션 표준, 관행 및 계획에 대해 합의를 이룹니다.
4. 핵심 커뮤니케이션 팀이 조직 및 프로그램 리더와 협력하여 리더를 대신해 적절한 직원에게 보낼 메시지를 작성하는지 확인합니다.
5. 팀원이 취해야 할 조치에 대해 적절한 기대치를 갖도록 공지, 일정 공유, 전사 회의, 대면 또는 일대일 방식을 통해 변화를 관리하는 전략적 커뮤니케이션 메커니즘을 구축합니다.
6. 조치가 필요한지 판단하는 데 필요한 맥락, 세부 정보 및 가능한 경우 시간을 안내합니다. 조치가 필요한 경우 필요한 조치와 그 영향을 알립니다.
7. 내부 채팅, 이메일, 지식 관리와 같은 전술적 커뮤니케이션을 촉진하는 도구를 도입합니다.
8. 모든 커뮤니케이션이 원하는 성과로 이어지는지 측정하고 검증하는 메커니즘을 구현합니다.
9. 모든 커뮤니케이션의 효과를 측정하는 피드백 루프를 구축합니다. 특히 커뮤니케이션이 조직 전반에서 변화에 대한 저항과 관련된 경우 피드백 루프가 더 중요합니다.

10. 모든 AWS 계정 계정에서 청구, 보안 및 운영을 위한 대체 연락처를 설정합니다. 각 연락처를 개인의 연락처가 아닌 이메일 목록으로 설정하는 것이 가장 좋습니다.
11. 에스컬레이션 및 역에스컬레이션 커뮤니케이션 계획을 수립하여 내부 팀 및 AWS 지원 팀과 기타 서드파티 제공업체를 포함한 외부 팀과 협력합니다.
12. 각 혁신 프로그램의 전체 기간에 일관되게 커뮤니케이션 전략을 시작하고 수행합니다.
13. 가능한 경우 반복 가능한 작업에 높은 우선순위를 지정하여 대규모로 안전하게 자동화합니다.
14. 자동화된 작업이 포함된 시나리오에서 커뮤니케이션이 필요한 경우 커뮤니케이션은 팀에 정보 제공 또는 감사를 위한 것이거나 변경 관리 프로세스의 일부여야 합니다.
15. 알림 시스템의 커뮤니케이션을 분석하여 오탐이나 지속적으로 생성되는 알림이 있는지 확인합니다. 사람의 개입이 필요할 때 시작되도록 이러한 알림을 제거하거나 변경합니다. 알림이 시작되면 런북 또는 플레이북을 제공합니다.
    a. AWS Systems Manager 문서를 사용하여 알림용 플레이북과 런북을 작성할 수 있습니다.
16. 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 Chatbot은 조직 메시징 플랫폼에서 알림을 보내고 이벤트에 응답하는 데 사용할 수 있습니다.
17. 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. 이 도구는 팀원에게 안전하게 변경할 수 있는 시점을 알려줍니다.
18. 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.
19. 다양한 의견과 관점 모색: 모든 사람이 기여하도록 장려합니다. 소외된 그룹에 커뮤니케이션의 기회를 제공합니다. 회의에서 역할과 책임을 교대로 맡습니다.
    a. 역할 및 책임 확대: 다른 상황에서는 맡을 수 없는 역할을 맡을 기회를 팀원에게 제공합니다. 팀원은 다른 상황에서는 불가능할 수 있는 새로운 팀원과의 상호 작용 및 역할에서 경험과 관점을 얻습니다. 또한 자신의 경험과 관점을 바탕으로 새로 맡은 역할을 수행하고 새로운 팀원과 상호 작용합니다. 관점이 발전함에 따라 새로 나타나는 비즈니스 기회나 새로운 개선 기회를 찾습니다. 팀 내에서 대개 다른 사람들이 수행하는 일반적인 업무를 팀원들이 번갈아 맡도록 하여 해당 업무 수행의 요구 사항과 영향을 이해하도록 합니다.

b. 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. 조직의 다양성이 높을수록 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 커집니다.

c. 팀원의 완전한 참여 장려: 직원이 모든 업무 관련 활동에 완전히 참여하는 데 필요한 리소스를 제공합니다. 일상적인 문제에 직면하는 팀원들은 이러한 문제를 해결할 수 있는 기술 역량을 개발합니다. 고유하게 개발된 이러한 기술 역량은 조직에 상당한 이점을 가져올 수 있습니다. 팀원들에게 필요한 편의를 제공하면 그들의 조력을 통해 얻을 수 있는 혜택이 늘어납니다.

리소스

관련 모범 사례:

- OPS03-BP01 경영진의 후원 제공
- OPS07-BP03 런북을 사용한 절차 수행
- OPS07-BP04 플레이북을 사용하여 문제 조사

관련 문서:

- AWS 블로그 게시물 | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | 복잡성이 아닌 혁신을 확대하는 방법 배우기 | 단일 스레드 리더
- AWS 보안 공지
- Open CVE
- 지원 App in Slack to Manage Support Cases
- 채팅 애플리케이션 내 Amazon Q Developer를 사용하여 Slack 채널의 AWS 리소스 관리

관련 서비스:

- 채팅 애플리케이션의 Amazon Q Developer
- AWS Systems Manager Change Calendar
- AWS Systems Manager Documents

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 성과:

- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 계정 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 생성할 수 있는 기능을 제공합니다.
   b. AWS Lambda 버전을 사용하여 베타 테스트를 위해 새로운 함수 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
- Best practices for creating and managing sandbox accounts in AWS
- Create a Culture of Experimentation Enabled by the Cloud
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros
- Experiment More, Fail Less
- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU
- Using AWS AppConfig Feature Flags

관련 비디오:

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)
- Programmatically Create an AWS 계정 with AWS Control Tower
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

관련 예제:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce

관련 서비스:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 팀원의 기술 역량 유지와 강화 장려

팀은 새로운 기술을 도입하고 워크로드 지원 책임과 수요 변화를 지원하기 위해 기술 역량을 키워야 합니다. 새로운 기술 영역의 기술 역량 증진은 팀원의 만족도를 높이고 혁신을 뒷받침합니다. 발전하는 기술 역량을 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, AWS Well-Architected Labs 등의 리소스를 제공합니다. 이러한 리소스에서는 팀을 대상으로 교육을 진행하는 데 활용할 수 있는 지침, 예제 및 자세한 연습 과정을 제공합니다.

지원, AWS re:Post, 지원 Center, AWS 설명서와 같은 리소스는 기술적 장애물을 제거하고 운영을 개선하는 데 도움이 됩니다. 문의 사항에 대해 지원을 받으려면 지원 센터를 통해 지원에 지원을 요청하세요.

AWS는 Amazon Builders' Library에서 AWS의 운영을 통해 학습한 모범 사례와 패턴을 공유하며, AWS 블로그 및 공식 AWS 팟캐스트를 통해 도움이 되는 방대한 기타 교육 자료를 제공합니다.

AWS 교육 및 자격증에서는 역할 또는 영역별 학습 계획과 함께 자습형 디지털 과정을 통한 무료 교육이 제공됩니다. 강사 주도형 교육에 등록하여 팀이 AWS 기술을 연마하도록 추가로 지원할 수도 있습니다.

원하는 성과: 조직은 지속적으로 기술 격차를 평가하고 체계적인 예산과 투자를 통해 기술 격차를 해소합니다. 팀은 업계 최고의 자격증 취득과 같은 기술 역량 향상 활동을 통해 팀원을 격려하고 인센티브를 제공합니다. 또한 점심 시간을 활용한 학습, 이머전 데이, 해커톤, 게임 데이와 같이 서로 지식을 서로 공유하는 전용 프로그램을 활용합니다. 조직은 신입 사원 온보딩 교육을 포함하여 팀원이 서로 교육하는 데 활용할 수 있도록 지식 시스템을 최신 상태로 유지합니다.

일반적인 안티 패턴:

- 체계적인 교육 프로그램과 예산이 없는 상황에서 기술 진화에 보조를 맞추려는 팀은 불확실성을 경험하게 되며, 이로 인해 퇴사율이 증가합니다.
- AWS로 마이그레이션하는 과정에서 팀 간의 기술 역량 격차와 각기 다른 클라우드 유창성이 드러납니다. 기술 역량을 향상시키려는 노력이 없으면 비효율적인 레거시 클라우드 환경을 관리하는 업무가 과중하게 주어지고, 이로 인해 운영자의 수고가 증가합니다. 이러한 번아웃으로 직원들의 불만이 늘어납니다.

이 모범 사례 확립의 이점: 조직이 팀의 기술 향상에 의식적으로 투자하면 클라우드 채택 및 최적화를 가속화하고 규모를 조정하는 데도 도움이 됩니다. 대상이 정해진 학습 프로그램을 통해 혁신을 이끌고 팀이 이벤트 처리에 대비할 수 있도록 운영 능력을 갖추게 됩니다. 팀은 모범 사례를 구현하고 발전시키기 위해 의식적으로 노력합니다. 팀 사기가 높고 팀원들은 비즈니스에 대한 기여도를 중요하게 생각합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

새로운 기술을 도입하고, 혁신을 촉진하며, 수요 및 책임의 변화에 대응하여 워크로드를 지원하려면 팀의 전문적 성장에 지속적으로 투자해야 합니다.

구현 단계

1. 구조화된 클라우드 옹호 프로그램 사용: AWS Skills Guild는 클라우드 기술 역량에 대한 자신감을 높이고 지속적인 학습의 문화를 활성화하기 위한 컨설팅 교육을 제공합니다.
2. 교육 리소스 제공: 교육 자료 및 실습 리소스를 정해진 시간에 전용으로 제공하고, 교육자 및 동료로부터 배울 수 있는 컨퍼런스 참여 및 전문 기관 이용을 지원합니다. 주니어 팀원들이 시니어 팀원들을 멘토로 만나게 하거나, 시니어가 일할 때 주니어 팀원들이 따라다니며 업무 방식과 기술 역량을

접할 수 있게 합니다. 보다 넓은 시야를 확보하기 위해 작업과 직접적으로 관련되지 않은 콘텐츠에 대해 학습하도록 장려합니다.

3. 전문 기술 리소스 활용 장려: AWS re:Post와 같은 리소스를 활용하여 선별된 지식을 얻고 활발한 커뮤니티에 참여합니다.
4. 최신 지식 리포지토리 구축 및 유지 관리: Wiki 및 런북과 같은 지식 공유 플랫폼을 사용합니다. AWS re:Post Private을 통해 재사용 가능한 전문 지식 소스를 만들어 협업을 간소화하고 생산성을 개선하며 직원 온보딩을 가속화합니다.
5. 팀 교육 및 팀 간 참여: 팀원의 지속적인 교육 요구 사항에 대비하여 계획을 세웁니다. 팀원이 다른 팀(임시로 또는 영구적으로)에 합류하여 전체 조직에 도움이 되는 기술과 모범 사례를 공유할 수 있는 기회를 제공합니다.
6. 업계 자격증 획득 및 유지 지원: 팀원이 배운 내용을 검증하고 그 성과를 인정하는 업계 자격증을 획득하고 유지하도록 지원합니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- OPS03-BP01 경영진의 후원 제공
- OPS11-BP04 지식 관리 수행

관련 문서:

- AWS Whitepaper | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS 교육 및 인증
- 지원
- AWS re:Post
- AWS 시작하기 리소스 센터
- AWS 블로그
- AWS 클라우드 규정 준수
- AWS 설명서

- The Official AWS Podcast.
- AWS Online Tech Talks
- AWS 이벤트 및 웨비나
- AWS Well-Architected Labs
- The Amazon Builders' Library

관련 비디오:

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs
- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 팀에 적절한 리소스 제공

적절한 인원의 능숙한 팀원을 배치하고 워크로드 요구 사항을 충족하는 도구와 리소스를 제공하세요. 팀원에게 과도한 업무를 부여하면 인적 오류가 발생할 위험이 커집니다. 자동화와 같은 도구 및 리소스에 투자하면 팀의 효율성을 확대하고 팀이 추가 용량 없이도 더 많은 워크로드를 지원하도록 도울 수 있습니다.

원하는 성과:

- 마이그레이션 계획에 따라 AWS에서 워크로드를 운영하는 데 필요한 기술 역량을 습득할 수 있도록 팀에 인력을 적절히 배치했습니다. 마이그레이션 프로젝트가 진행되는 동안 팀의 규모가 커짐에 따라 팀은 기업에서 애플리케이션을 마이그레이션하거나 현대화할 때 사용하려는 핵심 AWS 기술을 능숙하게 활용할 수 있게 되었습니다.
- 자동화와 워크플로를 활용하여 리소스를 효율적으로 사용할 수 있도록 인력 배치 계획을 세심하게 조정했습니다. 이제 소규모 팀이 애플리케이션 개발 팀을 대신하여 더 많은 인프라를 관리할 수 있습니다.
- 운영 우선순위가 바뀌면서 리소스 인력 배치 제약을 사전에 파악하여 비즈니스 이니셔티브가 문제 없이 성공하도록 합니다.
- 운영 부담을 보고하는 운영 지표(예: 당직 근무로 인한 피로 또는 과도한 통화)를 검토하여 직원이 부담을 느끼지 않는지 확인합니다.

일반적인 안티 패턴:

- 다년간의 클라우드 마이그레이션 계획이 임박해져도 직원들의 AWS 기술 역량 수준이 향상되지 않습니다. 이로 인해 워크로드를 지원할 수 없게 되고 직원의 사기가 저하됩니다.

- 전체 IT 조직이 애자일 업무 방식으로 전환하고 있습니다. 기업에서는 제품 포트폴리오에 우선순위를 두고 어떤 기능을 먼저 개발해야 하는지에 대한 지표를 설정하고 있습니다. 애자일 프로세스에서는 팀이 업무 계획에 스토리 포인트를 할당할 필요가 없습니다. 따라서 다음 업무량에 필요한 용량 수준을 알 수 없거나 해당 업무에 적합한 기술 역량을 가진 사람을 배정했는지 알 수 없습니다.
- AWS 파트너에게 워크로드를 마이그레이션하도록 요청하고 있으며, 파트너가 마이그레이션 프로젝트를 완료한 후에는 팀을 위한 지원 전환 계획이 없습니다. 팀은 워크로드를 효율적이고 효과적으로 지원하는 데 어려움을 겪고 있습니다.

이 모범 사례 확립의 이점: 조직에 워크로드를 지원할 수 있는 적절한 기술을 갖춘 팀원이 보유할 수 있습니다. 리소스 배정은 성능에 영향을 주지 않고 변화하는 우선순위에 맞게 조정할 수 있습니다. 그 결과 팀은 고객 혁신에 집중할 시간을 최대화하면서 워크로드를 능숙하게 지원하므로 직원 만족도가 높아집니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

클라우드 마이그레이션을 위한 리소스 계획은 마이그레이션 계획과 새로운 클라우드 환경을 지원하기 위해 구현되는 원하는 운영 모델에 부합하도록 조직 수준에서 세워야 합니다. 여기에는 비즈니스 및 애플리케이션 개발 팀에 배포되는 클라우드 기술을 이해하는 것을 포함해야 합니다. 인프라 및 운영 부문의 리더는 클라우드 도입을 주도하는 엔지니어의 기술 역량 격차 분석, 교육 및 역할 정의를 계획해야 합니다.

구현 단계

1. 직원 생산성과 같은 관련 운영 지표를 사용하여 팀의 성공에 대한 성공의 기준을 정의합니다(예: 워크로드 지원 비용 또는 인시던트 발생 시 소요된 운영자 시간).
2. 리소스 용량 계획 및 검사 메커니즘을 정의하여 적격 용량의 적절한 균형을 필요할 때 사용할 수 있고 시간이 지남에 따라 조정할 수 있는지 확인합니다.
3. 팀에 영향을 미치는 업무 관련 문제(예: 책임 증가, 기술 변화, 인력 손실, 지원 고객 증가 등)를 이해하기 위한 메커니즘을 만듭니다(예: 매달 팀에 설문조사 전송).
4. 이러한 메커니즘을 사용하여 팀과 소통하고 직원 생산성 문제를 야기할 수 있는 추세를 파악합니다. 팀이 외부 요인의 영향을 받는 경우 목표를 재평가하고 적절하게 타겟을 조정합니다. 팀의 업무 진행을 방해하는 장애물을 파악합니다.
5. 현재 프로비저닝된 리소스가 여전히 충분한지, 추가 리소스가 필요한지 정기적으로 검토하고 팀에 맞게 조정하여 지원합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS03-BP06 팀원의 기술 역량 유지와 강화 장려
- OPS09-BP03 운영 지표 검토 및 개선 우선순위 지정
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용
- OPS10-BP07 이벤트 대응 자동화

관련 문서:

- AWS 클라우드 Adoption Framework: People Perspective
- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth
- 성과 좋은 조직 - Amazon의 2-피자 팀
- How Cloud-Mature Enterprises Succeed

준비

Questions

- OPS 4. 워크로드에 어떻게 관찰성을 구현하나요?
- OPS 5. 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하고 있나요?
- OPS 6. 배포 위험을 어떻게 최소화하고 있나요?
- OPS 7. 귀사가 워크로드를 지원할 준비가 되어있는지 어떻게 알 수 있나요?

OPS 4. 워크로드에 어떻게 관찰성을 구현하나요?

워크로드에 관찰성을 구현하여 상태를 파악하고 비즈니스 요구 사항에 따라 데이터 기반 결정을 내릴 수 있습니다.

모범 사례

- OPS04-BP01 핵심 성과 지표 파악

- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS04-BP03 사용자 경험 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현
- OPS04-BP05 분산 추적 구현

OPS04-BP01 핵심 성과 지표 파악

워크로드에 관찰성을 구현하는 것은 워크로드의 상태를 이해하고 비즈니스 요구 사항에 따라 데이터에 기반한 결정을 내리는 것에서 시작됩니다. 모니터링 활동과 비즈니스 목표를 일치시키는 가장 효과적인 방법 중 하나는 핵심 성과 지표(KPI)를 정의하고 모니터링하는 것입니다.

원하는 성과: 비즈니스 목표와 긴밀하게 연계된 효율적인 관찰성 관행을 통해 모니터링 노력이 항상 가시적인 비즈니스 성과에 도움이 되도록 합니다.

일반적인 안티 패턴:

- 정의되지 않은 KPI: 명확한 KPI 없이 작업하면 모니터링이 너무 많거나 너무 적어 중요한 신호가 누락될 수 있습니다.
- 고정 KPI: 워크로드 또는 비즈니스 목표의 변화에 따라 KPI를 재검토하거나 수정하지 않습니다.
- 불일치: 비즈니스 성과와 직접적인 상관관계가 없거나 실제 문제와 연관시키기 어려운 기술 지표에 초점을 맞춥니다.

이 모범 사례 확립의 이점:

- 손쉬운 문제 식별: 비즈니스 KPI는 종종 기술적 지표보다 문제를 더 명확하게 드러냅니다. 비즈니스 KPI를 낮게 설정하면 수많은 기술적 지표를 살펴보는 것보다 더 효과적으로 문제를 찾아낼 수 있습니다.
- 비즈니스 조정: 모니터링 활동이 비즈니스 목표를 직접 지원하도록 합니다.
- 효율성: 모니터링 리소스와 중요한 지표에 대한 관심을 우선시합니다.
- 사전 조치: 문제가 비즈니스에 더 광범위하게 영향을 미치기 전에 문제를 파악하고 해결합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드 KPI를 효과적으로 정의하는 방법:

1. 비즈니스 성과부터 시작: 지표를 자세히 살펴보기 전에 원하는 비즈니스 성과를 파악합니다. 매출 증대, 사용자 참여 증대 또는 응답 시간 단축이 필요한가요?
2. 기술 지표와 비즈니스 목표의 상관관계 파악: 모든 기술 지표가 비즈니스 성과에 직접적인 영향을 미치는 것은 아닙니다. 비즈니스 성과에 직접적인 영향을 미치는 기술 지표를 파악하세요. 하지만 비즈니스 KPI를 사용하여 문제를 식별하는 것이 더 간단한 경우가 많습니다.
3. Amazon CloudWatch 사용: CloudWatch를 사용하여 KPI를 나타내는 지표를 정의하고 모니터링합니다.
4. 정기적으로 KPI 검토 및 업데이트: 워크로드와 비즈니스가 진화함에 따라 적절한 KPI를 유지합니다.
5. 이해관계자 참여: KPI를 정의하고 검토하는 데 기술 팀과 비즈니스 팀 모두를 참여시킵니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- the section called “OPS04-BP02 애플리케이션 원격 측정 구현”
- the section called “OPS04-BP03 사용자 경험 원격 측정 구현”
- the section called “OPS04-BP04 종속성 원격 측정 구현”
- the section called “OPS04-BP05 분산 추적 구현”

관련 문서:

- AWS Observability Best Practices
- CloudWatch 사용 설명서
- AWS Observability Skill Builder 과정

관련 비디오:

- Developing an observability strategy

관련 예제:

- One Observability 워크숍

OPS04-BP02 애플리케이션 원격 측정 구현

애플리케이션 원격 측정은 워크로드를 관찰하기 위한 기반입니다. 애플리케이션 상태와 기술 및 비즈니스 성과 달성에 대한 실행 가능한 인사이트를 제공하는 원격 분석을 내보내는 것이 중요합니다. 문제 해결부터 새로운 기능의 영향 측정 또는 비즈니스 핵심 성과 지표(KPI)와의 조정에 이르기까지 애플리케이션 원격 측정은 워크로드를 구축, 운영 및 발전시키는 방법을 알려줍니다.

지표, 로그, 추적은 관찰성의 세 가지 기본 원칙을 형성합니다. 이들은 애플리케이션의 상태를 설명하는 진단 도구 역할을 합니다. 시간이 지남에 따라 기준을 만들고 이상 징후를 식별하는 데 도움을 줍니다. 그러나 모니터링 활동과 비즈니스 목표를 일치시키기 위해서는 KPI를 정의하고 모니터링하는 것이 중요합니다. 비즈니스 KPI는 기술 지표만 사용하는 것보다 문제를 더 쉽게 식별할 수 있게 해주는 경우가 많습니다.

실제 사용자 모니터링(RUM) 및 가상 트랜잭션과 같은 다른 원격 측정 유형은 이러한 기본 데이터 소스를 보완합니다. RUM은 실시간 사용자 상호 작용에 대한 인사이트를 제공하는 반면 가상 트랜잭션은 잠재적 사용자 행동을 시뮬레이션하여 실제 사용자가 병목 현상을 경험하기 전에 병목 현상을 감지하는 데 도움이 됩니다.

원하는 성과: 워크로드 성능에 대한 실행 가능한 인사이트를 도출합니다. 이러한 인사이트를 통해 성능 최적화에 대한 사전 결정을 내리고, 워크로드 안정성을 높이며, CI/CD 프로세스를 간소화하며, 리소스를 효과적으로 활용할 수 있습니다.

일반적인 안티 패턴:

- 불완전한 관찰성: 워크로드의 모든 레이어에 관찰성을 통합하지 않으면 사각 지대가 발생하여 중요한 시스템 성능 및 동작 인사이트를 모호하게 만들 수 있습니다.
- 단편화된 데이터 보기: 데이터가 여러 도구 및 시스템에 분산되어 있는 경우 워크로드의 상태와 성능을 전체적으로 파악하기가 어려워집니다.
- 사용자가 보고한 문제: 원격 측정 및 비즈니스 KPI 모니터링을 통한 사전 예방적 문제 탐지가 부족하다는 신호입니다.

이 모범 사례 확립의 이점:

- 정보에 입각한 의사 결정: 원격 측정 및 비즈니스 KPI의 인사이트를 바탕으로 데이터에 기반한 결정을 내릴 수 있습니다.
- 운영 효율성 향상: 데이터 기반 리소스 활용은 비용 효율성으로 이어집니다.
- 워크로드 안정성 향상: 문제를 더 빠르게 감지하고 해결하여 가동 시간을 개선합니다.

- 간소화된 CI/CD 프로세스: 원격 측정 데이터에서 얻은 인사이트를 통해 프로세스를 개선하고 신뢰할 수 있는 코드를 전달할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드에 애플리케이션 원격 측정을 구현하기 위해 Amazon CloudWatch 및 AWS X-Ray와 같은 AWS 서비스를 사용하세요. Amazon CloudWatch는 AWS 및 온프레미스 환경에서 리소스와 애플리케이션을 관찰할 수 있는 포괄적인 모니터링 도구 모음을 제공합니다. 지표를 수집, 추적 및 분석하고, 로그 데이터를 통합 및 모니터링하며, 리소스 변화에 대응하여 워크로드 운영 방식에 대한 이해를 높입니다. 동시에 AWS X-Ray를 통해 애플리케이션을 추적, 분석 및 디버깅하여 워크로드 동작을 심층적으로 이해할 수 있습니다. 서비스 맵, 지연 시간 분포, 추적 타임라인과 같은 기능을 통해 AWS X-Ray는 워크로드의 성능과 이에 영향을 미치는 병목 현상에 대한 인사이트를 제공합니다.

구현 단계

1. 수집할 데이터 식별: 워크로드의 상태, 성능 및 행동에 대한 실질적인 인사이트를 제공하는 필수 지표, 로그 및 추적을 확인하세요.
2. CloudWatch 에이전트 배포: CloudWatch 에이전트는 워크로드와 기본 인프라에서 시스템 및 애플리케이션 지표와 로그를 확보하는 데 중요한 역할을 합니다. CloudWatch 에이전트를 사용하여 OpenTelemetry 또는 X-Ray 추적을 수집하여 X-Ray에 전송할 수도 있습니다.
3. 로그 및 지표에 대한 이상 탐지 구현: CloudWatch Logs 이상 탐지 및 CloudWatch 지표 이상 탐지를 사용하여 애플리케이션 운영의 비정상적인 활동을 자동으로 식별합니다. 이러한 도구는 기계 학습 알고리즘을 사용하여 이상 징후를 감지하고 알림을 제공하므로 모니터링 역량이 향상되고 잠재적 장애 또는 보안 위협에 대한 대응 시간이 단축됩니다. 이러한 기능을 설정하여 애플리케이션 상태 및 보안을 사전에 관리하세요.
4. 민감한 로그 데이터 보호: Amazon CloudWatch Logs 데이터 보호를 사용하여 로그 내의 민감한 정보를 마스킹합니다. 이 기능은 액세스하기 전에 민감한 데이터를 자동으로 감지하고 마스킹하여 프라이버시 및 규정 준수를 유지하는 데 도움이 됩니다. 데이터 마스킹을 구현하여 개인 식별 정보(PII)와 같은 민감한 세부 정보를 안전하게 처리하고 보호합니다.
5. 비즈니스 KPI 정의 및 모니터링: 비즈니스 성과에 맞는 사용자 지정 지표를 설정합니다.
6. AWS X-Ray로 애플리케이션 계측: CloudWatch 에이전트를 배포하는 것 외에도 추적 데이터를 내보내도록 애플리케이션을 계측하는 것이 중요합니다. 이 프로세스는 워크로드의 동작과 성능에 대한 추가 인사이트를 제공할 수 있습니다.

7. 애플리케이션 전반의 데이터 수집 표준화: 전체 애플리케이션에서 데이터 수집 관행을 표준화합니다. 일관성은 데이터를 상호 연관시키고 분석하는 데 도움이 되며, 이를 통해 애플리케이션 동작을 포괄적으로 파악할 수 있습니다.
8. 크로스 계정 관찰성 구현: Amazon CloudWatch 크로스 계정 관찰성을 통해 여러 AWS 계정 계정의 모니터링 효율성을 개선합니다. 이 기능을 사용하면 여러 계정의 지표, 로그 및 경보를 단일 보기로 통합하여 관리를 간소화하고 조직의 AWS 환경 전반에서 식별된 문제에 대한 대응 시간을 개선할 수 있습니다.
9. 데이터 분석 및 활용: 데이터 수집 및 정규화가 완료되면 지표 및 로그 분석에는 Amazon CloudWatch를 사용하고 추적 분석에는 AWS X-Ray를 사용합니다. 이러한 분석을 통해 워크로드의 상태, 성능 및 행동에 대한 중요한 인사이트를 얻어 의사 결정 프로세스에 반영할 수 있습니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- OPS04-BP01 워크로드 KPI 정의
- OPS04-BP03 사용자 활동 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현
- OPS04-BP05 트랜잭션 추적 기능 구현

관련 문서:

- AWS Observability Best Practices
- CloudWatch 사용 설명서
- AWS X-Ray 개발자 안내서
- 운영 가시성을 위한 분산 시스템 계측
- AWS Observability Skill Builder 과정
- Amazon CloudWatch의 새로운 소식
- AWS X-Ray의 새로운 소식

관련 비디오:

- AWS re:Invent 2022 - Observability best practices at Amazon

- AWS re:Invent 2022 - Developing an observability strategy

관련 예제:

- One Observability 워크숍
- AWS Solutions Library: Application Monitoring with Amazon CloudWatch

OPS04-BP03 사용자 경험 원격 측정 구현

고객 경험과 애플리케이션과의 상호 작용에 대한 심층적인 인사이트를 얻는 것이 중요합니다. 실제 사용자 모니터링(RUM)과 가상 트랜잭션은 이러한 목적을 위한 강력한 도구 역할을 합니다. RUM은 실제 사용자 상호 작용에 대한 데이터를 제공하여 사용자 만족도에 대한 필터링되지 않은 관점을 제공하는 반면, 가상 트랜잭션은 사용자 상호 작용을 시뮬레이션하여 실제 사용자에게 영향을 미치기 전에 잠재적 문제를 감지하는 데 도움을 줍니다.

원하는 성과: 고객 경험을 총체적으로 파악하고, 문제를 사전에 감지하고, 사용자 상호 작용을 최적화하여 원활한 디지털 경험을 제공합니다.

일반적인 안티 패턴:

- 실제 사용자 모니터링(RUM)이 없는 애플리케이션:
  - 지연된 문제 감지: RUM이 없으면 사용자가 불만을 제기할 때까지 성능 병목 현상이나 문제를 인지하지 못할 수 있습니다. 이러한 사후 대응적 접근 방식은 고객 불만족으로 이어질 수 있습니다.
  - 사용자 경험 인사이트 부족: RUM을 사용하지 않으면 실제 사용자가 애플리케이션과 상호 작용하는 방식을 보여주는 중요한 데이터를 잃게 되어 사용자 경험을 최적화할 수 없게 됩니다.
- 가상 트랜잭션이 없는 애플리케이션:
  - 놓친 엣지 케이스: 가상 트랜잭션을 사용하면 일반 사용자는 자주 사용하지 않지만 특정 비즈니스 기능에 중요한 경로와 기능을 테스트할 수 있습니다. 가상 트랜잭션이 없으면 이러한 경로가 오작동하여 눈에 띄지 않을 수 있습니다.
  - 애플리케이션을 사용하지 않을 때 문제 확인: 정기적인 가상 테스트를 통해 실제 사용자가 애플리케이션과 적극적으로 상호 작용하지 않는 시간을 시뮬레이션하여 시스템이 항상 올바르게 작동하는지 확인할 수 있습니다.

이 모범 사례 확립의 이점:

- 사전 문제 감지: 실제 사용자에게 영향을 미치기 전에 잠재적 문제를 식별하여 해결합니다.

- 최적화된 사용자 경험: RUM의 지속적인 피드백은 전반적인 사용자 경험을 개선하고 향상하는 데 도움이 됩니다.
- 디바이스 및 브라우저 성능에 대한 인사이트: 다양한 디바이스 및 브라우저에서 애플리케이션이 어떻게 작동하는지 파악하여 더욱 최적화할 수 있습니다.
- 검증된 비즈니스 워크플로: 정기적인 가상 트랜잭션을 통해 핵심 기능과 중요 경로가 운영 및 효율성을 유지할 수 있습니다.
- 애플리케이션 성능 향상: 실제 사용자 데이터에서 수집한 인사이트를 활용하여 애플리케이션 응답성과 신뢰성을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

사용자 활동 원격 측정에 RUM 및 가상 트랜잭션을 활용하기 위해 AWS에서 Amazon CloudWatch RUM 및 Amazon CloudWatch Synthetics와 다음과 같은 서비스를 제공합니다. 지표, 로그 및 추적은 사용자 활동 데이터와 결합되어 애플리케이션의 작동 상태와 사용자 경험을 포괄적으로 보여줍니다.

구현 단계

1. Amazon CloudWatch RUM 배포: 애플리케이션을 CloudWatch RUM과 통합하여 실제 사용자 데이터를 수집, 분석 및 제공합니다.
   a. CloudWatch RUM 자바스크립트 라이브러리를 사용하여 RUM을 애플리케이션과 통합합니다.
   b. 대시보드를 설정하여 실제 사용자 데이터를 시각화하고 모니터링할 수 있습니다.
2. CloudWatch Synthetics 구성: 애플리케이션과 사용자 상호 작용을 시뮬레이션하는 canary 또는 스크립팅된 루틴을 만들 수 있습니다.
   a. 중요 애플리케이션 워크플로 및 경로를 정의합니다.
   b. CloudWatch Synthetics 스크립트를 사용하여 이러한 경로에 대한 사용자 상호 작용을 시뮬레이션하도록 canary를 설계합니다.
   c. canary가 지정된 간격으로 실행되도록 스케줄링하고 모니터링하여 일관된 성능 검사를 보장합니다.
3. 데이터 분석 및 조치: RUM 및 가상 트랜잭션의 데이터를 활용하여 인사이트를 얻고 이상이 감지되면 수정 조치를 취하세요. CloudWatch 대시보드와 경보를 사용하여 최신 정보를 확인하세요.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현
- OPS04-BP05 분산 추적 구현

관련 문서:

- Amazon CloudWatch RUM 가이드
- Amazon CloudWatch Synthetics 가이드

관련 비디오:

- Optimize applications through end user insights with Amazon CloudWatch RUM
- AWS on Air ft. Real-User Monitoring for Amazon CloudWatch

관련 예제:

- One Observability 워크숍
- Git Repository for Amazon CloudWatch RUM Web Client
- Using Amazon CloudWatch Synthetics to measure page load time

OPS04-BP04 종속성 원격 측정 구현

종속성 원격 측정은 워크로드가 의존하는 외부 서비스 및 구성 요소의 상태와 성능을 모니터링하는 데 필수적입니다. DNS, 데이터베이스 또는 서드파티 API와 같은 종속성과 관련된 연결성, 시간 초과 및 기타 중요한 이벤트에 대한 귀중한 인사이트를 제공합니다. 이러한 종속성에 대한 지표, 로그 및 추적을 내보내도록 애플리케이션을 계측하면 워크로드에 영향을 미칠 수 있는 잠재적 병목 현상, 성능 문제 또는 장애를 더 명확하게 이해할 수 있습니다.

원하는 성과: 워크로드가 의존하는 종속성이 예상대로 수행되므로 문제를 사전에 해결하고 최적의 워크로드 성능을 보장할 수 있습니다.

일반적인 안티 패턴:

- 외부 종속성 간과: 내부 애플리케이션 지표에만 초점을 맞추고 외부 종속성과 관련된 지표는 무시합니다.
- 사전 모니터링 부족: 종속성 상태 및 성능을 지속적으로 모니터링하는 대신 문제가 발생할 때까지 기다립니다.
- 사일로 모니터링: 여러 개의 다른 모니터링 도구를 사용하면 종속성 상태에 대해 단편적이고 일관성 없는 보기가 발생할 수 있습니다.

이 모범 사례 확립의 이점:

- 워크로드 신뢰성 향상: 외부 종속성을 지속적으로 사용할 수 있고 최적의 성능을 발휘하도록 보장합니다.
- 더 빠른 문제 감지 및 해결: 종속성 관련 문제가 워크로드에 영향을 미치기 전에 사전에 식별하고 해결합니다.
- 포괄적 보기: 워크로드 상태에 영향을 미치는 내부 및 외부 구성 요소를 모두 포괄적으로 파악합니다.
- 워크로드 확장성 향상: 외부 종속 확장성의 한계와 성능 특성을 이해합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드가 의존하는 서비스, 인프라 및 프로세스를 식별하는 것부터 시작하여 종속성 원격 측정을 구현하세요. 이러한 종속성이 예상대로 작동할 때 양호한 조건이 어떻게 보이는지 정량화한 다음 이를 측정하는 데 필요한 데이터를 결정하세요. 이 정보를 사용하여 운영 팀에 이러한 종속성 상태에 대한 인사이트를 제공하는 대시보드 및 알림을 만들 수 있습니다. AWS 도구를 사용하여 종속성이 필요한 만큼 제공할 수 없을 때 미치는 영향을 발견하고 정량화하세요. 전략을 지속적으로 재검토하여 우선순위, 목표 및 얻은 인사이트의 변화를 고려하세요.

구현 단계

종속성 원격 측정을 효과적으로 구현하는 방법:

1. 외부 종속성 식별: 이해관계자와 협업하여 워크로드가 의존하는 외부 종속성을 정확히 파악하세요. 외부 종속성에는 외부 데이터베이스, 서드파티 API, 다른 환경으로의 네트워크 연결 경로, DNS 서비스와 같은 서비스가 포함될 수 있습니다. 효과적인 종속성 원격 측정을 위한 첫 번째 단계는 이러한 종속성이 무엇인지 포괄적으로 이해하는 것입니다.

2. 모니터링 전략 개발: 외부 종속성을 명확하게 파악한 후에는 그에 맞는 모니터링 전략을 세우세요. 여기에는 각 종속성의 중요도, 예상되는 동작, 관련 서비스 수준에 관한 계약 또는 대상(SLA 또는 SLT)을 이해하는 것이 포함됩니다. 사전 알림을 설정하여 상태 변경 또는 성능 편차에 대한 알림을 받습니다.
3. 네트워크 모니터링 사용: 전 세계 인터넷 및 네트워크 상태에 대한 포괄적인 인사이트를 제공하는 Internet Monitor 및 Network Monitor를 사용합니다. 이러한 도구는 외부 종속성에 영향을 미치는 운영 중단, 장애 또는 성능 저하를 이해하고 이에 대응하는 데 도움이 됩니다.
4. AWS Health로 최신 정보를 확인하세요: AWS Health는 AWS 클라우드 리소스 상태에 대한 신뢰할 수 있는 정보 소스입니다. AWS Health를 사용해 계획된 수명 주기 이벤트와 같은 현재 서비스 이벤트 및 예정된 변경 사항을 시각화하고 알림을 받아 영향 완화 조치를 취할 수 있습니다.
   a. AWS 사용자 알림를 통해 이메일 및 채팅 채널에 적합한 AWS Health 이벤트 알림을 생성하고, AWS Health API 또는 Amazon EventBridge를 통해 모니터링 및 알림 도구와 프로그래밍 방식으로 통합할 수 있습니다.
   b. Amazon EventBridge 또는 AWS Health API를 통해 이미 사용할 수 있는 변경 관리 또는 ITSM 도구(예: Jira 또는 ServiceNow)와 통합하여 조치가 필요한 상태 이벤트에 대한 진행 상황을 계획하고 추적하세요.
   c. AWS Organizations를 사용하는 경우 AWS Health에 대한 조직 보기를 활성화하여 계정 간에 AWS Health 이벤트를 집계합니다.
5. AWS X-Ray로 애플리케이션 계측: AWS X-Ray에서는 애플리케이션과 기본 종속성이 어떻게 수행되는지에 대한 인사이트를 제공합니다. 요청을 처음부터 끝까지 추적하여 애플리케이션이 의존하는 외부 서비스 또는 구성 요소의 병목 현상이나 장애를 식별할 수 있습니다.
6. Amazon DevOps Guru 사용: 이 기계 학습 기반 서비스는 운영 문제를 식별하고, 중대한 문제가 발생할 수 있는 시기를 예측하며, 취해야 할 구체적인 조치를 제시합니다. 종속성에 대한 인사이트를 얻고 종속성에서 운영 문제가 발생하지 않도록 하는 데 매우 중요합니다.
7. 정기적으로 모니터링: 외부 종속성과 관련된 지표 및 로그를 지속적으로 모니터링합니다. 예상치 못한 동작이나 성능 저하에 대한 알림을 설정합니다.
8. 변경 후 검증: 외부 종속성이 업데이트되거나 변경될 때마다 성능을 검증하고 애플리케이션 요구 사항에 맞는지 확인합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 워크로드 KPI 정의

- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS04-BP03 사용자 활동 원격 측정 구현
- OPS04-BP05 트랜잭션 추적 기능 구현
- OPS08-BP04 실행 가능한 알림 생성

관련 문서:

- Amazon Personal AWS Health Dashboard 사용 설명서
- AWS Internet Monitor 사용 설명서
- AWS X-Ray 개발자 안내서
- AWS DevOps Guru 사용 설명서

관련 비디오:

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru
- Manage resource lifecycle events at scale with AWS Health

관련 예제:

- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 분산 추적 구현

분산 추적은 분산 시스템의 다양한 구성 요소를 통과하는 요청을 모니터링하고 시각화하는 방법을 제공합니다. 여러 소스에서 추적 데이터를 캡처하고 통합 보기에서 분석함으로써 팀은 요청의 흐름, 병목 현상, 최적화 작업이 집중되는 위치를 더 잘 이해할 수 있습니다.

원하는 성과: 분산 시스템을 통해 흐르는 요청을 전체적으로 파악하여 정확한 디버깅, 최적화된 성능 및 향상된 사용자 경험을 제공합니다.

일반적인 안티 패턴:

- 일관되지 않은 계측: 분산 시스템의 일부 서비스가 추적을 위해 계측되지 않습니다.
- 지연 시간 무시: 오류에만 초점을 맞추고 지연 시간이나 점진적인 성능 저하는 고려하지 않습니다.

이 모범 사례 확립의 이점:

- 포괄적인 시스템 개요: 시작부터 종료까지 요청의 전체 경로를 시각화합니다.
- 향상된 디버깅: 장애 또는 성능 문제가 발생한 위치를 신속하게 식별합니다.
- 향상된 사용자 경험: 실제 사용자 데이터를 기반으로 모니터링 및 최적화하여 시스템이 실제 요구 사항을 충족하는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

먼저 계측이 필요한 워크로드의 모든 요소를 식별합니다. 모든 구성 요소가 고려되면 AWS X-Ray 및 OpenTelemetry와 같은 도구를 활용하여 X-Ray 및 Amazon CloudWatch ServiceLens Map과 같은 도구를 사용하여 분석에 사용할 추적 데이터를 수집할 수 있습니다. 개발자와 정기적으로 검토하고 Amazon DevOps Guru, X-Ray Analytics, X-Ray Insights와 같은 도구를 사용하여 이러한 논의를 보완하여 더 심층적인 결과를 발견하세요. 추적 데이터로부터 알림을 설정하여 워크로드 모니터링 계획에 정의된 대로 결과가 위험에 처했을 때 이를 알립니다.

구현 단계

분산 추적을 효과적으로 구현하는 방법:

1. AWS X-Ray 채택: X-Ray를 애플리케이션에 통합하여 애플리케이션 동작에 대한 인사이트를 얻고 성능을 이해하며 병목 현상을 정확히 찾아내세요. 자동 추적 분석을 위해 X-Ray Insights를 활용하세요.
2. 서비스 계측: AWS Lambda 함수에서 EC2 인스턴스까지 모든 서비스가 추적 데이터를 전송하는지 확인합니다. 더 많은 서비스를 계측할수록 엔드 투 엔드 보기가 더 명확해집니다.
3. CloudWatch 실제 사용자 모니터링 및 가상 모니터링 통합: 실제 사용자 모니터링(RUM) 및 가상 모니터링을 X-Ray와 통합합니다. 이를 통해 실제 사용자 경험을 캡처하고 사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 식별할 수 있습니다.
4. CloudWatch 에이전트 사용: 에이전트는 X-Ray 또는 OpenTelemetry 중 하나에서 트레이스를 전송하여 더 심도 깊은 인사이트를 얻을 수 있습니다.
5. Amazon DevOps Guru 사용: DevOps Guru에서는 X-Ray, CloudWatch, AWS Config, AWS CloudTrail의 데이터를 사용하여 실행 가능한 권장 사항을 제공합니다.
6. 추적 분석: 추적 데이터를 정기적으로 검토하여 애플리케이션 성능에 영향을 줄 수 있는 패턴, 이상 또는 병목 현상을 식별합니다.

7. 알림 설정: CloudWatch에서 비정상적인 패턴이나 연장된 지연 시간에 대한 경보를 구성하여 선제적으로 문제를 해결합니다.
8. 지속적인 개선: 모든 관련 데이터 포인트를 캡처하도록 서비스가 추가 또는 수정되면 추적 전략을 재검토합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS04-BP03 사용자 경험 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현

관련 문서:

- AWS X-Ray 개발자 안내서
- Amazon CloudWatch Agent 사용 설명서
- Amazon DevOps Guru 사용 설명서

관련 비디오:

- Use AWS X-Ray Insights
- AWS on Air ft. Observability: Amazon CloudWatch and AWS X-Ray

관련 예제:

- AWS X-Ray용 애플리케이션 계측

OPS 5. 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하고 있나요?

프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정을 지원해야 합니다. 이렇게 하면 유용한 변경 사항을 프로

덕션 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 구축 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 소규모 변경 자주 적용
- OPS05-BP10 통합 및 배포 완전 자동화

OPS05-BP01 버전 관리 사용

버전 관리를 사용하여 변경 사항과 릴리스를 추적합니다.

많은 AWS 서비스가 버전 관리 기능을 제공합니다. 리비전 또는 소스 제어 시스템(예: Git)을 사용하여 코드와 기타 아티팩트(예: 인프라의 버전 제어 AWS CloudFormation 템플릿)를 관리합니다.

원하는 성과: 팀이 코드를 사용하여 협업할 수 있습니다. 코드를 병합할 때 코드가 일관되고 변경 내용이 손실되지 않습니다. 올바른 버전 관리를 통해 오류를 쉽게 되돌릴 수 있습니다.

일반적인 안티 패턴:

- 코드는 워크스테이션에서 개발 및 저장해 왔습니다. 워크스테이션에서 복구할 수 없는 스토리지 오류가 발생하면 코드가 손실되었습니다.
- 기존 코드를 변경 사항으로 덮어쓴 후 애플리케이션을 다시 시작하면 애플리케이션이 더 이상 작동하지 않습니다. 이 경우 변경 사항을 되돌릴 수 없습니다.
- 다른 사람이 편집해야 하는 보고서 파일에 대한 쓰기 잠금이 있습니다. 작업을 완료할 수 있도록 해당 작업의 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 확립의 이점: 버전 관리 기능을 사용하면 쉽게 알려진 정상 상태와 이전 버전으로 되돌리고 자산 손실 위험을 제한할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

리소스

관련 모범 사례:

- OPS05-BP04 구축 및 배포 관리 시스템 사용

관련 비디오:

- AWS re:Invent 2,023 - How Lockheed Martin builds software faster, powered by DevSecOps
- AWS re:Invent 2,023 - How GitHub operationalizes AI for team collaboration and productivity

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 성과: 소프트웨어 변경 사항이 제공되기 전에 테스트됩니다. 개발자가 테스트 결과 및 검증에 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 감소합니다. 소프트웨어 품질이 개선됩니다. 개발자가 코드의 가시성에 대한 인식을 높였습니다. 조직의 규정 준수를 지원한다는 확신을 가지고 보안 정책을 롤아웃할 수 있습니다. 트래픽 수요를 충족하기 위해 자동 조정 정책 업데이트 등과 같은 인프라 변경을 사전에 테스트합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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.

Amazon Q Developer를 통해 생성형 AI의 성능을 활용하여 개발자 생산성과 코드 품질을 개선합니다. Amazon Q Developer에는 코드 제안 생성(대규모 언어 모델 기반), 단위 테스트 생성(경계 조건 포함), 보안 취약성 탐지 및 해결을 통한 코드 보안 강화 기능이 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?

1. AWS 배포 파이프라인 참조 아키텍처에는 통합 파이프라인의 일부로 소프트웨어 아티팩트에 대해 수행할 수 있는 신뢰할 수 있는 테스트 유형 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. 단위 테스트 사례(경계 조건 포함)를 생성하고, 코드 및 주석을 사용하여 함수를 생성하며, 잘 알려진 알고리즘을 구현하는 데 도움이 되는 생성형 AI 도구인 Amazon Q Developer를 사용합니다.

   b. Amazon CodeGuru Reviewer를 사용하여 애플리케이션 코드에 결함이 있는지 테스트합니다.

   c. AWS CodeBuild를 사용하여 소프트웨어 아티팩트에 대한 테스트를 수행할 수 있습니다.

   d. AWS CodePipeline에서는 소프트웨어 테스트를 파이프라인으로 오케스트레이션할 수 있습니다.

리소스

관련 모범 사례:

- OPS05-BP01 버전 관리 사용
- OPS05-BP06 설계 표준 공유
- OPS05-BP07 코드 품질 개선을 위한 사례 구현
- OPS05-BP10 통합 및 배포 완전 자동화

관련 문서:

- 테스트 기반 개발 접근 방식 채택
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Automated AWS CloudFormation Testing Pipeline with TaskCat and CodePipeline

- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- AWS에서 지속적 통합 및 지속적 전달 적용 백서

관련 비디오:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development
- Testing Your Infrastructure as Code with AWS CDK

관련 리소스:

- AWS Deployment Pipeline Reference Architecture - Application
- AWS Kubernetes DevSecOps Pipeline
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

관련 서비스:

- Amazon Q Developer
- 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. 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 변경할 수 있습니다.

구성은 알려진 일관된 상태로 배포해야 합니다. 여러 환경 및 리전에서 리소스 구성을 지속적으로 모니터링하려면 자동 검사를 사용해야 합니다. 규칙이 여러 환경에서 일관되게 적용되도록 하려면 이러한 제어를 코드로 정의하고 관리를 자동화해야 합니다. 구성 변경은 합의된 변경 관리 절차를 통해 업데이트되고 버전 관리를 준수하며 일관되게 적용되어야 합니다. 애플리케이션 구성은 애플리케이션 및 인프라 코드와 독립적으로 관리해야 합니다. 이를 통해 여러 환경에서 일관되게 배포할 수 있습니다. 구성 변경으로 인해 애플리케이션이 재구축되거나 재배포되지는 않습니다.

원하는 성과: 지속적 통합 및 지속적 전달(CI/CD) 파이프라인의 일부로 구성, 검증 및 배포합니다. 모니터링하여 구성이 올바른지 확인합니다. 이를 통해 최종 사용자와 고객에게 미치는 영향을 최소화할 수 있습니다.

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.
- 검증 없이 CI/CD를 통해 사전 프로덕션 구성을 프로덕션 환경으로 푸시합니다. 사용자와 고객을 잘못된 데이터와 서비스에 노출시킵니다.

이 모범 사례 확립의 이점: 구성 관리 시스템을 도입하면 변경 수행 및 추적을 위한 작업량과 수동 절차로 인한 오류 발생 빈도가 줄어듭니다. 구성 관리 시스템은 거버넌스, 규정 준수 및 규제 요구 사항과 관련하여 보증을 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

구성 관리 시스템은 애플리케이션 및 환경 구성의 변경 사항을 추적하고 구현하는 데 사용됩니다. 또한 구성 관리 시스템은 수동 프로세스로 인한 오류를 줄이고, 구성 변경을 반복 및 감사할 수 있도록 하며, 작업량을 감소시킵니다.

AWS에서 AWS Config를 사용하여 여러 계정 및 리전에서 AWS 리소스 구성을 지속적으로 모니터링할 수 있습니다. 이를 통해 구성 이력을 추적하고, 구성 변경이 다른 리소스에 어떤 영향을 미치는지 이해하며, AWS Config 규칙 및 AWS Config Conformance Packs를 사용해 예상되는 구성이나 원하는 구성을 기준으로 해당 구성을 감사할 수 있습니다.

Amazon EC2 인스턴스, AWS Lambda, 컨테이너, 모바일 애플리케이션 또는 IoT 디바이스에서 실행되는 애플리케이션에 동적 구성이 있는 경우 AWS AppConfig를 사용하여 여러 환경에서 애플리케이션을 구성, 검증, 배포 및 모니터링할 수 있습니다.

구현 단계

1. 구성 소유자를 식별합니다.
   a. 구성 소유자에게 모든 규정 준수, 거버넌스 또는 규제 요구 사항을 알립니다.
2. 구성 항목 및 결과물을 식별합니다.
   a. 구성 항목은 CI/CD 파이프라인 내 배포의 영향을 받는 모든 애플리케이션 및 환경 구성입니다.
   b. 결과물에는 성공 기준, 검증, 모니터링 대상 등이 포함됩니다.
3. 비즈니스 요구 사항 및 제공 파이프라인에 따라 구성 관리를 위한 도구를 선택합니다.
4. 잘못된 구성으로 인한 영향을 최소화하기 위해 중요한 구성 변경의 경우 카나리 배포와 같은 가중치 기반 배포를 고려하세요.
5. 구성 관리를 CI/CD 파이프라인에 통합합니다.
6. 푸시된 모든 변경 사항을 확인합니다.

리소스

관련 모범 사례:

- OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립
- OPS06-BP02 테스트 배포
- OPS06-BP03 안전한 배포 전략 채택
- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

- AWS Control Tower
- AWS Landing Zone Accelerator

- AWS Config
- What is AWS Config?
- AWS AppConfig
- What is AWS CloudFormation?
- AWS 개발자 도구
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy

관련 비디오:

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- Manage and Deploy Application Configurations with AWS AppConfig

OPS05-BP04 구축 및 배포 관리 시스템 사용

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

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

원하는 성과: 빌드 및 배포 관리 시스템은 올바른 구성으로 안전한 롤아웃을 자동화하는 기능을 제공하는 조직의 지속적 통합 및 지속적 전달(CI/CD) 시스템을 지원합니다.

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 구축하고 코드를 품질 보증(QA) 팀에 제공합니다. 정적 자산이 누락되어 QA에 실패합니다.
- 금요일에는 많은 노력을 기울이고 새로 코딩된 기능을 포함하여 개발 환경에서 수동으로 애플리케이션을 성공적으로 구축했습니다. 월요일에는 애플리케이션을 성공적으로 구축할 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.

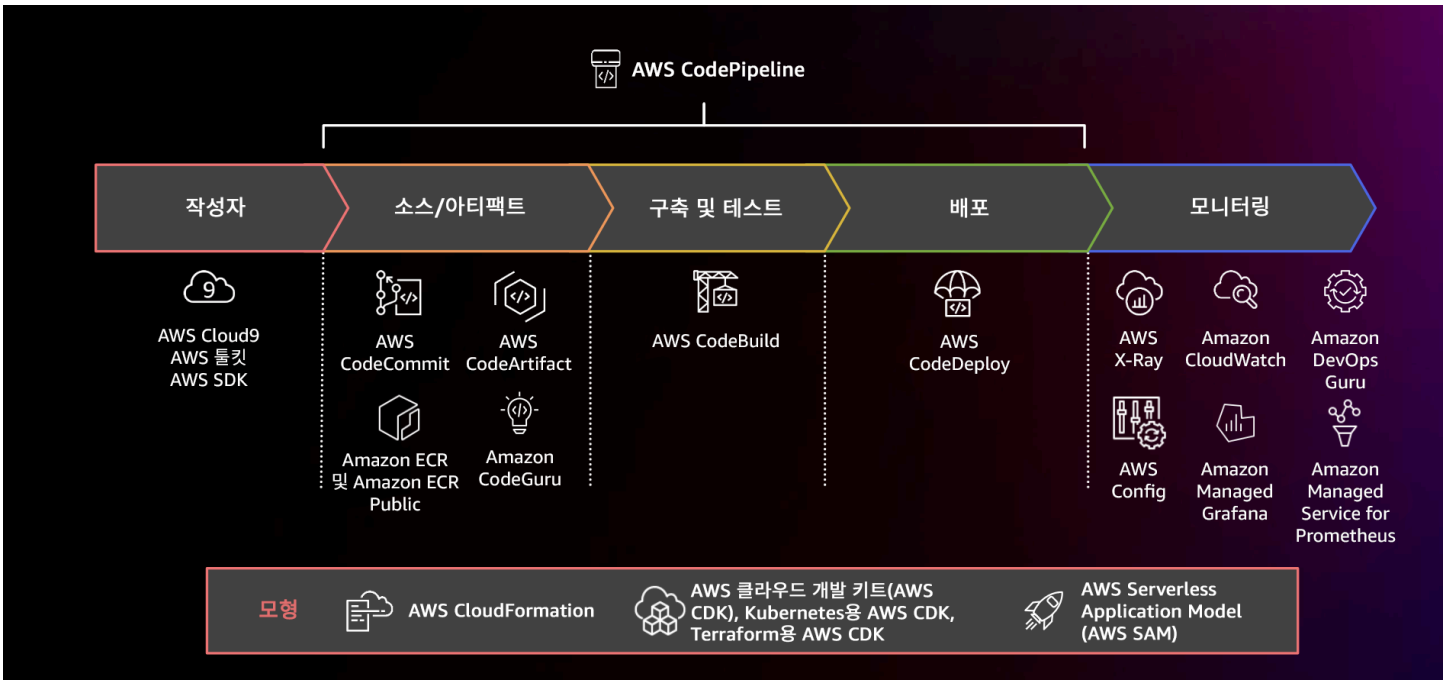
이 모범 사례 확립의 이점:빌드 및 배포 활동을 관리하는 메커니즘을 제공하여 반복적인 작업 수행을 위한 작업량을 줄이고, 팀원이 고가치 창조 작업에 집중할 수 있게 하며, 수동 절차에서 발생하는 오류의 도입을 제한할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

구축 및 배포 관리 시스템은 변경 사항을 추적 및 구현하고, 수동 프로세스로 인한 오류를 줄이며, 안전한 배포에 필요한 노력을 줄이는 데 사용됩니다. 코드 체크인에서 구축, 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 완전히 자동화합니다. 이를 통해 리드 타임, 비용 절감, 변경 빈도 증가, 작업량 감소, 협업 증대 등의 효과를 얻을 수 있습니다.

구현 단계



AWS CodePipeline과 관련 서비스를 사용하는 CI/CD 파이프라인을 보여주는 다이어그램

1. 자산(예: 문서, 소스 코드, 바이너리 파일)을 저장 및 관리하는 데 버전 관리를 사용합니다.
2. CodeBuild를 사용하여 소스 코드를 컴파일하고 유닛 테스트를 실행하며 배포 준비가 완료된 아티팩트를 생성합니다.

3. Amazon EC2 인스턴스, 온프레미스 인스턴스, 서버리스 AWS Lambda 함수 또는 Amazon ECS 서비스로 애플리케이션 배포를 자동화하는 배포 서비스로 CodeDeploy를 사용합니다.
4. 배포를 모니터링하세요.

리소스

관련 모범 사례:

- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

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

관련 비디오:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS05-BP05 패치 관리 수행

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

패치 및 취약성 관리는 이점 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.

AWS Health는 계획된 수명 주기 이벤트 및 AWS 클라우드 리소스 상태에 영향을 미치는 기타 조치가 필요한 이벤트에 대한 신뢰할 수 있는 정보 소스입니다. 수행해야 할 예정된 변경 사항 및 업데이트를 알고 있어야 합니다. 계획된 주요 수명 주기 이벤트는 최소 6개월 전에 전송됩니다.

Amazon EC2 Image Builder는 머신 이미지를 업데이트하기 위한 파이프라인을 제공합니다. 패치 관리의 일환으로 AMI 이미지 파이프라인을 사용하는 Amazon Machine Image(AMI) 또는 Docker 이미지 파

이프라인에서 컨테이너 이미지를 고려합니다. 한편, AWS Lambda에서는 취약성을 제거하기 위해 사용자 지정 런타임 및 추가 라이브러리에 대한 패턴을 제공합니다.

Amazon EC2 Image Builder를 사용하여 Linux 또는 Windows Server 이미지용 Amazon Machine Images에 대한 업데이트를 관리해야 합니다. 기존 파이프라인과 함께 Amazon Elastic Container Registry(Amazon ECR)를 사용하여 Amazon ECS 이미지 및 Amazon EKS 이미지를 관리할 수 있습니다. Lambda에는 버전 관리 기능이 포함되어 있습니다.

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

원하는 성과: AMI 및 컨테이너 이미지는 패치가 적용되고 최신 상태이며 시작할 준비가 되었습니다. 배포된 모든 이미지의 상태를 추적하고 패치 규정 준수 여부를 알 수 있습니다. 현재 상태를 보고하고 규정 준수 요구 사항을 충족하는 프로세스를 마련할 수 있습니다.

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 인스턴스에 상용 소프트웨어(기성품)를 패치하지 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.
- 긴급 수정이 필요한 제로데이 취약성에 대한 알림을 받게 되며 모든 환경을 수동으로 패치해야 합니다.
- 예정된 계획된 수명 주기 이벤트 및 기타 정보를 검토하지 않아 필수 버전 업데이트와 같이 리소스를 유지하는 데 필요한 중요한 조치를 알지 못합니다. 계획 및 실행에 중요한 시간을 놓쳐 팀의 긴급 변경과 잠재적 영향 또는 예상치 못한 가동 중지 시간이 발생합니다.

이 모범 사례 확립의 이점: 패치 적용 기준 및 환경 전체에 배포를 위한 방법론을 포함하여 패치 관리 프로세스를 설정하면 패치 수준을 조정하고 보고할 수 있습니다. 이를 통해 보안 패치를 보장하고 알려진 수정 사항의 상태를 명확하게 파악할 수 있습니다. 이를 통해 원하는 기능을 도입하고, 문제를 신속히 제거하며, 거버넌스를 지속적으로 준수할 수 있습니다. 패치 관리 시스템 및 자동화를 구현하여 패치 배포를 위한 작업량을 줄이고 수동 프로세스로 인한 오류를 제한합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

원하는 기능을 생성하고 거버넌스 정책과 공급업체 지원 요구 사항을 준수하는 상태를 유지할 수 있도록 시스템에 패치를 적용하여 문제를 해결합니다. 변경 불가능한 시스템에서는 원하는 성과를 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.

구현 단계

Amazon EC2 Image Builder의 경우:

1. Amazon EC2 Image Builder를 사용하여 파이프라인 세부 정보를 지정합니다.
   a. 이미지 파이프라인 생성 및 이름 지정
   b. 파이프라인 일정 및 시간대 정의
   c. 모든 종속성 구성
2. 레시피 선택:
   a. 기존 레시피 선택 또는 새 레시피 생성
   b. 이미지 유형 선택
   c. 레시피 이름 및 버전 지정
   d. 기본 이미지 선택
   e. 빌드 구성 요소 추가 및 대상 레지스트리에 추가
3. 선택 사항 - 인프라 구성을 정의합니다.
4. 선택 사항 - 구성 설정을 정의합니다.
5. 설정을 검토합니다.
6. 레시피 상태를 정기적으로 유지 관리합니다.

Systems Manager Patch Manager의 경우:

1. 패치 기준선을 생성합니다.

2. 패치 작업 방법을 선택합니다.
3. 규정 준수 보고 및 스캔을 활성화합니다.

리소스

관련 모범 사례:

- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

- What is Amazon EC2 Image Builder
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager
- Patch Manager 작업
- 패치 규정 준수 보고서 작업
- AWS 개발자 도구

관련 비디오:

- CI/CD for Serverless Applications on AWS
- Design with Ops in Mind

관련 예제:

- AWS Systems Manager Patch Manager 자습서

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

원하는 성과: 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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. 팀에 게시된 콘텐츠를 알립니다. 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 이해관계자와 협력하여 설계 표준, 운영 절차, 체크리스트, 지침 및 거버넌스 요구 사항을 개발합니다. 설계 표준을 문서화하고 조직 내에서 공유합니다.
   a. AWS Service Catalog를 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 표준이 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 Catalog Account Factory-Enhanced
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog
- Maintain visibility over the use of cloud architecture patterns
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup

관련 비디오:

- AWS Service Catalog – Getting Started
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert

관련 예제:

- AWS Service Catalog Reference Architecture
- 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 성과: 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 로깅되지 않습니다.

이 모범 사례 확립의 이점: 코드 품질 개선을 위한 사례를 도입하면 프로덕션에서 발생하는 문제를 최소화할 수 있습니다. 코드 품질 모범 사례에는 페어 프로그래밍, 코드 검토, AI 생산성 도구 구현이 포함됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

Amazon Q Developer를 통해 생성형 AI의 성능을 활용하여 개발자 생산성과 코드 품질을 개선합니다. Amazon Q Developer에는 코드 제안 생성(대규모 언어 모델 기반), 단위 테스트 생성(경계 조건 포함), 보안 취약성 탐지 및 해결을 통한 코드 보안 강화 기능이 있습니다.

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 기법을 사용하여 소프트웨어 품질을 개선합니다.
   a. 단위 테스트 사례(경계 조건 포함)를 생성하고, 코드 및 주석을 사용하여 함수를 생성하며, 잘 알려진 알고리즘을 구현하고, 코드에서 보안 정책 위반 및 취약성을 탐지하며, 보안 암호를 탐지하고, 코드형 인프라(IaC)를 스캔하며, 코드를 문서화하고 서드파티 코드 라이브러리를 보다 빠르게 학습하는 데 도움이 되는 생성형 AI 도구인 Amazon Q Developer를 사용합니다.
   b. Amazon CodeGuru Reviewer에서는 기계 학습을 사용하여 Java 및 Python 코드에 대한 프로그래밍 권장 사항을 제공할 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-BP02 변경 사항 테스트 및 확인
- OPS05-BP06 설계 표준 공유

관련 문서:

- 테스트 기반 개발 접근 방식 채택
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Agile Software Guide
- My CI/CD pipeline is my release captain

- Automate code reviews with Amazon CodeGuru Reviewer
- 테스트 기반 개발 접근 방식 채택
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming
- RENGA Inc. automates code reviews with Amazon CodeGuru
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

관련 비디오:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development

관련 서비스:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler

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.

원하는 성과: 규정 준수 및 거버넌스 요구 사항을 반영하는 여러 환경이 있습니다. 프로덕션 단계로 진행하는 동안 여러 환경을 통해 코드를 테스트하고 승격합니다.

1. 조직은 거버넌스, 제어, 계정 자동화, 네트워킹, 보안 및 운영 관찰성을 제공하는 랜딩 존을 구축하여 이를 수행합니다. 여러 환경을 사용하여 이러한 랜딩 존 기능을 관리합니다. 일반적인 예는 AWS IAM Identity Center 및 서비스 제어 정책(SCP)과 같은 정책이 포함된 AWS Control Tower 기반 랜

딩 존에 대한 변경 사항을 개발하고 테스트하기 위한 샌드박스 조직입니다. 이러한 모든 요소는 랜딩 존 내의 AWS 계정에 대한 액세스 및 작업에 상당한 영향을 미칠 수 있습니다.

2. 이러한 서비스 외에도 팀은 AWS 및 AWS 파트너가 게시한 솔루션 또는 조직 내에서 개발된 사용자 지정 솔루션으로 랜딩 존 기능을 확장합니다. AWS에서 게시한 솔루션의 예로는 Customizations for AWS Control Tower(CfCT) 및 AWS Control Tower Account Factory for Terraform(AFT)이 있습니다.
3. 조직은 프로덕션으로 가는 경로의 환경을 통해 랜딩 존에 대해 동일한 테스트, 코드 승격 및 정책 변경 원칙을 적용합니다. 이 전략은 애플리케이션 및 워크로드 팀에 안정적이고 안전한 랜딩 존 환경을 제공합니다.

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 사용자 측에서 사용 중단이 발생합니다.
- 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 사용할 수 없도록 해야 합니다.
- 멀티 테넌트 서비스를 운영 중이며 전용 환경에 대한 고객 요청을 지원할 수 없습니다.
- 항상 테스트하지는 않지만 테스트할 때는 프로덕션 환경에서 테스트합니다.
- 단일 환경의 단순성이 환경 내 변경 사항의 영향 범위보다 우선한다고 생각합니다.
- 주요 랜딩 존 기능을 업그레이드하지만 변경으로 인해 새 프로젝트 또는 기존 워크로드에 대한 계정 벤딩 기능이 저하됩니다.
- AWS 계정에 새 컨트롤을 적용하지만 변경 사항은 워크로드 팀이 AWS 계정 내에서 변경 사항을 배포하는 능력에 영향을 미칩니다.

이 모범 사례 확립의 이점: 여러 환경을 배포할 때 여러 개발자 또는 사용자 커뮤니티 간에 충돌을 일으키지 않고 여러 동시 개발, 테스트 및 프로덕션 환경을 지원할 수 있습니다. 랜딩 존과 같은 복잡한 기능의 경우 변경 위험을 크게 줄이고 개선 프로세스를 간소화하며 환경에 대한 중요한 업데이트 위험을 줄입니다. 랜딩 존을 사용하는 조직은 계정 구조, 거버넌스, 네트워크 및 보안 구성을 통해 AWS 환경의 다중 계정에서 자연스럽게 이점을 얻습니다. 시간이 지나면서 조직이 성장함에 따라 랜딩 존은 워크로드와 리소스를 보호하고 정리하는 방향으로 변화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

다중 환경을 사용하고 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 지원합니다. 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 환경에 상응하는 환경을 배포합니다.

플랫폼 엔지니어링, 네트워킹 및 보안 운영과 같은 팀은 종종 고유한 요구 사항을 사용하여 조직 수준에서 기능을 관리합니다. 계정 분리만으로는 실험, 개발 및 테스트를 위한 별도의 환경을 제공하고 유지하기에 충분하지 않습니다. 이러한 경우 별도의 AWS Organizations 인스턴스를 생성합니다.

리소스

관련 문서:

- Instance Scheduler on AWS
- AWS CloudFormation란 무엇입니까?
- Organizing Your AWS Environment Using Multiple Accounts - Multiple organizations - Test changes to your overall AWS environment
- AWS Control Tower 안내서

OPS05-BP09 되돌릴 수 있는 소규모 변경 자주 적용

되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 영향과 범위가 감소합니다. 변경 관리 시스템, 구성 관리 시스템, 구축 및 전송 시스템과 함께 사용할 경우 되돌릴 수 있는 빈번한 소규모 변경으로 인해 변경의 범위와 영향이 줄어듭니다. 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. 이때 변경 기간은 코어 서비스가 해제되었음을 의미합니다.
- 관리 시스템의 변경 내용을 추적하지 않고 데이터베이스 스키마를 변경하는 경우가 많습니다.
- 수동 내부 업데이트를 수행하고 기존 설치 및 구성을 덮어쓰며 명확한 롤백 계획이 없습니다.

이 모범 사례 확립의 이점: 작은 변경 사항을 자주 배포하여 개발 작업을 더 빠르게 진행할 수 있습니다. 변경 사항이 작으면 의도하지 않은 결과가 있는지 파악하기 훨씬 더 쉽고 되돌리기도 더 쉽습니다. 변경 사항을 되돌릴 수 있는 경우 복구가 간소화됨에 따라 변경 사항 구현의 위험이 줄어듭니다. 변경 프로세스를 수행할 때 위험이 줄어들고 변경 실패로 인한 영향도 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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. 또한 업무에 유용한 기능을 더 빠르게 제공할 수 있습니다.

리소스

관련 모범 사례:

- OPS05-BP03 구성 관리 시스템 사용
- OPS05-BP04 구축 및 배포 관리 시스템 사용
- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

- Implementing Microservices on AWS
- Microservices - Observability

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.

리소스 태그와 AWS Resource Groups를 사용하여 메타데이터를 적용하고 일관된 태그 지정 전략을 시행하면 리소스를 식별할 수 있습니다. 조직, 비용 회계, 액세스 제어에 대한 리소스에 태그를 지정하여 자동화된 운영 활동을 실행할 대상을 설정합니다.

원하는 성과: 개발자는 도구를 사용하여 코드를 제공하고 프로덕션으로 승격합니다. 개발자는 업데이트를 제공하기 위해 AWS Management Console에 로그인할 필요가 없습니다. 변경 및 구성에 대한 전체 감사 추적이 있어 거버넌스 및 규정 준수 요구 사항을 충족합니다. 프로세스는 반복 가능하며 팀 간에 표준화되어 있습니다. 개발자는 자유롭게 개발 및 코드 푸시에 집중할 수 있어 생산성이 향상됩니다.

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.
- 개발자는 AWS Management Console에 로그인하여 비표준 방법 및 시스템을 사용하여 새 개발 환경을 만듭니다.

이 모범 사례 확립의 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. 프로덕션으로 승격하면서 전달 속도를 높일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

빌드 및 배포 관리 시스템을 사용하면 변경 사항을 추적 및 구현하고, 수동 프로세스로 인해 발생하는 오류와 작업량을 줄일 수 있습니다. 코드 체크인에서 구축, 테스트, 배포 및 확인까지의 전체 통합 및 배포 파이프라인을 완전히 자동화합니다. 이를 통해 리드 타임을 줄이고, 변경 빈도를 높이며, 작업 수준을 줄이고, 시장 출시 속도를 높이며, 생산성을 높이고, 프로덕션으로 승격하면서 코드의 보안을 강화할 수 있습니다.

리소스

관련 모범 사례:

- OPS05-BP03 구성 관리 시스템 사용
- OPS05-BP04 구축 및 배포 관리 시스템 사용

관련 문서:

- AWS CodeBuild란 무엇입니까?
- AWS CodeDeploy란 무엇입니까?

관련 비디오:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS 6. 배포 위험을 어떻게 최소화하고 있나요?

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 테스트 배포
- OPS06-BP03 안전한 배포 전략 채택
- OPS06-BP04 테스트 및 롤백 자동화

OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립

배포로 인해 원치 않는 결과가 발생하는 경우 알려진 정상 상태로 되돌릴 수 있는 계획을 세우거나 프로덕션 환경에서 관련 문제를 해결합니다. 이러한 계획을 수립하기 위한 정책이 있으면 모든 팀이 변경 실패에서 복구하기 위한 전략을 개발할 수 있습니다. 전략의 예로는 배포 및 롤백 단계, 변경 정책, 기능 플래그, 트래픽 격리, 트래픽 이동 등이 있습니다. 단일 릴리스에는 관련된 구성 요소 변경 사항이 여러 개 포함될 수 있습니다. 이 전략을 통해 구성 요소 변경 실패를 견디거나 복구할 수 있어야 합니다.

원하는 성과: 변경이 제대로 되지 않은 경우에 대비하여 상세한 복구 계획을 준비했습니다. 또한 다른 워크로드 구성 요소에 미치는 잠재적 영향을 최소화하기 위해 릴리스 크기를 줄였습니다. 그 결과, 변경 실패로 인한 잠재적 가동 중지 시간을 줄이고 복구 시간의 유연성과 효율성을 높여 비즈니스에 미치는 영향을 줄였습니다.

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 있으므로 기다렸다가 롤백할 것인지를 결정해야 합니다.
- 루틴을 변경한 후에는 새 환경에 액세스할 수 있지만, 서브넷 중 하나에 연결할 수 없게 됩니다. 전부 롤백할지 아니면 액세스할 수 없는 서브넷을 수정할지 결정해야 합니다. 이러한 결정을 내리는 동안 서브넷에는 계속 연결할 수 없습니다.
- 시스템이 소규모 릴리스로 업데이트할 수 있는 방식으로 설계되지 않았습니다. 따라서 실패한 배포 중에 이러한 대량 변경 사항을 되돌리기가 어렵습니다.

- 코드형 인프라(IaC)를 사용하지 않으며 인프라가 수동으로 업데이트되어 원치 않는 구성을 초래했습니다. 수동 변경 사항을 효과적으로 추적하고 되돌릴 수 없습니다.
- 배포 빈도의 증가를 측정하지 않았으므로, 변경 사항의 크기를 줄이고 각 변경에 대한 롤백 계획을 개선하도록 팀을 장려하지 못하여 위험이 늘어나고 실패율이 증가합니다.
- 부적절한 변경으로 인한 운영 중단의 총 기간을 측정하지 않습니다. 팀이 배포 프로세스와 복구 계획 효율성의 우선순위를 지정하고 개선할 수 없습니다.

이 모범 사례 확립의 이점: 실패한 변경을 복구하기 위한 계획을 세우면 평균 복구 시간(MTTR)을 최소화하고 비즈니스에 미치는 영향을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

릴리스 팀에서 채택한 일관되고 문서화된 정책 및 관행을 통해 조직은 변경이 부적절한 경우 어떤 일이 발생할지 계획할 수 있습니다. 정책은 특정 상황에서 수정이 허용되어야 합니다. 어떤 상황에서든 변경 사항을 되돌리는 데 걸리는 시간을 최소화하려면 라이브 프로덕션에 배포하기 전에 수정 사항이나 롤백 계획을 올바르게 문서화하고 테스트해야 합니다.

구현 단계

1. 팀이 지정된 기간 내에 변경 사항을 되돌릴 수 있는 효과적인 계획을 수립하도록 요구하는 정책을 문서화합니다.
   a. 정책에는 수정 상황이 허용되는 시기가 명시되어야 합니다.
   b. 관련된 모든 사람이 액세스할 수 있도록 롤백 계획을 문서화해야 합니다.
   c. 롤백 요구 사항을 지정합니다(예: 무단 변경 사항이 배포된 것으로 확인된 경우).
2. 워크로드의 각 구성 요소와 관련된 모든 변경의 영향 수준을 분석합니다.
   a. 반복 가능한 변경 사항이 변경 정책을 적용하는 일관된 워크플로를 따르는 경우 표준화 및 템플릿화되고 사전 승인되도록 허용합니다.
   b. 복구에 들이는 시간을 줄이고 비즈니스에 미치는 영향을 줄일 수 있도록 변경 크기를 줄여 변경 사항의 잠재적 영향을 줄입니다.
   c. 가능한 경우 롤백 프로시저가 코드를 알려진 정상 상태로 되돌려 사고가 발생하지 않도록 합니다.
3. 도구와 워크플로를 통합하여 정책을 프로그래밍 방식으로 적용합니다.
4. 변경 사항에 대한 데이터를 다른 워크로드 책임자가 볼 수 있도록 하여 롤백할 수 없는 실패한 변경 사항의 진단 속도를 개선합니다.

   a. 가시적인 변경 데이터를 사용하여 이러한 관행의 성공을 측정하고 반복적인 개선 사항을 파악합니다.
5. 모니터링 도구를 사용하여 배포의 성공 또는 실패를 확인하여 롤백에 대한 의사 결정 속도를 높입니다.
6. 변경이 부적절한 경우 운영 중단 기간을 측정하여 복구 계획을 지속적으로 개선합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

- AWS Builders Library | 배포 중 롤백 안전 보장
- AWS 백서 | Change Management in the Cloud

관련 비디오:

- re:Invent 2019 | Amazon’s approach to high-availability deployment

OPS06-BP02 테스트 배포

프로덕션 환경에서와 동일한 배포 구성, 보안 제어, 단계 및 절차를 사용하여 사전 프로덕션에서 릴리스 절차를 테스트합니다. 파일, 구성 및 서비스 검사 등의 배포된 모든 단계가 예상대로 완료되었는지 확인합니다. 상태 확인과 같은 모니터링과 함께 기능, 통합 및 로드 테스트를 통해 모든 변경 사항을 추가로 테스트합니다. 이러한 테스트를 수행하면 배포 문제를 조기에 찾아내 프로덕션에 앞서 계획을 세우고 문제를 완화할 수 있습니다.

모든 변경을 테스트하기 위한 임시 병렬 환경을 만들 수 있습니다. 코드형 인프라(IaC)를 사용하여 테스트 환경 배포를 자동화하면 관련된 작업량을 줄이고 안정성, 일관성 및 더 빠른 기능 제공을 보장할 수 있습니다.

원하는 성과: 조직에 테스트 배포를 포함하는 테스트 기반 개발 문화를 도입합니다. 이를 통해 팀은 릴리스 관리보다는 비즈니스 가치 제공에 집중할 수 있습니다. 배포 위험이 식별되면 팀이 조기에 참여하여 적절한 완화 방법을 결정합니다.

일반적인 안티 패턴:

- 프로덕션 릴리스 중에 테스트되지 않은 배포로 인해 문제 해결과 에스컬레이션이 필요한 문제가 자주 발생합니다.
- 릴리스에 기존 리소스를 업데이트하는 코드형 인프라(IaC)가 포함되어 있습니다. IaC가 성공적으로 실행될지 또는 리소스에 영향을 미치게 될지 확실히 알 수 없습니다.
- 애플리케이션에 새로운 기능을 배포합니다. 애플리케이션이 의도한 대로 작동하지 않으며 영향을 받은 사용자가 신고하기 전까지는 가시성이 없습니다.
- 인증서를 업데이트합니다. 실수로 잘못된 구성 요소에 인증서를 설치하면 웹 사이트에 대한 보안 연결을 설정할 수 없기 때문에 이 문제가 감지되지 않고 웹 사이트 방문자에게 영향을 미칩니다.

이 모범 사례 확립의 이점: 배포 절차의 사전 프로덕션 단계에서 광범위한 테스트를 수행하고 이에 따라 변경 사항을 도입하면 배포 단계로 인해 프로덕션에 미치는 잠재적 영향을 최소화할 수 있습니다. 그러면 프로덕션 릴리스 시 신뢰도가 높아지고 제공되는 변경 사항의 속도를 늦추지 않고도 운영 지원을 최소화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

배포 프로세스를 테스트하는 것은 배포로 인한 변경 사항을 테스트하는 것만큼 중요합니다. 이렇게 하려면 프로덕션을 최대한 비슷하게 미러링하는 사전 프로덕션 환경에서 배포 단계를 테스트합니다. 불완전하거나 잘못된 배포 단계, 구성 오류 등과 같은 일반적인 문제는 프로덕션에 들어가기 전에 발견할 수 있습니다. 또한 복구 단계를 테스트할 수 있습니다.

고객 사례

AnyCompany Retail은 지속적 통합 및 지속적 전달(CI/CD) 파이프라인의 일환으로 프로덕션과 유사한 환경에서 고객을 위한 인프라 및 소프트웨어 업데이트를 릴리스하는 데 필요한 정의된 단계를 수행합니다. 이 파이프라인은 배포 전에 리소스의 드리프트를 감지(IaC 외부에서 수행된 리소스의 변경을 감지)하기 위한 사전 검사와 IaC 시작 시 취하는 작업에 대한 검증으로 구성됩니다. 로드 밸런서에 다시 등록하기 전에 특정 파일 및 구성이 제자리에 있고 서비스가 실행 상태이고 로컬 호스트의 상태 확인에 올바르게 응답하는지 확인하는 등, 배포 단계를 검증합니다. 또한 모든 변경 사항은 기능, 보안, 회귀, 통합 및 로드 테스트 등의 여러 자동 테스트에 플래그를 지정합니다.

구현 단계

1. 설치 전 검사를 수행하여 사전 프로덕션 환경을 프로덕션에 미러링합니다.

   a. 드리프트 감지를 사용하여 리소스가 AWS CloudFormation 외부에서 변경된 시점을 감지합니다.
   b. 변경 세트를 사용하여 스택 업데이트의 의도가 변경 세트가 시작될 때 AWS CloudFormation에서 수행하는 작업과 일치하는지 확인합니다.
2. 이렇게 하면 AWS CodePipeline에서 수동 승인 단계가 트리거되어 사전 프로덕션 환경에 대한 배포를 승인합니다.
3. AWS CodeDeploy AppSpec 파일과 같은 배포 구성을 사용하여 배포 및 검증 단계를 정의합니다.
4. 해당하는 경우 AWS CodeDeploy를 다른 AWS 서비스와 통합하거나 AWS CodeDeploy를 파트너 제품 및 서비스와 통합합니다.
5. Amazon CloudWatch, AWS CloudTrail, Amazon SNS 이벤트 알림을 사용하여 배포를 모니터링합니다.
6. 기능, 보안, 회귀, 통합 및 로드 테스트를 포함하여 배포 후 자동화된 테스트를 수행합니다.
7. 배포 관련 문제를 해결합니다.
8. 이전 단계에 대한 검증이 성공적으로 끝나면 프로덕션으로의 배포를 승인하는 수동 승인 워크플로가 시작됩니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- OPS05-BP02 변경 사항 테스트 및 확인

관련 문서:

- AWS Builders' Library | 안전한 자동 배포 자동화 | 테스트 배포
- AWS 백서 | AWS에서의 지속적 통합 및 지속적 전달 사례
- The Story of Apollo - Amazon's Deployment Engine
- How to test and debug AWS CodeDeploy locally before you ship your code
- Integrating Network Connectivity Testing with Infrastructure Deployment

관련 비디오:

- re:Invent 2020 | Testing software and systems at Amazon

관련 예제:

- Tutorial | Deploy and Amazon ECS service with a validation test

OPS06-BP03 안전한 배포 전략 채택

안전한 프로덕션 롤아웃은 유익한 변경 사항이 고객에게 미치는 영향을 최소화하기 위해 이러한 변경 사항의 흐름을 제어합니다. 안전 제어는 검사 메커니즘을 제공하여 원하는 성과를 검증하고 변경 사항 또는 배포 실패로 인한 결함의 영향 범위를 제한합니다. 안전한 롤아웃에는 기능 플래그, 원박스, 롤링 (canary 릴리스), 변경 불가, 트래픽 분할, 블루/그린 배포와 같은 전략이 포함될 수 있습니다.

원하는 성과: 조직이 안전한 롤아웃을 자동화하는 기능을 제공하는 지속적 통합 및 지속적 전달(CI/CD) 시스템을 사용합니다. 팀은 적절한 안전한 롤아웃 전략을 사용해야 합니다.

일반적인 안티 패턴:

- 실패한 변경 사항을 모든 프로덕션에 한 번에 배포합니다. 결과적으로 모든 고객이 동시에 영향을 받습니다.
- 모든 시스템에 동시에 배포할 때 결함이 발생하면 긴급 릴리스가 필요합니다. 모든 고객의 결함을 수정하려면 며칠이 걸립니다.
- 프로덕션 릴리스를 관리하려면 여러 팀이 계획을 수립하고 참여해야 합니다. 이로 인해 고객을 위해 기능을 자주 업데이트하는 데 제약이 따릅니다.
- 기존 시스템을 수정하여 변경 가능한 배포를 수행합니다. 변경이 적절하지 못했음을 발견한 후에는 이전 버전 복원을 위해 시스템을 다시 수정하여 복구 시간이 연장해야 합니다.

이 모범 사례 확립의 이점: 자동 배포는 고객에게 유익한 변경 사항을 일관되게 제공하는 것과 롤아웃 속도의 균형을 맞춥니다. 영향을 제한하면 비용이 많이 드는 배포 실패를 방지하고 팀이 실패에 효율적으로 대응할 수 있는 능력을 최대화할 수 있습니다.

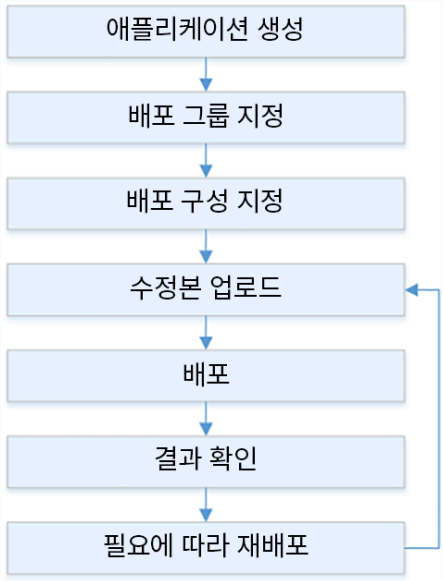
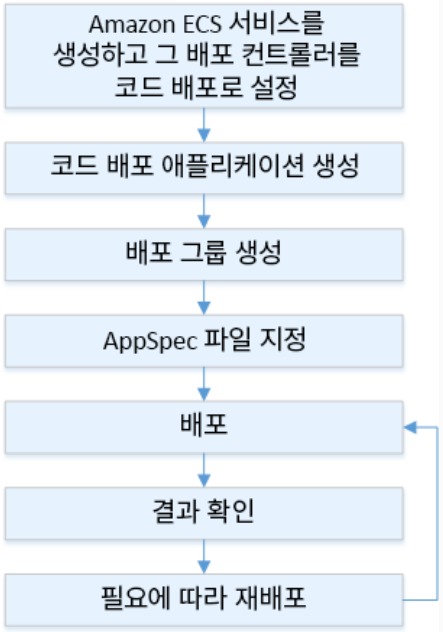
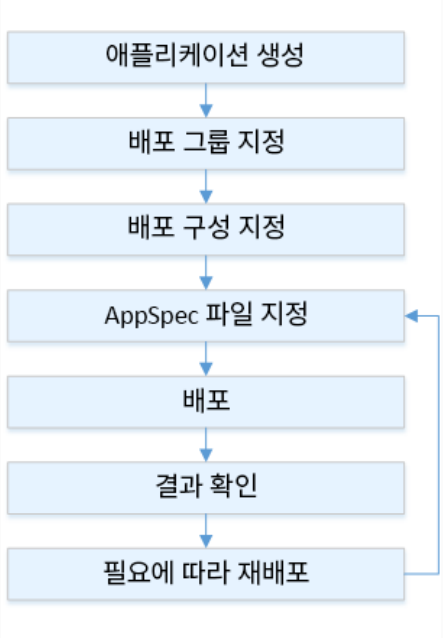
이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

지속적 전달 실패는 서비스 가용성 감소와 고객 불만족으로 이어질 수 있습니다. 배포 성공률을 최대화하려면 배포 실패 제로 달성을 목표로 엔드 투 엔드 릴리스 프로세스에 안전 제어를 구현하여 배포 오류를 최소화합니다.

고객 사례

AnyCompany Retail은 가동 중단이 거의 없거나 전혀 없는 배포를 목표로 삼고 있습니다. 배포 중에 사용자에게 미치는 영향이 전혀 없다는 의미입니다. 이를 위해 이 회사는 롤링 및 블루/그린 배포와 같은 배포 패턴(다음 워크플로 다이어그램 참조)을 확립했습니다. 모든 팀은 CI/CD 파이프라인에 이러한 패턴 중 하나 이상을 채택합니다.

Amazon EC2에 대한 CodeDeploy 워크플로	Amazon ECS에 대한 CodeDeploy 워크플로	Lambda에 대한 CodeDeploy 워크플로
		

구현 단계

1. 승인 워크플로를 사용하여 프로덕션 단계로 진입할 때 프로덕션 롤아웃 단계의 순서를 시작합니다.
2. AWS CodeDeploy과 같은 자동화된 배포 시스템을 사용합니다. AWS CodeDeploy 배포 옵션에는 EC2/온프레미스에 대한 인플레이스 배포와 EC2/온프레미스, AWS Lambda, Amazon ECS에 대한 블루/그린 배포를 포함합니다(이전 워크플로 다이어그램 참조).
   a. 해당하는 경우 AWS CodeDeploy를 다른 AWS 서비스와 통합하거나 AWS CodeDeploy를 파트너 제품 및 서비스와 통합합니다.
3. Amazon Aurora 및 Amazon RDS와 같은 데이터베이스에서는 블루/그린 배포를 사용합니다.
4. Amazon CloudWatch, AWS CloudTrail, Amazon Simple Notification Service(SNS) 이벤트 알림을 사용하여 배포를 모니터링합니다.
5. 기능, 보안, 회귀, 통합 및 로드 테스트를 비롯한 자동화된 배포 후 테스트를 수행합니다.

6. 배포 관련 문제를 해결합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS05-BP02 변경 사항 테스트 및 확인
- OPS05-BP09 되돌릴 수 있는 소규모 변경 자주 적용
- OPS05-BP10 통합 및 배포 완전 자동화

관련 문서:

- AWS Builders' Library | 안전한 자동 배포 자동화 | 프로덕션 배포
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS 백서 | AWS에서의 지속적 통합 및 지속적 전달 사례 | 배포 방법
- AWS CodeDeploy 사용 설명서
- Working with deployment configurations in AWS CodeDeploy
- API Gateway Canary 릴리스 배포 설정
- Amazon ECS Deployment Types
- Fully Managed Blue/Green Deployments in Amazon Aurora and Amazon RDS
- Blue/Green deployments with AWS Elastic Beanstalk

관련 비디오:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

관련 예제:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- 워크숍 | Building CI/CD pipelines for Lambda canary deployments using AWS CDK

- 워크숍 | Building your first DevOps Blue/Green pipeline with Amazon ECS
- 워크숍 | Building your first DevOps Blue/Green pipeline with Amazon EKS
- 워크숍 | EKS GitOps with ArgoCD
- 워크숍 | CI/CD on AWS Workshop
- Implementing cross-account CI/CD with AWS SAM for container-based Lambda functions

OPS06-BP04 테스트 및 롤백 자동화

배포 프로세스의 속도, 신뢰성 및 정확성을 높이려면 사전 프로덕션 및 프로덕션 환경에서 자동화된 테스트 및 롤백 기능을 위한 전략이 있어야 합니다. 프로덕션에 배포할 때 테스트를 자동화하여 배포되는 변경 사항을 확인하는 사람과 시스템의 상호 작용을 시뮬레이션합니다. 롤백을 자동화하면 이전에 알려진 정상 상태로 빠르게 되돌릴 수 있습니다. 롤백은 원하는 변경 결과를 얻지 못하거나 자동화된 테스트가 실패할 때와 같이 사전 정의된 조건에서 자동으로 시작되어야 합니다. 이 두 가지 활동을 자동화하면 배포 성공률이 향상되고 복구 시간이 최소화되며 비즈니스에 미치는 잠재적 영향이 줄어듭니다.

원하는 성과: 자동화된 테스트 및 롤백 전략이 지속적 통합 및 지속적 전달(CI/CD) 파이프라인에 통합됩니다. 모니터링은 성공 기준과 비교하여 검증하고 실패 시 자동 롤백을 시작할 수 있습니다. 이를 통해 최종 사용자와 고객에게 미치는 영향을 최소화할 수 있습니다. 예를 들어 모든 테스트 결과가 만족되면 동일한 테스트 사례를 활용하여 자동 회귀 테스트가 시작되는 프로덕션 환경으로 코드를 승격시킵니다. 회귀 테스트 결과가 기대치와 일치하지 않으면 파이프라인 워크플로에서 자동 롤백이 시작됩니다.

일반적인 안티 패턴:

- 시스템이 소규모 릴리스로 업데이트할 수 있는 방식으로 설계되지 않았습니다. 따라서 실패한 배포 중에 이러한 대량 변경 사항을 되돌리기가 어렵습니다.
- 배포 프로세스가 일련의 수동 단계로 구성되어 있습니다. 변경 사항을 워크로드에 배포한 후, 배포 후 테스트를 시작합니다. 테스트 후 워크로드가 작동하지 않으며 고객 연결이 끊어짐을 알게 됩니다. 그런 다음 이전 버전으로 롤백을 시작합니다. 이러한 모든 수동 단계는 전체 시스템 복구를 지연시키고 고객에게 장기적인 영향을 미칩니다.
- 애플리케이션에서 자주 사용되지 않는 기능에 대한 자동화된 테스트 사례를 개발하는 데 시간을 투자하여 자동화된 테스트 기능에 대한 투자 수익을 최소화했습니다.
- 릴리스가 서로 독립적인 애플리케이션, 인프라, 패치 및 구성 업데이트로 구성되어 있습니다. 하지만 모든 변경 사항을 한 번에 전달하는 단일 CI/CD 파이프라인이 있습니다. 한 구성 요소에 실패가 발생하면 모든 변경 사항을 되돌려야 하므로 롤백이 복잡하고 비효율적입니다.

- 팀이 스프린트 1에서 코딩 작업을 완료하고 스프린트 2 작업을 시작하지만 스프린트 3까지의 테스트는 계획에 포함되지 않았습니다. 그 결과, 자동화된 테스트를 통해 스프린트 2 산출물에 대한 테스트를 시작하기 전에 해결해야 했던 결함이 스프린트 1에서 드러났으며, 전체 릴리스가 지연되어 자동 테스트의 가치가 떨어졌습니다.
- 프로덕션 릴리스의 자동 회귀 테스트 사례는 완료되었지만 워크로드 상태를 모니터링하고 있지는 않습니다. 서비스 재시작 여부를 확인할 수 없으므로 롤백이 필요한지 또는 이미 발생했는지 확실하지 않습니다.

이 모범 사례 확립의 이점: 자동화된 테스트는 테스트 프로세스의 투명성을 높이고 더 짧은 기간에 더 많은 기능을 처리하는 능력을 향상시킵니다. 프로덕션에서 변경 사항을 테스트하고 검증하면 문제를 즉시 식별할 수 있습니다. 자동화된 테스트 도구를 사용하여 일관성을 개선하면 결함을 더 잘 감지할 수 있습니다. 이전 버전으로 자동 롤백하면 고객에게 미치는 영향이 최소화됩니다. 자동화된 롤백은 비즈니스에 대한 영향을 줄임으로써 궁극적으로 배포 기능에 대한 신뢰성을 높여줍니다. 전반적으로 이러한 기능은 품질을 보장하는 동시에 제공 시간을 단축합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

배포된 환경의 테스트를 자동화하여 원하는 성과를 더 빨리 확인합니다. 사전 정의된 결과를 달성할 수 없는 경우 이전의 알려진 정상 상태로 롤백하는 과정을 자동화하면 수동 프로세스에서 발생하는 오류를 줄이고 복구 시간을 최소화할 수 있습니다. 테스트 도구를 파이프라인 워크플로와 통합하여 지속적으로 테스트하고 수동 입력을 최소화합니다. 가장 큰 위험을 완화하고 변경 사항이 발생할 때마다 자주 테스트해야 하는 사례와 같이 테스트 사례를 자동화하는 것에 우선순위를 둡니다. 또한 테스트 계획에 사전 정의된 특정 조건에 따라 롤백을 자동화합니다.

구현 단계

1. 요구 사항 계획부터 테스트 사례 개발, 도구 구성, 자동화된 테스트, 테스트 사례 종료에 이르기까지 테스트 프로세스의 각 단계를 정의하는 개발 수명 주기에 대한 테스트 수명 주기를 설정합니다.
   a. 전체 테스트 전략을 바탕으로 워크로드별 테스트 접근 방식을 만듭니다.
   b. 개발 수명 주기 전반에 걸쳐 적절한 경우 지속적 테스트 전략을 고려합니다.
2. 비즈니스 요구 사항 및 파이프라인 투자를 기반으로 테스트 및 롤백을 위한 자동화된 도구를 선택합니다.
3. 자동화하려는 테스트 사례와 수동으로 수행할 테스트 사례를 결정합니다. 테스트 중인 기능의 비즈니스 가치 우선순위에 따라 테스트 사례를 지정할 수 있습니다. 모든 팀원을 이 계획에 맞춰 조정하고 수동 테스트를 수행할 책임을 확인합니다.

   a. 반복 가능하거나 자주 실행되는 사례, 반복 작업이 필요한 사례 또는 여러 구성에서 필요한 사례와 같이 자동화에 적합한 특정 테스트 사례에 자동화된 테스트 기능을 적용합니다.
   b. 특정 사례가 실패할 경우 지속적인 워크플로 자동화를 시작할 수 있도록 테스트 자동화 스크립트와 자동화 도구의 성공 기준을 정의합니다.
   c. 자동 롤백에 대한 구체적인 실패 기준을 정의합니다.
4. 테스트 자동화에 우선순위를 두고 복잡성과 인적 상호 작용의 실패 위험이 높은 철저한 테스트 사례 개발을 통해 일관된 결과를 도출합니다.
5. 자동화된 테스트 및 롤백 도구를 CI/CD 파이프라인에 통합합니다.
   a. 변경 사항에 대한 명확한 성공 기준을 개발합니다.
   b. 모니터링과 관찰을 통해 이러한 기준을 감지하고 특정 롤백 기준이 충족되면 변경 사항을 자동으로 되돌립니다.
6. 다음과 같은 다양한 유형의 자동 프로덕션 테스트를 수행합니다.
   a. A/B 테스트: 두 사용자 테스트 그룹 간의 결과를 현재 버전과 비교하여 보여줍니다.
   b. Canary 테스트: 모든 사용자에게 변경 사항을 릴리스하기 전에 일부 사용자에게 변경 사항을 롤아웃할 수 있습니다.
   c. 기능 플래그 테스트: 한 번에 새 버전의 단일 기능에 대해 애플리케이션 외부에서 플래그를 설정하거나 해제하여 새로운 기능을 한 번에 하나씩 검증할 수 있습니다.
   d. 회귀 테스트: 상관관계가 있는 기존 구성 요소를 사용하여 새로운 기능을 확인합니다.
7. 애플리케이션의 운영 측면, 트랜잭션, 다른 애플리케이션 및 구성 요소와의 상호 작용을 모니터링합니다. 워크로드별 변경 사항의 성공 여부를 보여주는 보고서를 개발하여 자동화 및 워크플로에서 추가로 최적화할 수 있는 부분을 파악할 수 있도록 합니다.
   a. 롤백 프로시저 간접 호출 여부를 신속하게 결정하는 데 도움이 되는 테스트 결과 보고서를 개발합니다.
   b. 하나 이상의 테스트 방법에서 나온 사전 정의된 실패 조건을 기반으로 자동 롤백을 허용하는 전략을 구현합니다.
8. 향후 반복 가능한 변경 사항에서 재사용할 수 있도록 자동화된 테스트 사례를 개발합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립

- OPS06-BP02 테스트 배포

관련 문서:

- AWS Builders Library | 배포 중 롤백 안전 보장
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

관련 예제:

- Serverless UI testing using Selenium, AWS Lambda, AWS Fargate, and AWS Developer Tools

관련 비디오:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

OPS 7. 귀사가 워크로드를 지원할 준비가 되어있는지 어떻게 알 수 있나요?

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 생성

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 지식

을 제공합니다. 워크로드의 정상 작동을 지원하고 발생하는 인시던트 문제를 해결할 수 있도록 충분한 교육을 받은 직원이 있어야 합니다. 번아웃을 방지하기 위해 당직 및 휴가 기간에 다른 인력이 교대될 수 있도록 충분한 인원을 확보합니다.

원하는 성과:

- 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.
- 당직 교대 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 당직 교대 근무를 지원할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드를 지원할 숙련된 인력이 충분히 있는지 검증합니다. 당직 교대 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. 당직 교대 근무를 지원할 엔지니어가 충분히 있습니다. 직원은 워크로드가 구축된 소프트웨어 및 플랫폼에 대한 교육을 받으며, 인증을 획득하도록 권장됩니다. 워크로드와 당직 교대 근무를 계속 지원하면서 휴가를 낼 수 있을 정도로 인력이 충분합니다.

구현 단계

1. 당직 교대 근무, 보안 문제, 수명 주기 이벤트(예: 지원 종료 및 인증서 교체 작업)를 비롯하여 워크로드를 운영하고 지원하기에 적절한 수의 인력을 할당합니다.
2. 워크로드를 구성하는 소프트웨어 및 플랫폼에 대해 직원을 교육합니다.

   a. AWS 교육 및 자격증에는 AWS에 대한 교육 과정 라이브러리가 있습니다. 무료 및 유료의 온라인, 오프라인 과정을 제공합니다.
   b. AWS 호스트 이벤트 및 웨비나에서는 AWS 전문가의 이야기를 들을 수 있습니다.
3. 정기적으로 다음을 수행합니다.
   - 운영 조건 및 워크로드 변화에 따라 팀 규모와 기술을 평가합니다.
   - 운영 요구 사항에 맞게 팀 규모와 기술을 조정합니다.
   - AWS Health를 통해 계획된 수명 주기 이벤트, 계획되지 않은 보안 및 운영 알림을 해결할 수 있는 기능과 용량을 확인합니다.

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 - 팀원은 워크로드를 운영하고 지원하는 데 필요한 정보를 가지고 있어야 합니다. 지식 관리는 이를 제공하는 열쇠입니다.

관련 문서:

- AWS 이벤트 및 웨비나
- AWS 교육 및 자격증

OPS07-BP02 일관된 방식으로 운영 준비 상태 검토

운영 준비 상태 검토(ORR)를 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 학습한 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 학습한 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 성과: 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.
- 학습한 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 위험 수준: 높음

구현 지침

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, 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 학습한 교훈을 통해 구성된 체크리스트를 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 Config를 사용하여 일부 모범 사례를 탐지해 ORR 프로세스의 속도를 높였습니다.

구현 단계

ORR에 대한 자세한 내용은 Operational Readiness Reviews(ORR) 백서를 참조하세요. 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 이해관계자를 한 자리에 모읍니다.
2. 각 이해관계자가 한 가지 이상의 요구 사항을 제공하도록 합니다. 첫 반복의 경우 항목의 수를 30개 이하로 제한합니다.
   - Operational Readiness Reviews(ORR) 백서의 Appendix B: Example ORR questions에는 시작 시 사용 가능한 샘플 질문이 포함되어 있습니다.
3. 요구 사항을 스프레드시트에 수집합니다.
   - AWS Well-Architected Tool에서 사용자 지정 렌즈를 사용하여 ORR을 개발하고 이를 계정 및 AWS 조직 전체에서 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 적절할 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 지원 고객은 기술 계정 관리자에게 운영 준비 상태 검토 워크숍을 요청할 수 있습니다. 이 워크숍은 자체 ORR 체크리스트를 개발하기 위한 대화형 역방향 작업 세션입니다.

구현 계획의 작업 수준: 높음. 조직에서 ORR 사례를 도입하려면 경영진의 후원과 이해관계자의 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 팀에 적절한 리소스 제공 – 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 바람직한 ORR 요구 사항입니다.
- COST02-BP01 조직 요구 사항에 따라 정책 개발 - 비용 관리 정책은 ORR 체크리스트에 포함하는 것이 좋습니다.

관련 문서:

- AWS Control Tower - Guardrails in AWS Control Tower
- AWS Well-Architected Tool - Custom Lenses
- Operational Readiness Review Template by Adrian Hornsby
- Operational Readiness Reviews (ORR) 백서

관련 비디오:

- AWS Supports You | Building an Effective Operational Readiness Review (ORR)

관련 예제:

- Sample Operational Readiness Review (ORR) Lens

관련 서비스:

- AWS Config
- AWS Control Tower
- AWS Security Hub
- AWS Well-Architected Tool

OPS07-BP03 런북을 사용한 절차 수행

런북은 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 성과를 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 어떤 방식으로든 자동화되어야 합니다.

원하는 성과: 팀에 워크로드 작업을 수행하기 위한 단계별 가이드 모음이 있습니다. 런북에는 원하는 성과, 필요한 도구, 권한 및 오류 처리 지침이 들어 있습니다. 런북이 중앙 위치(버전 관리 시스템)에 저장되고 자주 업데이트됩니다. 예를 들어, 런북을 통해 팀은 애플리케이션 경보, 운영 문제 및 계획된 수명 주기 이벤트 중에 중요한 계정에 대한 AWS Health 이벤트를 모니터링하고, 전달하고, 이에 대응할 수 있습니다.

일반적인 안티 패턴:

- 프로세스의 각 단계를 완료하기 위해 기억에 의존합니다.
- 체크리스트 없이 변경 사항을 수동으로 배포합니다.
- 동일한 프로세스를 팀원 여러 명이 수행하지만 사용하는 단계와 결과가 다릅니다.
- 런북이 시스템 변경 사항 및 자동화와 동기화되지 않도록 둡니다.

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

런북은 조직의 성숙도에 따라 여러 가지 형태일 수 있습니다. 최소한 단계별 텍스트 문서로 구성되어야 합니다. 원하는 성과가 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 자동화되어야 합니다. AWS Systems Manager Automation과 같은 서비스를 사용하여 일반 텍스트를 워크로드에 대해 실행할 수 있는 자동화로 변환할 수 있습니다. 이러한 자동화를 이벤트에 대응하여 실행해 워크로드 유지를 위한 운영 부담을 줄일 수 있습니다. AWS Systems Manager Automation은 자동화 런북을 보다 쉽게 생성할 수 있는 로우코드 시각적 디자인 경험도 제공합니다.

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 관리 리포지토리에서 런북 라이브러리 빌드를 시작하는 것이 좋습니다. 런북은 마크다운을 사용하여 빌드할 수 있습니다. 런북 빌드를 시작하는 데 사용할 수 있는 런북 템플릿 예제가 제공되어 있습니다.

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
```

```
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

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 마크다운 문서 초안을 작성합니다. 런북 제목 및 런북 정보 아래의 필수 필드를 입력합니다.
4. 첫 번째 단계부터 시작하여 런북의 단계 부분을 채웁니다.
5. 팀원에게 런북을 제공합니다. 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 빌드합니다. 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
- OPS07-BP04 플레이북을 사용하여 문제 조사
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용
- OPS10-BP02 알림별 프로세스 마련
- OPS11-BP04 지식 관리 수행

관련 문서:

- AWS Well-Architected Framework: 개념: 런북 개발
- Achieving Operational Excellence using automated playbook and runbook

- AWS Systems Manager: 런북 작업
- Migration playbook for AWS large migrations - Task 4: Improving your migration runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

관련 비디오:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

관련 예제:

- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- AWS 블로그 게시물: Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Systems Manager: 자동화 시연
- AWS Systems Manager: 최신 스냅샷에서 루트 볼륨 복원
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab - Runbooks
- Rubix - A Python library for building runbooks in Jupyter Notebooks
- 문서 빌더를 사용하여 사용자 지정 런북 생성

관련 서비스:

- AWS Systems Manager Automation

OPS07-BP04 플레이북을 사용하여 문제 조사

플레이북은 인시던트를 조사하는 데 사용하는 단계별 지침입니다. 인시던트가 발생하면 플레이북을 사용하여 조사하고, 영향의 범위를 살펴보며, 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 있습니다. 이를 통해 사용자에게 탐색 프로세스를 단계별로 안내합니다. 외부 관점에서 생각할 때, 인시던트를 진단하기 위해 어떤 단계를 따라야 할까요? 플레이

북에 특수 도구나 승격된 권한이 필요한 경우 플레이북에서 이를 명확하게 정의합니다. 이해관계자에게 조사 상황을 알리기 위한 커뮤니케이션 계획을 수립하는 것이 중요합니다. 근본 원인을 파악할 수 없는 경우에 대비한 에스컬레이션 계획도 있어야 합니다. 근본 원인이 파악되었다면 플레이북을 통해 해결 방법이 설명된 런북을 알 수 있어야 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 성과: 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 프로세스 규모를 조정할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

텍스트 플레이북은 조직이 성숙해지면 자동화되어야 합니다. AWS Systems Manager Automation과 같은 서비스를 사용하여 일반 텍스트를 자동화로 변환할 수 있습니다. 이러한 자동화를 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 Manager를 사용하여 인시던트에 대응할 수 있습니다. 이 서비스는 인시던트를 분류하고, 복구 및 완화 과정에서 이해관계자에게 이를 알리며, 인시던트 전반에서 협업할 수 있는 단일 인터페이스를 제공합니다. AWS Systems Manager Automation을 사용하여 탐지 및 복구 속도를 높입니다.

고객 사례

AnyCompany Retail에 생산 인시던트가 발생했습니다. 당직 근무 중인 엔지니어가 플레이북을 사용하여 문제를 조사했습니다. 단계에 따라 진행하면서 플레이북에서 파악한 주요 이해관계자에게 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 관리 리포지토리를 생성하는 것이 좋습니다. 플레이북은 대부분의 플레이북 자동화 시스템과 호환되는 마크다운을 사용하여 구축할 수 있습니다. 처음부터 시작하는 경우 다음 예제 플레이북 템플릿을 사용합니다.

```
# Playbook Title
## Playbook Info
| Playbook ID | Description | Tools Used | Special Permissions | Playbook Author | Last
 Updated | Escalation POC | Stakeholders | Communication Plan |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this playbook for? What incident is it used for? | Tools |
 Permissions | Your Name | 2022-09-21 | Escalation Name | Stakeholder Name | How will
 updates be communicated during the investigation? |
```

```
## Steps
1. Step one
2. Step two
```

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. 마크다운 템플릿을 사용하여 플레이북 이름 섹션과 플레이북 정보 아래의 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 이해관계자에게 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 Automation과 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
- OPS07-BP03 런북을 사용한 절차 수행
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용
- OPS10-BP02 알림별 프로세스 마련
- OPS11-BP04 지식 관리 수행

관련 문서:

- AWS Well-Architected Framework: 개념: 플레이북 개발
- Achieving Operational Excellence using automated playbook and runbook

- AWS Systems Manager: 런북 작업
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

관련 비디오:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- Integrate Scripts into AWS Systems Manager

관련 예제:

- AWS Customer Playbook Framework
- AWS Systems Manager: 자동화 시연
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – A Python library for building runbooks in Jupyter Notebooks
- 문서 빌더를 사용하여 사용자 지정 런북 생성

관련 서비스:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정

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

원하는 성과:

- 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 변경 실패에 대비합니다.
- 워크로드에 대한 변경 사항은 거버넌스 정책을 준수합니다.

이 모범 사례가 확립되지 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 시작하는 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 이해관계자의 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 테스트 배포 - 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 Shared Responsibility Model
- Governance in the AWS 클라우드: The Right Balance Between Agility and Safety

OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 생성

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 성과:

- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

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 활성화할 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 활성화했습니다. 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, 해당 수준의 구독것을 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. 또한 AWS Business Support는 AWS Management Console 및 Amazon EventBridge 채널 등의 다른 액세스 방법과 함께 시스템과의 프로그래밍 방식 통합을 위한 AWS Trusted Advisor 및 AWS Health에 대한 API 액세스를 비롯한 추가 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 Plans

관련 서비스:

- AWS Business Support
- AWS Enterprise Support

운영

Questions

- OPS 8. 조직에서 워크로드 관찰성을 어떻게 활용하고 있나요?
- OPS 9. 운영 상태를 어떻게 파악하나요?

- OPS 10. 워크로드 및 운영 이벤트를 어떻게 관리하나요?

OPS 8. 조직에서 워크로드 관찰성을 어떻게 활용하고 있나요?

관찰성을 활용하여 워크로드 상태를 최적화합니다. 관련 지표, 로그, 추적을 활용하여 워크로드 성능을 종합적으로 파악하고 문제를 효율적으로 해결합니다.

모범 사례

- OPS08-BP01 워크로드 지표 분석
- OPS08-BP02 워크로드 로그 분석
- OPS08-BP03 워크로드 추적 데이터 분석
- OPS08-BP04 실행 가능한 알림 생성
- OPS08-BP05 대시보드 만들기

OPS08-BP01 워크로드 지표 분석

애플리케이션 원격 측정을 구현한 후 수집된 지표를 정기적으로 분석합니다. 지연 시간, 요청, 오류, 용량(또는 할당량)은 시스템 성능에 대한 인사이트를 제공하지만 비즈니스 성과 지표 검토의 우선순위를 정하는 것이 중요합니다. 이를 통해 비즈니스 목표에 부합하는 데이터 기반 의사 결정을 내릴 수 있습니다.

원하는 성과: 워크로드 성능에 대한 정확한 인사이트를 통해 데이터에 기반한 의사 결정을 내리고 비즈니스 목표에 부합하도록 합니다.

일반적인 안티 패턴:

- 지표가 비즈니스 성과에 미치는 영향을 고려하지 않고 개별적으로 지표를 분석합니다.
- 기술 지표에 지나치게 의존하고 비즈니스 지표는 배제합니다.
- 지표를 자주 검토하지 않아 실시간 의사 결정 기회를 놓치고 있습니다.

이 모범 사례 확립의 이점:

- 기술 성과와 비즈니스 성과 간의 상관관계에 대해 더 잘 이해합니다.
- 실시간 데이터를 기반으로 의사 결정 프로세스를 개선합니다.
- 비즈니스 성과에 영향을 미치기 전에 문제를 사전에 식별하고 완화합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Amazon CloudWatch와 같은 도구를 활용하여 지표 분석을 수행합니다. CloudWatch 이상 탐지 및 Amazon DevOps Guru와 같은 AWS 서비스는 특히 정적 임곗값을 알 수 없거나 행동 패턴이 이상 탐지에 더 적합한 경우 이상을 탐지하는 데 사용할 수 있습니다.

구현 단계

1. 분석 및 검토: 워크로드 지표를 정기적으로 검토하고 해석하세요.
   a. 순전히 기술적인 지표보다 비즈니스 성과 지표를 우선시하세요.
   b. 데이터의 급증, 하락 또는 패턴의 중요성을 이해하세요.
2. Amazon CloudWatch 활용: 중앙 집중식 보기 및 심층 분석을 위해 Amazon CloudWatch를 사용합니다.
   a. 지표를 시각화하고 시간 경과에 따라 비교하도록 CloudWatch 대시보드를 구성합니다.
   b. CloudWatch에서 백분위수를 사용하여 지표 분포를 명확하게 파악하면 SLA를 정의하고 이상치를 이해하는 데 도움이 될 수 있습니다.
   c. CloudWatch 이상 탐지를 설정하여 정적 임곗값에 의존하지 않고 비정상적 패턴을 식별합니다.
   d. CloudWatch 크로스 계정 관찰성을 구현하여 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.
   e. CloudWatch Metric Insights를 사용하여 계정 및 리전 전반의 지표 데이터를 쿼리하고 분석해 추세와 이상 현상을 식별합니다.
   f. CloudWatch 지표 수식을 적용하여 지표를 변환, 집계 또는 계산을 수행해 심층적인 인사이트를 확보할 수 있습니다.
3. Amazon DevOps Guru 사용: 서버리스 애플리케이션의 운영 문제에 관한 초기 징후를 식별하고 고객에게 영향을 미치기 전에 문제를 해결할 수 있도록 기계 학습에 기반한 향상된 이상 탐지를 위해 Amazon DevOps Guru를 통합합니다.
4. 인사이트 기반 최적화: 지표 분석을 기반으로 정보에 입각한 결정을 내려 워크로드를 조정하고 개선하세요.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현

관련 문서:

- The Wheel 블로그 - Emphasizing the importance of continually reviewing metrics
- Percentile are important
- AWS Cost Anomaly Detection 사용
- CloudWatch 크로스 계정 관찰성
- CloudWatch Metrics Insights를 사용하는 지표 쿼리

관련 비디오:

- Enable Cross-Account Observability in Amazon CloudWatch
- Introduction to Amazon DevOps Guru
- Continuously Analyze Metrics using AWS Cost Anomaly Detection

관련 예제:

- One Observability 워크숍
- Gaining operation insights with AIOps using Amazon DevOps Guru

OPS08-BP02 워크로드 로그 분석

워크로드 로그를 정기적으로 분석하는 것은 애플리케이션의 운영 측면을 더 깊이 이해하는 데 필수적입니다. 로그 데이터를 효율적으로 선별, 시각화 및 해석함으로써 애플리케이션 성능과 보안을 지속적으로 최적화할 수 있습니다.

원하는 성과: 철저한 로그 분석을 통해 애플리케이션 동작 및 운영에 대한 풍부한 인사이트를 얻어 사전 예방적 문제 감지 및 완화를 보장합니다.

일반적인 안티 패턴:

- 심각한 문제가 발생할 때까지 로그 분석을 무시합니다.
- 로그 분석에 사용할 수 있는 모든 도구를 사용하지 않아 중요한 인사이트를 놓칩니다.
- 자동화 및 쿼리 기능을 활용하지 않고 수동 로그 검토에만 의존합니다.

이 모범 사례 확립의 이점:

- 운영 병목 현상, 보안 위협 및 기타 잠재적 문제를 사전에 식별합니다.
- 지속적인 애플리케이션 최적화를 위해 로그 데이터를 효율적으로 활용합니다.
- 애플리케이션 동작에 대한 이해도를 높여 디버깅 및 문제 해결을 지원합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Amazon CloudWatch Logs는 로그 분석을 위한 강력한 도구입니다. CloudWatch 로그 인사이트 및 Contributor Insights와 같은 통합 기능을 사용하면 로그에서 의미 있는 정보를 직관적이고 효율적으로 도출할 수 있습니다.

구현 단계

1. CloudWatch Logs 설정: CloudWatch Logs에 로그를 전송하도록 애플리케이션 및 서비스를 구성합니다.
2. 로그 이상 탐지 사용: Amazon CloudWatch Logs 이상 탐지 기능을 활용하여 비정상적인 로그 패턴을 자동으로 식별하고 이에 대해 알립니다. 이 도구를 사용하면 로그의 이상 현상을 사전에 관리하고 잠재적 문제를 조기에 발견할 수 있습니다.
3. CloudWatch 로그 인사이트 설정: CloudWatch 로그 인사이트를 사용하여 로그 데이터를 대화식으로 검색하고 분석합니다.
   a. 쿼리를 만들어 패턴을 추출하고, 로그 데이터를 시각화하며, 실행 가능한 인사이트를 도출합니다.
   b. CloudWatch 로그 인사이트 패턴 분석을 사용하여 빈번한 로그 패턴을 분석하고 시각화합니다. 이 기능은 로그 데이터의 일반적인 운영 추세와 잠재적 이상값을 이해하는 데 도움이 됩니다.
   c. CloudWatch Logs 비교(diff)를 사용하여 서로 다른 기간 간 또는 여러 로그 그룹 간의 차이 분석을 수행합니다. 이 기능을 사용하여 변경 사항을 정확히 찾아내고 시스템 성능 또는 동작에 미치는 영향을 평가할 수 있습니다.
4. Live Tail을 통한 실시간 로그 모니터링: Amazon CloudWatch Logs Live Tail을 사용하여 로그 데이터를 실시간으로 확인합니다. 애플리케이션의 운영 활동이 발생할 때 이를 적극적으로 모니터링할 수 있으므로 시스템 성능 및 잠재적 문제를 즉시 파악할 수 있습니다.
5. Contributor Insights 활용: CloudWatch Contributor Insights를 사용하여 IP 주소 또는 사용자 에이전트와 같은 높은 카디널리티 차원에서 볼륨이 높은 항목을 식별합니다.

6. CloudWatch Logs 지표 필터 구현: CloudWatch Logs 지표 필터를 구성하여 로그 데이터를 실행 가능한 지표로 변환합니다. 이를 통해 경보를 설정하거나 패턴을 추가로 분석할 수 있습니다.
7. CloudWatch 크로스 계정 관찰성 구현: 한 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.
8. 정기적 검토 및 개선: 정기적으로 로그 분석 전략을 검토하여 모든 관련 정보를 캡처하고 애플리케이션 성능을 지속적으로 최적화합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS08-BP01 워크로드 지표 분석

관련 문서:

- Analyzing Log Data with CloudWatch Logs Insights
- Using CloudWatch Contributor Insights
- Creating and Managing CloudWatch Log Metric Filters

관련 비디오:

- Analyze Log Data with CloudWatch Logs Insights
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

관련 예제:

- CloudWatch Logs Sample Queries
- One Observability 워크숍

OPS08-BP03 워크로드 추적 데이터 분석

추적 데이터를 분석하는 것은 애플리케이션의 운영 여정을 포괄적으로 파악하는 데 매우 중요합니다. 다양한 구성 요소 간의 상호 작용을 시각화하고 이해함으로써 성능을 미세 조정하고 병목 현상을 식별하며 사용자 경험을 개선할 수 있습니다.

원하는 성과: 애플리케이션의 분산 운영에 대한 명확한 가시성을 확보하여 더 빠른 문제 해결과 향상된 사용자 경험을 제공합니다.

일반적인 안티 패턴:

- 로그와 지표에만 의존하고 추적 데이터는 간과합니다.
- 추적 데이터를 관련 로그와 연관시키지 않습니다.
- 지연 시간 및 장애율과 같은 추적에서 도출된 지표를 무시합니다.

이 모범 사례 확립의 이점:

- 문제 해결을 개선하고 평균 문제 해결 시간(MTTR)을 줄입니다.
- 종속성과 그 영향에 대한 인사이트를 얻습니다.
- 성능 문제를 신속하게 식별하고 수정합니다.
- 정보에 입각한 의사 결정을 위해 추적에서 도출된 지표를 활용합니다.
- 최적화된 구성 요소 상호 작용을 통해 사용자 경험을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS X-Ray에서는 추적 데이터 분석을 위한 포괄적인 제품군을 제공하여 서비스 상호 작용에 대한 전체적인 보기를 제공하고, 사용자 활동을 모니터링하며, 성능 문제를 감지합니다. ServiceLens, X-Ray Insights, X-Ray Analytics, Amazon DevOps Guru와 같은 기능은 추적 데이터에서 더 심도 깊은 실행 가능한 인사이트를 얻을 수 있습니다.

구현 단계

아래 단계는 AWS 서비스를 사용하여 추적 데이터 분석을 효과적으로 구현하기 위한 체계적인 접근 방식을 제공합니다.

1. AWS X-Ray 통합: 애플리케이션과 X-Ray가 통합되어 추적 데이터를 캡처할 수 있도록 보장합니다.

2. X-Ray 지표 분석: 서비스 맵으로 애플리케이션 상태를 모니터링하여 지연 시간, 요청률, 장애율, 응답 시간 분포와 같은 X-Ray 추적에서 파생된 지표를 자세히 살펴봅니다.
3. ServiceLens 사용: ServiceLens 맵을 활용하여 서비스 및 애플리케이션의 관찰성을 개선합니다. 이를 통해 추적, 지표, 로그, 경보 및 기타 건강 정보를 통합적으로 볼 수 있습니다.
4. X-Ray Insights 활성화:
   a. 추적에서 자동 이상 탐지를 위해 X-Ray Insights를 켭니다.
   b. 인사이트를 검토하여 패턴을 정확히 찾아내고 장애율 또는 지연 시간 증가와 같은 근본 원인을 파악합니다.
   c. 인사이트 타임라인을 참조하여 감지된 문제를 시간순으로 분석합니다.
5. X-Ray Analytics 사용: X-Ray Analytics를 사용하면 추적 데이터를 철저히 탐색하고, 패턴을 정확히 찾아내며, 인사이트를 추출할 수 있습니다.
6. X-Ray에서 그룹 사용: X-Ray에서 그룹을 생성하여 높은 지연 시간과 같은 기준에 따라 추적을 필터링하여 보다 표적화된 분석을 수행할 수 있습니다.
7. Amazon DevOps Guru 통합: Amazon DevOps Guru를 통해 추적에서 운영 이상을 찾아내는 기계 학습 모델의 이점을 활용합니다.
8. CloudWatch Synthetics 사용: CloudWatch Synthetics를 사용하여 엔드포인트 및 워크플로를 지속적으로 모니터링하기 위한 canary를 생성합니다. 이러한 canary를 X-Ray와 통합하여 테스트 대상 애플리케이션의 심층 분석을 위한 추적 데이터를 제공할 수 있습니다.
9. 실제 사용자 모니터링(RUM) 사용: AWS X-Ray 및 CloudWatch RUM을 사용하면 애플리케이션의 최종 사용자부터 시작하여 다운스트림 AWS 관리형 서비스까지 요청 경로를 분석하고 디버그할 수 있습니다. 이를 통해 최종 사용자에게 영향을 미치는 지연 시간 추세와 오류를 파악할 수 있습니다.
10. 로그와의 상관관계 파악: 애플리케이션 동작을 세부적으로 파악할 수 있도록 X-Ray 추적 보기 내에서 추적 데이터와 관련 로그의 상관관계를 분석합니다. 이렇게 하면 추적된 트랜잭션과 직접 관련된 로그 이벤트를 볼 수 있습니다.
11. CloudWatch 크로스 계정 관찰성 구현: 한 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS08-BP01 워크로드 지표 분석
- OPS08-BP02 워크로드 로그 분석

관련 문서:

- Using ServiceLens to Monitor Application Health
- Exploring Trace Data with X-Ray Analytics
- Detecting Anomalies in Traces with X-Ray Insights
- Continuous Monitoring with CloudWatch Synthetics

관련 비디오:

- Analyze and Debug Applications Using Amazon CloudWatch Synthetics & AWS X-Ray
- Use AWS X-Ray Insights

관련 예제:

- One Observability 워크숍
- Implementing X-Ray with AWS Lambda
- CloudWatch Synthetics Canary Templates

OPS08-BP04 실행 가능한 알림 생성

애플리케이션 동작의 편차를 즉시 감지하고 이에 대응하는 것이 중요합니다. 특히 중요한 것은 핵심 성과 지표(KPI)를 기반으로 한 결과가 위험에 처하거나 예상치 못한 이상 현상이 발생할 때를 인식하는 것입니다. KPI에 기반한 알림을 통해 수신되는 신호가 비즈니스 또는 운영상의 영향과 직접 연계되도록 할 수 있습니다. 실행 가능한 알림에 대한 이러한 접근 방식은 사전 대응을 촉진하고 시스템 성능 및 신뢰성을 유지하는 데 도움이 됩니다.

원하는 성과: 특히 KPI 결과가 위험할 때 잠재적 문제를 신속하게 식별하고 완화할 수 있도록 시기적절하고 실행 가능한 알림을 받을 수 있습니다.

일반적인 안티 패턴:

- 중요하지 않은 알림을 너무 많이 설정하여 알림으로 인한 피로가 발생합니다.
- KPI에 따라 알림의 우선순위를 정하지 않아 문제가 비즈니스에 미치는 영향을 파악하기 어렵습니다.
- 근본 원인 해결을 소홀히 하여 동일한 문제에 대해 반복적인 알림이 발생합니다.

이 모범 사례 확립의 이점:

- 실행 가능하고 관련성이 높은 알림에 집중하여 알림 피로가 줄어듭니다.
- 사전 예방적 문제 감지 및 완화를 통해 시스템 가동 시간 및 신뢰성을 개선했습니다.
- 널리 사용되는 알림 및 커뮤니케이션 도구와 통합하여 팀 협업을 강화하고 문제를 더 빠르게 해결합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

효과적인 알림 메커니즘을 만들려면 KPI를 기반으로 한 결과가 위험에 처하거나 이상 징후가 감지될 때 플래그를 표시하는 지표, 로그 및 추적 데이터를 사용하는 것이 중요합니다.

구현 단계

1. 핵심 성과 지표(KPI) 결정: 애플리케이션의 KPI를 식별합니다. 알림을 이러한 KPI와 연계하여 비즈니스에 미치는 영향을 정확하게 반영해야 합니다.
2. 이상 감지 구현:
   - Amazon CloudWatch 이상 탐지 사용: 비정상적인 패턴을 자동으로 탐지하도록 Amazon CloudWatch 이상 탐지를 설정하면 실제 이상 징후가 있을 때만 알림을 생성할 수 있습니다.
   - AWS X-Ray 인사이트 사용:
     a. X-Ray Insights를 설정하여 추적 데이터에서 이상을 감지합니다.
     b. 탐지된 문제에 대해 알림을 받을 수 있도록 X-Ray Insights에 대한 알림을 구성합니다.
   - Amazon DevOps Guru 통합:
     a. Amazon DevOps Guru의 기계 학습 기능을 활용하여 기존 데이터로 운영 이상 징후를 탐지합니다.
     b. DevOps Guru의 알림 설정으로 이동하여 이상 알림을 설정합니다.
3. 실행 가능한 알림 구현: 즉각적인 조치를 위한 적절한 정보를 제공하는 알림을 설계합니다.
   1. Amazon EventBridge 규칙을 사용하여 AWS Health 이벤트를 모니터링하거나 AWS Health API와 프로그래밍 방식으로 통합하여 AWS Health 이벤트를 수신할 때 작업을 자동화합니다. 이러한 작업은 계획된 모든 수명 주기 이벤트 메시지를 채팅 인터페이스로 보내는 것과 같은 일반적인 작업이거나 IT 서비스 관리 도구에서 워크플로를 시작하는 것과 같은 구체적인 작업일 수 있습니다.
4. 알림 피로 감소: 중요하지 않은 알림을 최소화합니다. 대수롭지 않은 알림으로 팀이 부담을 느끼면 중요한 문제를 감독하지 못할 수 있고 결과적으로 알림 메커니즘의 전반적인 효율성이 떨어질 수 있습니다.

5. 복합 경보 설정: Amazon CloudWatch 복합 경보를 사용하여 여러 경보를 통합합니다.
6. 알림 도구와 통합: Ops Genie 및 PagerDuty와 같은 도구를 통합합니다.
7. 채팅 애플리케이션 내 Amazon Q Developer 참여: 채팅 애플리케이션 내 Amazon Q Developer를 통합하여 Amazon Chime, Microsoft Teams 및 Slack에 알림을 전달합니다.
8. 로그 기반 알림: CloudWatch의 로그 지표 필터를 사용하여 특정 로그 이벤트를 기반으로 경보를 생성합니다.
9. 검토 및 반복: 알림 구성을 정기적으로 재검토하고 개선합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS04-BP03 사용자 경험 원격 측정 구현
- OPS04-BP04 종속성 원격 측정 구현
- OPS04-BP05 분산 추적 구현
- OPS08-BP01 워크로드 지표 분석
- OPS08-BP02 워크로드 로그 분석
- OPS08-BP03 워크로드 추적 데이터 분석

관련 문서:

- Amazon CloudWatch 경보 사용
- 복합 경보 생성
- 이상 탐지를 기반으로 CloudWatch 경보 생성
- DevOps Guru Notifications
- X-ray insights notifications
- 상호작용형 ChatOps로 AWS 리소스 모니터링, 운영 및 문제 해결
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

관련 비디오:

- Create Composite Alarms in Amazon CloudWatch
- Amazon Q Developer in chat applications Overview
- AWS On Air ft. Mutative Commands in Amazon Q Developer in chat applications

관련 예제:

- Alarms, incident management, and remediation in the cloud with Amazon CloudWatch
- Tutorial: Creating an Amazon EventBridge rule that sends notifications to Amazon Q Developer in chat applications
- One Observability 워크숍

OPS08-BP05 대시보드 만들기

대시보드는 워크로드의 원격 측정 데이터를 사람 중심으로 볼 수 있는 보기입니다. 중요한 시각적 인터페이스를 제공하지만 경고 메커니즘을 대체하는 것이 아니라 보완해야 합니다. 주의를 기울여 제작하면 시스템 상태 및 성능에 대한 빠른 인사이트를 제공할 뿐만 아니라 이해관계자에게 비즈니스 성과 및 문제의 영향에 대한 실시간 정보를 제공할 수 있습니다.

원하는 성과:

시각적 표현을 사용하여 시스템 및 비즈니스 상태에 대한 명확하고 실행 가능한 인사이트를 제공합니다.

일반적인 안티 패턴:

- 너무 많은 지표로 인해 대시보드가 지나치게 복잡해집니다.
- 이상 항목 탐지에 대한 경고 없이 대시보드를 사용합니다.
- 워크로드가 진화해도 대시보드를 업데이트하지 않습니다.

이 모범 사례의 이점:

- 중요한 시스템 지표와 KPI에 대한 가시성을 즉각적으로 확보합니다.
- 이해관계자 커뮤니케이션 및 이해 강화.
- 운영 문제의 영향에 대해 신속한 인사이트를 얻습니다.

이 모범 사례가 확립되지 않을 경우 위험 수준: 중간

구현 지침

비즈니스 중심 대시보드

비즈니스 KPI에 맞게 조정된 대시보드는 다양한 이해관계자를 참여시킵니다. 이러한 개인은 시스템 지표에 관심이 없을 수도 있지만 이러한 수치가 비즈니스에 미치는 영향을 이해하는 데 관심이 있습니다. 비즈니스 중심 대시보드를 사용하면 모니터링 및 분석되는 모든 기술 및 운영 지표가 중요한 비즈니스 목표와 동기화됩니다. 이러한 정렬은 모든 사람이 무엇이 필수적이고 무엇이 아닌지에 대해 동일한 이해를 가질 수 있도록 명확성을 제공합니다. 또한 비즈니스 KPI를 강조하는 대시보드는 실행 가능성이 더 높은 경향이 있습니다. 이해관계자는 운영 상태, 주의가 필요한 영역, 비즈니스 성과에 미치는 잠재적 영향을 빠르게 이해할 수 있습니다.

이를 염두에 두고 대시보드를 만들 때는 기술 지표와 비즈니스 KPI 간에 균형을 유지해야 합니다. 둘 다 중요하지만 다양한 청중을 수용하도록 해야 합니다. 시스템의 상태와 성능을 전체적으로 볼 수 있는 동시에 주요 비즈니스 성과와 그 영향을 강조하는 대시보드를 사용하는 것이 가장 좋습니다.

Amazon CloudWatch 대시보드는 CloudWatch 콘솔에서 사용자 지정이 가능한 홈 페이지로, 다른 AWS 리전 및 계정에 분산되어 있는 리소스를 비롯하여 단일 보기에서 리소스를 모니터링하는 데 사용할 수 있습니다.

구현 단계

1. 기본 대시보드 생성: CloudWatch에서 새 대시보드를 생성하고 설명이 포함된 이름을 지정합니다.
2. 마크다운 위젯 사용: 지표를 자세히 살펴보기 전에 마크다운 위젯을 사용하여 대시보드 상단에 텍스트 컨텍스트를 추가합니다. 여기에는 대시보드에서 다루는 내용, 표시된 지표의 중요성이 설명되어야 하며 다른 대시보드 및 문제 해결 도구에 대한 링크도 포함될 수 있습니다.
3. 대시보드 변수 생성: 유연한 동적 대시보드 보기를 위해 해당되는 경우 대시보드 변수를 통합합니다.
4. 지표 위젯 생성: 애플리케이션이 내보내는 다양한 지표를 시각화하도록 지표 위젯을 추가합니다. 그리고 시스템 상태 및 비즈니스 결과를 효과적으로 나타내도록 이 위젯을 조정합니다.
5. 로그 인사이트 쿼리: CloudWatch 로그 인사이트를 활용하여 로그에서 실행 가능한 지표를 도출하고 대시보드에 이러한 인사이트를 표시합니다.
6. 경보 설정: CloudWatch 경보를 대시보드에 통합하여 임곗값을 위반하는 모든 지표를 빠르게 확인할 수 있습니다.
7. Contributor Insights 사용: CloudWatch Contributor Insights를 통합하여 카디널리티가 높은 필드를 분석하고 리소스를 가장 많이 사용하는 항목을 더 명확하게 파악할 수 있습니다.

8. 사용자 지정 위젯 설계: 표준 위젯으로는 충족되지 않는 특정 요구 사항에 대해서는 사용자 지정 위젯을 생성하는 방법을 고려합니다. 사용자 지정 위젯은 다양한 데이터 소스에서 데이터를 가져오거나 고유한 방식으로 데이터를 표현할 수 있습니다.
9. AWS Health 사용: AWS Health는 AWS 클라우드 리소스 상태에 대한 신뢰할 수 있는 정보 소스입니다. AWS Health Dashboard를 바로 사용하거나, 자체 대시보드 및 도구의 AWS Health 데이터를 사용하여 정보에 입각한 결정을 내리는 데 적합한 정보를 확보할 수 있습니다.
10. 반복 및 개선: 애플리케이션이 발전함에 따라 정기적으로 대시보드를 재검토하여 관련성을 확인합니다.

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS08-BP01 워크로드 지표 분석
- OPS08-BP02 워크로드 로그 분석
- OPS08-BP03 워크로드 추적 데이터 분석
- OPS08-BP04 실행 가능한 알림 생성

관련 문서:

- 운영 가시성을 위한 대시보드 구축
- Amazon CloudWatch 대시보드 사용

관련 비디오:

- Create Cross Account & Cross Region CloudWatch Dashboards
- AWS re:Invent 2021 - Gain enterprise visibility with AWS 클라우드 operation dashboards)

관련 예제:

- One Observability 워크숍
- Application Monitoring with Amazon CloudWatch
- AWS Health Events Intelligence Dashboards and Insights
- Visualize AWS Health events using Amazon Managed Grafana

OPS 9. 운영 상태를 어떻게 파악하나요?

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

모범 사례

- OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정
- OPS09-BP02 상태 및 추세를 전달하여 운영에 대한 가시성 확보
- OPS09-BP03 운영 지표 검토 및 개선 우선순위 지정

OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정

조직의 운영 성공을 정의하는 목표와 KPI를 확보하고 지표가 이를 반영하는지 결정하세요. 기준선을 참조 지점으로 설정하고 정기적으로 재평가하세요. 평가를 위해 팀으로부터 이러한 지표를 수집하는 메커니즘을 개발하세요. DevOps Research and Assessment(DORA) 지표는 소프트웨어 제공의 DevOps 방식에 대한 진행 상황을 측정하는 인기 있는 방법을 제공합니다.

원하는 성과:

- 조직이 운영 팀을 위해 목표 및 KPI를 게시하고 공유합니다.
- 이러한 KPI를 반영하는 지표를 설정합니다. 예제로 다음이 포함될 수 있습니다.
  - 티켓 대기열 길이 또는 티켓의 평균 수명
  - 문제 유형별로 그룹화된 티켓 수
  - 표준화된 운영 절차(SOP)를 사용하거나 사용하지 않고 문제를 해결하는 데 소요된 시간
  - 실패한 코드 푸시를 복구하는 데 소요된 시간
  - 통화 볼륨

일반적인 안티 패턴:

- 개발자가 문제 해결 작업을 수행할 수 밖에 없기 때문에 배포 기한을 놓치는 경우가 있습니다. 개발팀은 더 많은 인력을 확보하기 위해 노력하고 있지만 소요되는 시간을 측정할 수 없기 때문에 필요한 인원을 정량화할 수 없습니다.
- 티어 1 데스크는 사용자 통화를 처리하도록 설정됩니다. 시간이 지나면서 더 많은 워크로드가 추가되었지만 티어 1 데스크에는 인력이 할당되지 않습니다. 통화 시간이 늘어나고 해결 없이 문제가 더 길어지면서 고객 만족도가 떨어지지만 경영진은 그러한 지표를 발견하지 못해 조치를 취하지 못합니다.

- 문제가 되는 워크로드는 유지 관리를 위해 별도의 운영 팀에 전달되었습니다. 다른 워크로드와 달리 이 새 워크로드는 적절한 설명서 및 런북과 함께 제공되지 않습니다. 따라서 팀은 문제를 해결하고 장애를 해결하는 데 더 많은 시간을 할애합니다. 그러나 이를 문서화하는 지표가 없기 때문에 책임 소재를 찾기가 어렵습니다.

이 모범 사례 확립의 이점: 워크로드 모니터링을 통해 애플리케이션 및 서비스의 상태를 확인하여 모니터링 운영 팀이 소유자에게 워크로드 소비자들 사이에서 일어나는 변화(예: 비즈니스 요구 사항 변화)에 대한 인사이트를 제공할 수 있습니다. 운영 상태를 반영할 수 있는 지표를 만들어 이러한 팀의 효율성을 측정하고 비즈니스 목표와 비교하여 평가합니다. 지표를 통해 지원 문제를 강조하거나 서비스 수준 목표에서 벗어나는 편차가 발생하는 시점을 파악할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

비즈니스 리더 및 이해관계자와 일정을 맞춰 서비스의 전반적인 목표를 결정합니다. 다양한 운영팀의 업무가 무엇인지 그리고 어떤 과제에 직면할 수 있는지 결정합니다. 이를 사용하여 이러한 운영 목표를 반영할 수 있는 핵심 성과 지표(KPI)를 브레인스토밍하세요. 여기에는 고객 만족도, 기능 구상부터 배포까지의 시간, 평균 문제 해결 시간, 비용 효율성이 포함될 수 있습니다.

KPI를 바탕으로 이러한 목표를 가장 잘 반영할 수 있는 지표와 데이터 소스를 식별하세요. 고객 만족도는 통화 대기 또는 응답 시간, 만족도 점수, 제기된 문제 유형과 같은 다양한 지표의 조합일 수 있습니다. 배포 시간은 테스트 및 배포에 필요한 시간과 추가해야 하는 배포 후 수정 사항의 총합일 수 있습니다. 다양한 유형의 문제에 소요된 시간(또는 해당 문제의 수)을 보여주는 통계를 통해 목표 집중이 필요한 부분을 파악할 수 있습니다.

리소스

관련 문서:

- QuickSight - KPI 사용
- Amazon CloudWatch 지표 사용
- 대시보드 구축
- How to track your cost optimization KPIs with KPI Dashboard
- AWS DevOps Guidance

관련 예제:

- Monitor the performance of your software delivery using native AWS monitoring and observability tools
- Balance deployment speed and stability with DORA metrics
- Example MLOps operational metrics in the financial services industry
- How to track your cost optimization KPIs with the KPI Dashboard

OPS09-BP02 상태 및 추세를 전달하여 운영에 대한 가시성 확보

결과가 위험에 처할 수 있는 시점, 추가된 작업을 지원할 수 있는지 여부, 변화가 팀에 미친 영향을 파악하려면 운영 상태와 추세 동향을 알아야 합니다. 운영 이벤트 중에 사용자와 운영팀이 참조하여 정보를 얻을 수 있는 상태 페이지를 마련하면 커뮤니케이션 채널에 가해지는 부담을 줄이고 정보를 사전에 전파할 수 있습니다.

원하는 성과:

- 운영 책임자는 팀이 얼만큼의 통화 볼륨을 받고 있는지, 배포와 같이 어떤 작업을 진행 중인지 한눈에 파악할 수 있습니다.
- 정상 운영에 영향이 발생할 경우 이해관계자와 사용자 커뮤니티에 알림이 전달됩니다.
- 조직 경영진과 이해관계자는 경고 또는 영향에 대응하여 상태 페이지를 확인하고 연락처, 티켓 정보, 예상 복구 시간 등 운영 이벤트와 관련된 정보를 얻을 수 있습니다.
- 경영진 및 기타 이해관계자에게 보고서를 제공하여 일정 기간의 통화 볼륨, 사용자 만족도 점수, 미결 티켓 수 및 연령과 같은 운영 통계를 보여줍니다.

일반적인 안티 패턴:

- 워크로드가 다운되어 서비스를 사용할 수 없게 됩니다. 사용자가 무슨 일이 일어나고 있는지 알려달라고 요청하면 통화 볼륨이 급증합니다. 관리자는 볼륨에 추가하여 누가 문제를 해결하고 있는지 확인하도록 요청합니다. 여러 운영 팀이 조사를 위해 중복적인 노력을 기울입니다.
- 새로운 기능에 대한 기대로 인해 여러 인력이 엔지니어링 작업에 재배치됩니다. 백필은 제공되지 않으며 문제 해결 시간이 급증합니다. 이 정보는 캡처되지 않으며, 몇 주 후 사용자 피드백이 만족스럽지 못한 후에야 경영진이 문제를 알게 됩니다.

이 모범 사례 확립의 이점: 비즈니스에 영향을 미치는 운영 이벤트 중에는 상황을 파악하기 위해 노력하는 여러 팀의 정보를 쿼리하느라 많은 시간과 에너지가 낭비될 수 있습니다. 널리 보급된 상태 페이지와 대시보드를 구축함으로써 이해관계자들은 문제가 감지되었는지 여부, 문제의 주체가 누구인지,

정상 운영 상태로 돌아갈 것으로 예상되는 시기와 같은 정보를 신속하게 얻을 수 있습니다. 이렇게 하면 팀원들이 다른 사람에게 상태를 전달하는 데 너무 많은 시간을 소비하지 않고 문제를 해결하는 데 더 많은 시간을 할애할 수 있습니다.

또한 대시보드와 보고서는 의사 결정권자와 이해관계자에게 운영 팀이 비즈니스 요구 사항에 어떻게 대응할 수 있는지, 리소스가 어떻게 할당되고 있는지를 파악할 수 있는 인사이트를 제공할 수 있습니다. 이는 비즈니스를 지원하는 데 필요한 적절한 리소스가 마련되어 있는지 판단하는 데 매우 중요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

운영 팀의 현재 주요 지표를 보여주는 대시보드를 구축하고 운영 리더와 경영진이 쉽게 액세스할 수 있도록 하세요.

인시던트나 이벤트가 언제 일어나는지, 누가 소유권을 갖고 있는지, 누가 대응을 조율하는지 알 수 있도록 신속하게 업데이트할 수 있는 상태 페이지를 구축하세요. 이 페이지에서 사용자가 고려해야 하는 단계 또는 해결 방법을 공유하고 위치를 널리 알리세요. 알 수 없는 문제가 발생하면 사용자가 먼저 이 위치를 확인하도록 권장합니다.

시간 경과에 따른 운영 상태를 보여주는 보고서를 수집 및 제공하고, 이를 리더와 의사 결정권자에게 배포하여 과제 및 요구 사항과 함께 운영 업무를 설명합니다.

목표와 KPI를 가장 잘 반영하고 변화를 주도하는 데 어떤 영향을 미쳤는지 이러한 지표와 보고서를 팀 간에 공유하세요. 이러한 활동에 시간을 할애하여 팀 내부 및 팀 간 운영의 중요성을 높이세요.

자체 대시보드와 함께 AWS Health를 사용하거나 AWS Health 이벤트를 여기에 통합하여 팀에서 애플리케이션 문제를 AWS 서비스 상태와 연관시킬 수 있도록 합니다.

리소스

관련 모범 사례:

- OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정

관련 문서:

- Measure Progress
- 운영 가시성을 위한 대시보드 구축

관련 예제:

- Data Operations
- How to track your cost optimization KPIs with KPI Dashboard
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations

OPS09-BP03 운영 지표 검토 및 개선 우선순위 지정

운영 상태를 검토하기 위한 전용 시간과 리소스를 따로 확보하면 일상적인 업무 부서에 서비스를 제공하는 것이 최우선 과제가 될 수 있습니다. 운영 리더와 이해관계자를 모아 정기적으로 지표를 검토하고, 목표와 목적을 재확인 또는 수정하고, 개선의 우선순위를 정하세요.

원하는 성과:

- 운영 책임자와 직원은 정기적으로 만나 지정된 보고 기간의 지표를 검토합니다. 도전 과제를 전달하고, 긍정적인 결과를 축하하며, 배운 교훈을 공유합니다.
- 이해관계자와 비즈니스 리더는 운영 현황에 대해 정기적으로 브리핑을 받고 목표, KPI 및 향후 이니셔티브에 대한 의견을 요청받습니다. 서비스 제공, 운영 및 유지 관리 사이에서 장단점을 논의하고 상황에 맞게 적용합니다.

일반적인 안티 패턴:

- 신제품이 출시되었지만 티어 1 및 티어 2 운영 팀은 적절한 지원 교육을 받지 못했거나 추가 인력을 배치 받지 못했습니다. 티켓 해결 시간 감소 및 인시던트 볼륨 증가를 보여주는 지표는 리더에게 보이지 않습니다. 불만을 품은 사용자가 플랫폼을 떠나면서 구독 수가 감소하기 시작하면 몇 주 후 조치가 취해집니다.
- 워크로드에 대한 유지 관리를 수행하는 수동 프로세스가 오랫동안 사용되어 왔습니다. 자동화에 대한 열망은 있었지만 시스템의 중요도가 낮았기 때문에 우선순위가 낮았습니다. 그러나 시간이 흐르면서 시스템의 중요성이 커져 이제는 이러한 수동 프로세스가 운영 시간의 대부분을 차지하게 됩니다. 운영 부서에 더 많은 도구를 제공하는 데 필요한 리소스가 계획되어 있지 않아 업무량이 증가함에 따라 직원 소진 문제가 발생합니다. 직원들이 다른 경쟁업체로 떠나고 있다는 소식이 전해지면 경영진은 이를 인지하게 됩니다.

이 모범 사례 확립의 이점: 일부 조직에서는 서비스 제공과 신제품 또는 서비스에 동일한 시간과 관심을 할당하는 것이 어려울 수 있습니다. 이 경우 예상 서비스 수준이 서서히 저하되어 업무 부서에 문제가 발생할 수 있습니다. 비즈니스가 성장해도 운영은 변화하지 않고 발전하지 않으며 곧 뒤쳐질 수 있기 때문입니다. 운영 팀에서 수집한 인사이트를 정기적으로 검토하지 않으면 비즈니스에 미치는 위험

은 너무 늦었을 때만 가시화될 수 있습니다. 운영 담당자와 경영진 모두에게 지표와 절차를 검토하는 데 시간을 할당함으로써 운영팀이 수행하는 중요한 역할을 가시화하고 위험 수준이 위험한 수준에 도달하기 훨씬 전에 위험을 식별할 수 있습니다. 운영 팀은 임박한 비즈니스 변경 및 이니셔티브를 더 잘 파악하여 사전 조치를 취할 수 있습니다. 운영 지표에 대한 리더십 가시성은 이러한 팀이 내부 및 외부 모두에서 고객 만족도에서 수행하는 역할을 보여주고, 팀이 우선순위에 대한 선택을 더 잘 판단하거나 운영팀이 새로운 비즈니스 및 워크로드 이니셔티브를 통해 변화하고 발전하는 데 필요한 시간과 리소스를 확보할 수 있도록 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

시간을 할애하여 이해관계자와 운영 팀 간의 운영 지표를 검토하고 보고서 데이터를 검토하세요. 이러한 보고서를 조직의 목표 및 목적의 맥락에 비추어 충족되고 있는지 판단하세요. 목표가 명확하지 않거나, 요청한 내용과 주어진 내용이 상충할 수 있는 모호함의 원인을 파악하세요.

시간, 인력, 도구가 운영 성과에 도움이 될 수 있는 부분을 파악하세요. 이것이 어떤 KPI에 영향을 미칠지 그리고 어떤 성공 목표를 세워야 하는지 결정하세요. 정기적으로 재검토하여 사업 부문을 지원할 수 있는 충분한 리소스가 운영되고 있는지 확인하세요.

리소스

관련 문서:

- Amazon Athena
- Amazon CloudWatch 지표 및 차원 참조
- Amazon QuickSight
- AWS Glue
- AWS Glue Data Catalog
- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- Amazon CloudWatch 지표 사용

OPS 10. 워크로드 및 운영 이벤트를 어떻게 관리하나요?

이벤트로 인해 워크로드가 중단될 가능성을 최소화할 수 있도록 이벤트 대응을 위한 절차를 준비하고 검증합니다.

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 서비스에 영향을 미치는 이벤트에 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

이벤트, 인시던트 및 문제를 효율적으로 관리하는 능력은 워크로드 상태 및 성능을 유지하는 데 매우 중요합니다. 효과적인 대응 및 해결 전략을 개발하려면 이러한 요소 간의 차이점을 인식하고 이해하는 것이 매우 중요합니다. 각 측면에 대해 잘 정의된 프로세스를 수립하고 준수하면 팀이 발생하는 모든 운영 문제를 신속하고 효과적으로 처리하는 데 도움이 됩니다.

원하는 성과: 체계적으로 문서화되고 중앙 집중식으로 저장된 프로세스를 통해 운영 이벤트, 인시던트 및 문제를 효과적으로 관리합니다. 이러한 프로세스는 변경 사항을 반영하여 지속적으로 업데이트되므로 처리가 간소화되고 높은 서비스 신뢰성과 워크로드 성능이 유지됩니다.

일반적인 안티 패턴:

- 이벤트에 사전 대응보다는 사후 대응 방식으로 대응합니다.
- 다양한 유형의 이벤트 또는 인시던트에 대해 일관되지 않은 접근 방식을 취합니다.
- 조직은 향후 인시던트 방지를 위해 인시던트를 분석하고 학습하는 과정을 진행하지 않습니다.

이 모범 사례 확립의 이점:

- 간소화되고 표준화된 대응 프로세스.
- 인시던트가 서비스 및 고객에게 미치는 영향 감소.
- 신속한 문제 해결.
- 운영 프로세스의 지속적인 개선.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

이 모범 사례를 구현하면 워크로드 이벤트를 추적하게 됩니다. 인시던트 및 문제를 처리하기 위한 프로세스를 보유하게 됩니다. 이 프로세스는 문서화되고 공유되며 자주 업데이트됩니다. 문제가 파악되면 우선순위가 지정되고 해결됩니다.

이벤트, 인시던트 및 문제에 대한 이해

- 이벤트: 이벤트는 동작, 발생 또는 상태 변경을 관찰한 결과일 수 있습니다. 이벤트는 계획된 것일 수도 있고 계획되지 않은 것일 수도 있으며 워크로드의 내부 또는 외부에서 발생할 수 있습니다.
- 인시던트: 인시던트는 예상치 못한 중단이나 서비스 품질 저하와 같이 대응이 필요한 이벤트를 말합니다. 이는 정상적인 워크로드 운영을 복원하기 위해 즉각적인 조치가 필요한 장애를 나타냅니다.
- 문제: 문제는 하나 이상의 인시던트의 근본 원인을 말합니다. 문제를 식별하고 해결하려면 인시던트를 더 깊이 파고들어 향후 발생을 방지해야 합니다.

구현 단계

이벤트

1. 이벤트 모니터링:
   - 관찰성을 구현하고 워크로드 관찰성을 활용하세요.
   - 사용자, 역할 또는 AWS 서비스에서 수행한 모니터링 작업은 AWS CloudTrail에 이벤트로 기록됩니다.
   - Amazon EventBridge에서 실시간으로 애플리케이션의 운영 변화에 대응합니다.
   - AWS Config에서 리소스 구성 변경 사항을 지속적으로 평가, 모니터링 및 기록합니다.
2. 프로세스 생성:
   - 어떤 이벤트가 중요하고 모니터링이 필요한지 평가하는 프로세스를 개발합니다. 여기에는 정상 및 비정상 활동에 대한 임곗값 및 파라미터 설정이 포함됩니다.
   - 이벤트를 인시던트로 에스컬레이션하는 기준을 결정합니다. 심각도, 사용자에게 미치는 영향 또는 예상 행동과의 차이를 토대로 결정할 수 있습니다.
   - 이벤트 모니터링 및 대응 프로세스를 정기적으로 검토합니다. 여기에는 과거 인시던트 분석, 임곗값 조정, 경고 메커니즘 개선이 포함됩니다.

인시던트

1. 인시던트에 대응:

- 관찰성 도구의 인사이트를 사용하여 인시던트를 빠르게 식별하고 이에 대응합니다.
- AWS Systems Manager Ops Center를 구현하여 운영 항목 및 인시던트를 집계하고 체계화하며 우선순위를 지정합니다.
- 심층적인 분석 및 문제 해결을 위해 Amazon CloudWatch 및 AWS X-Ray 같은 서비스를 사용합니다.
- 향상된 인시던트 관리를 위해 선제적, 사전 예방 및 감지 기능을 활용하는 AWS Managed Services(AMS)는 고려하세요. AMS는 모니터링, 인시던트 탐지 및 대응, 보안 관리와 같은 서비스를 통해 운영 지원을 확대합니다.
- Enterprise Support 고객은 프로덕션 워크로드에 대한 지속적인 사전 모니터링 및 인시던트 관리를 제공하는 AWS 인시던트 탐지 및 대응을 사용할 수 있습니다.

2. 인시던트 관리 프로세스 만들기:
   - 명확한 역할, 커뮤니케이션 프로토콜, 해결 단계를 포함한 구조화된 인시던트 관리 프로세스를 수립합니다.
   - 효율적인 대응 및 조정을 위해 채팅 애플리케이션 내 Amazon Q Developer와 같은 도구를 통해 인시던트 관리를 통합합니다.
   - 각 범주에 대해 사전 정의된 인시던트 대응 계획을 사용하여 심각도를 기준으로 인시던트를 분류합니다.
3. 학습 및 개선:
   - 근본 원인을 이해하고 해결 방법의 효과를 확인하기 위해 인시던트 사후 분석을 수행합니다.
   - 검토 및 발전하는 관행을 토대로 대응 계획을 지속적으로 업데이트하고 개선합니다.
   - 팀 전반에서 학습한 내용을 문서화하고 공유하여 운영 복원력을 개선합니다.
   - Enterprise Support 고객은 기술 계정 관리자로부터 Incident Management 워크숍을 요청할 수 있습니다. 이 안내 워크숍에서는 기존 인시던트 대응 계획을 테스트하고 개선할 수 있는 영역을 식별하도록 돕습니다.

문제

1. 문제 파악:
   - 이전 인시던트의 데이터를 사용하여 심층적인 시스템 문제를 시사하는 반복 패턴을 식별합니다.
   - AWS CloudTrail 및 Amazon CloudWatch와 같은 도구를 활용하여 추세를 분석하고 근본적인 문제를 파악합니다.
   - 운영, 개발, 사업부를 비롯한 여러 팀이 참여하여 근본 원인에 대한 다양한 관점을 확보합니다.
2. 문제 관리 프로세스 만들기:

- 빠른 해결보다는 장기적인 해결책에 초점을 맞춰 체계적인 문제 관리 프로세스를 개발합니다.
- 근본 원인 분석(RCA) 기술을 통합하여 인시던트의 근본 원인을 조사하고 이해합니다.
- 결과를 기반으로 운영 정책, 절차 및 인프라를 업데이트하여 재발을 방지합니다.

3. 지속적인 개선:
   - 지속적인 학습과 개선의 문화를 조성하여 팀이 잠재적인 문제를 사전에 식별하고 해결하도록 독려합니다.
   - 진화하는 비즈니스 및 기술 환경에 맞게 문제 관리 프로세스와 도구를 정기적으로 검토하고 수정합니다.
   - 조직 전반에 걸쳐 인사이트와 모범 사례를 공유하여 보다 복원력 있고 효율적인 운영 환경을 구축합니다.
4. AWS Support 참여:
   - 선제적 지침 및 최적화 권장 사항에 대해 AWS지원 리소스(예: AWS Trusted Advisor)를 사용합니다.
   - Enterprise Support 고객은 AWS Countdown과 같은 전문 프로그램을 통해 중요 이벤트 발생 시 지원을 받을 수 있습니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS04-BP02 애플리케이션 원격 측정 구현
- OPS07-BP03 런북을 사용한 절차 수행
- OPS07-BP04 플레이북을 사용하여 문제 조사
- OPS08-BP01 워크로드 지표 분석
- OPS11-BP02 인시던트 사후 분석 수행

관련 문서:

- AWS Security Incident Response Guide
- AWS Incident Detection and Response
- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management

- Incident Management in the Age of DevOps and SRE
- PagerDuty - What is Incident Management?

관련 비디오:

- Top incident response tips from AWS
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 yrs of Amazon operational excellence
- AWS re:Invent 2022 - AWS Incident Detection and Response (SUP201)
- Introducing Incident Manager from AWS Systems Manager

관련 예제:

- AWS Proactive Services – Incident Management 워크숍
- How to Automate Incident Response with PagerDuty and AWS Systems Manager Incident Manager
- Engage Incident Responders with the On-Call Schedules in AWS Systems Manager Incident Manager
- Improve the Visibility and Collaboration during Incident Handling in AWS Systems Manager Incident Manager
- Incident reports and service requests in AMS

관련 서비스:

- Amazon EventBridge

OPS10-BP02 알림별 프로세스 마련

효과적이고 효율적인 인시던트 관리를 위해서는 시스템의 각 알림에 대해 명확하고 정의된 프로세스를 마련하는 것이 필수적입니다. 이렇게 하면 모든 알림이 구체적이고 실행 가능한 대응으로 이어져 운영의 신뢰성과 대응력이 향상됩니다.

원하는 성과: 모든 알림은 구체적이고 잘 정의된 대응 계획을 개시합니다. 가능한 경우 명확한 소유권과 정의된 에스컬레이션 경로를 통해 대응이 자동화됩니다. 알림은 모든 운영자가 일관되고 효과적으로 대응할 수 있도록 최신 지식 베이스에 연결됩니다. 대응이 전반적으로 빠르고 균일하여 운영 효율성과 신뢰성이 향상됩니다.

일반적인 안티 패턴:

- 알림에는 사전 정의된 대응 프로세스가 없으므로 임시 조치 및 문제 해결이 지연될 수 있습니다.
- 알림 오버로드로 인해 중요한 알림이 간과됩니다.
- 명확한 소유권과 책임이 없기 때문에 알림이 일관되지 않은 방식으로 처리됩니다.

이 모범 사례 확립의 이점:

- 실행 가능한 알림만 발생시켜 알림 피로를 줄입니다.
- 운영 문제의 평균 해결 시간(MTTR)을 단축합니다.
- 평균 조사 시간(MTTI)이 단축되어 MTTR을 단축합니다.
- 운영 대응 규모를 조정할 수 있는 기능을 개선합니다.
- 운영 이벤트 처리의 일관성과 신뢰성이 향상됩니다.

예를 들어 애플리케이션 경보, 운영 문제 및 계획된 수명 주기 이벤트(클러스터가 자동 업데이트되기 전에 Amazon EKS 버전 업데이트 등)를 포함하여 중요한 계정에 대한 AWS Health 이벤트에 대해 정의된 프로세스가 있으며 팀이 이러한 이벤트를 적극적으로 모니터링하고, 소통하고, 대응할 수 있는 역량을 제공합니다. 이러한 작업을 통해 AWS 측 변경으로 인한 서비스 중단을 방지하거나 예상치 못한 문제가 발생할 때 더 빠르게 완화할 수 있습니다.

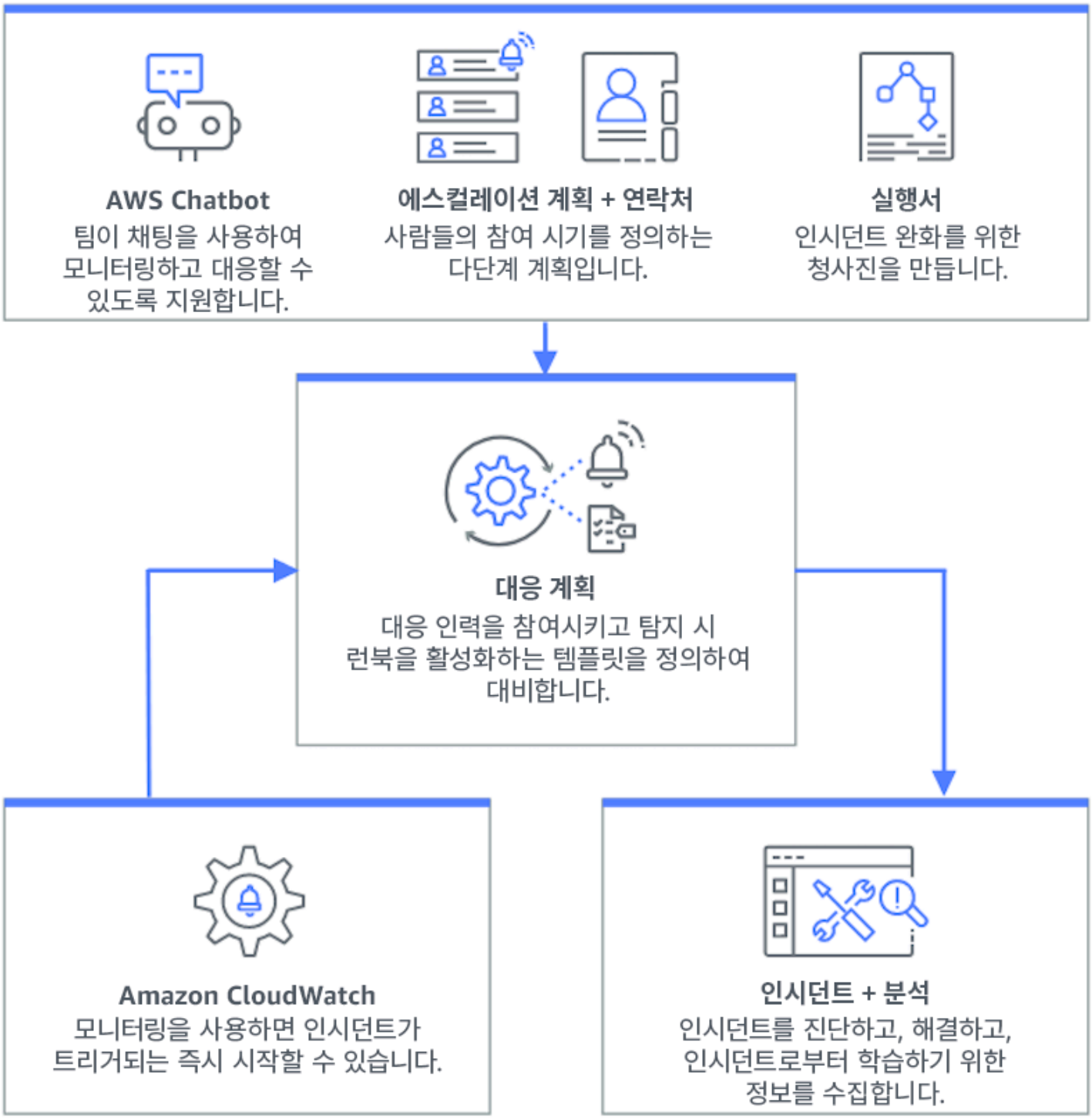
이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

알림별 프로세스를 갖추려면 각 알림에 대한 명확한 대응 계획을 마련하고, 가능한 경우 대응을 자동화하며, 운영 피드백과 변화하는 요구 사항을 기반으로 이러한 프로세스를 지속적으로 개선해야 합니다.

구현 단계

다음 다이어그램은 AWS Systems Manager Incident Manager 내 인시던트 관리 워크플로를 보여줍니다. 이는 Amazon CloudWatch 또는 Amazon EventBridge의 특정 이벤트에 대한 대응으로 인시던트를 자동으로 생성하여 운영 문제에 신속하게 대응할 수 있도록 설계되었습니다. 인시던트가 자동 또는 수동으로 생성되면 Incident Manager에서 인시던트 관리를 중앙 집중화하고 관련 AWS 리소스 정보를 구성하며 사전 정의된 대응 계획을 개시합니다. 여기에는 즉각적인 조치를 위한 Systems Manager Automation 런북 실행과 관련 작업 및 분석을 추적하기 위해 OpsCenter에 상위 운영 작업 항목을 생성하는 것도 포함됩니다. 이 간소화된 프로세스는 AWS 환경 전반에서 인시던트 대응을 가속화하고 조정합니다.



1. 복합 경보 사용: CloudWatch에서 복합 경보를 생성하여 경보를 그룹화하고 노이즈를 줄이며 보다 의미 있는 대응이 가능하게 합니다.

2. AWS Health로 최신 정보를 확인하세요: AWS Health는 AWS 클라우드 리소스 상태에 대한 신뢰할 수 있는 정보 소스입니다. AWS Health를 사용해 계획된 수명 주기 이벤트와 같은 현재 서비스 이벤트 및 예정된 변경 사항을 시각화하고 알림을 받아 영향 완화 조치를 취할 수 있습니다.
   a. AWS 사용자 알림를 통해 이메일 및 채팅 채널에 적합한 AWS Health 이벤트 알림을 생성하고, AWS Health API 또는 Amazon EventBridge를 통해 모니터링 및 알림 도구와 프로그래밍 방식으로 통합할 수 있습니다.
   b. Amazon EventBridge 또는 AWS Health API를 통해 이미 사용할 수 있는 변경 관리 또는 ITSM 도구(예: Jira 또는 ServiceNow)와 통합하여 조치가 필요한 상태 이벤트에 대한 진행 상황을 계획하고 추적하세요.
   c. AWS Organizations를 사용하는 경우 AWS Health에 대한 조직 보기를 활성화하여 계정 간에 AWS Health 이벤트를 집계합니다.
3. Amazon CloudWatch 경보를 Incident Manager와 통합: AWS Systems Manager Incident Manager에서 인시던트를 자동으로 생성하도록 CloudWatch 경보를 구성합니다.
4. Amazon EventBridge를 Incident Manager와 통합: EventBridge 규칙을 만들어 정의된 대응 계획에 따라 이벤트에 대응하고 인시던트를 생성합니다.
5. Incident Manager에서 인시던트 준비:
   - 알림 유형별 세부 대응 계획을 Incident Manager에서 수립합니다.
   - Incident Manager의 대응 계획에 연결된 채팅 애플리케이션 내 Amazon Q Developer를 통해 채팅 채널을 설정하여 Slack, Microsoft Teams 및 Amazon Chime과 같은 여러 플랫폼에서 인시던트 발생 시 실시간 커뮤니케이션을 용이하게 합니다.
   - Incident Manager 내에서 Systems Manager Automation 런북을 통합하여 인시던트에 대한 자동 대응을 유도합니다.

리소스

관련 모범 사례:

- OPS04-BP01 핵심 성과 지표 파악
- OPS08-BP04 실행 가능한 알림 생성

관련 문서:

- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Amazon CloudWatch 경보 사용
- Setting up AWS Systems Manager Incident Manager

- Preparing for incidents in Incident Manager

관련 비디오:

- Top incident response tips from AWS
- re:Invent 2,023 | Manage resource lifecycle events at scale with AWS Health

관련 예제:

- AWS 워크숍 - AWS Systems Manager Incident Manager - Automate incident response to security events

OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정

운영 이벤트에 즉시 대응하는 것이 중요하지만 모든 이벤트가 동일한 것은 아닙니다. 비즈니스 영향을 기준으로 우선순위를 정할 때는 안전, 재정적 손실, 규정 위반 또는 평판 손상과 같은 중대한 결과를 초래할 가능성이 있는 이벤트를 해결하는 데에도 우선순위를 둡니다.

원하는 성과: 운영 이벤트에 대한 대응은 비즈니스 운영 및 목표에 대한 잠재적 영향을 기반으로 우선순위가 지정됩니다. 이렇게 하면 효율적이고 효과적으로 대응할 수 있습니다.

일반적인 안티 패턴:

- 모든 이벤트는 동일한 수준의 긴급도로 처리되므로 중요한 문제를 해결하는 데 혼란과 지연이 발생합니다.
- 영향이 큰 이벤트와 그렇지 않은 이벤트를 구분하지 못해 리소스가 잘못 할당됩니다.
- 조직에 명확한 우선순위 지정 프레임워크가 없기 때문에 운영 이벤트에 대한 대응이 일관되지 않습니다.
- 이벤트는 비즈니스 성과에 미치는 영향보다는 보고된 순서를 기준으로 우선순위가 지정됩니다.

이 모범 사례 확립의 이점:

- 중요한 비즈니스 기능에 먼저 주의를 기울이도록 하여 잠재적 피해를 최소화합니다.
- 여러 동시 이벤트 발생 시 리소스 할당을 개선합니다.
- 조직의 신뢰 유지 및 규제 요구 사항 충족 능력을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

여러 운영 이벤트가 발생하는 경우 영향과 긴급성을 기반으로 우선순위를 정하는 체계적인 접근 방식이 필수적입니다. 이 접근 방식을 사용하면 정보에 입각한 결정을 내리고, 가장 필요한 부분에 노력을 기울이며, 비즈니스 연속성에 대한 위험을 완화할 수 있습니다.

구현 단계

1. 영향 평가: 이벤트가 비즈니스 운영 및 목표에 미치는 잠재적 영향을 기준으로 이벤트의 심각도를 평가하는 분류 체계를 개발합니다. 다음 예에서는 영향 범주를 보여줍니다.

영향 수준	설명
높음	많은 직원이나 고객에게 영향을 미치거나, 높은 재정적 영향, 높은 평판 손상 또는 부상을 초래합니다.
중간	직원 또는 고객 그룹에 영향을 미치거나, 재정적 영향이 크지 않거나, 평판에 어느 정도의 손해를 끼칩니다.
낮음	개별 직원 또는 고객에게 영향을 미치거나, 재정적 영향이 적거나, 평판에 미치는 영향이 적습니다.

2. 긴급성 평가: 안전, 재정적 영향, 서비스 수준에 관한 계약(SLA)과 같은 요소를 고려하여 이벤트에 얼마나 빨리 대응해야 하는지에 대한 긴급 수준을 정의합니다. 다음 예는 긴급성 범주를 보여줍니다.

긴급성 수준	설명
높음	피해가 기하급수적으로 증가하거나, 시간에 민감한 업무가 영향을 받거나, 에스컬레이션이 임박하거나, VIP 사용자 또는 그룹이 영향을 받습니다.

긴급성 수준	설명
중간	피해가 시간이 경과함에 따라 증가하거나 단일 VIP 사용자 또는 그룹이 영향을 받습니다.
낮음	시간이 지남에 따라 미미한 손상이 증가하거나 시간에 민감하지 않은 작업에 영향을 미칩니다.

3. 우선순위 매트릭스 만들기:
   - 매트릭스를 사용하여 영향과 긴급성을 상호 참조하여 다양한 조합에 우선순위 수준을 할당합니다.
   - 운영 이벤트 대응을 담당하는 모든 팀원이 매트릭스에 액세스하고 이를 이해할 수 있도록 하세요.
   - 다음 예제 매트릭스는 긴급성과 영향에 따라 인시던트 심각도를 표시합니다.

긴급성 및 영향	높음	중간	낮음
높음	심각	긴급	높음
중간	긴급	높음	정상
낮음	높음	정상	낮음

4. 교육 및 커뮤니케이션: 대응 팀에 우선순위 매트릭스와 이벤트 중 우선순위 매트릭스 준수의 중요성에 대해 교육합니다. 우선순위 지정 프로세스를 모든 이해관계자에게 전달하여 명확한 기대치를 설정합니다.
5. 인시던트 대응과 통합:
   - 우선순위 매트릭스를 인시던트 대응 계획 및 도구에 통합합니다.
   - 가능한 경우 이벤트의 분류 및 우선순위 지정을 자동화하여 대응 시간을 단축합니다.
   - Enterprise Support 고객은 프로덕션 워크로드에 대한 연중무휴 사전 모니터링 및 인시던트 관리를 제공하는 AWS 인시던트 탐지 및 대응을 활용할 수 있습니다.
6. 검토 및 조정: 우선순위 지정 프로세스의 효과를 정기적으로 검토하고 비즈니스 환경의 피드백과 변화를 기반으로 조정합니다.

리소스

관련 모범 사례:

- OPS03-BP03 에스컬레이션 장려
- OPS08-BP04 실행 가능한 알림 생성
- OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정

관련 문서:

- Atlassian - Understanding incident severity levels
- IT Process Map - Checklist Incident Priority

OPS10-BP04 에스컬레이션 경로 정의

인시던트 대응 프로토콜 내에 명확한 에스컬레이션 경로를 설정하여 시의적절하고 효과적인 조치를 취합니다. 여기에는 에스컬레이션 프롬프트 지정, 에스컬레이션 프로세스 상세 설명, 신속한 의사 결정 및 평균 해결 시간(MTTR) 단축을 위한 사전 승인 조치가 포함됩니다.

원하는 성과: 인시던트를 적절한 담당자에게 에스컬레이션하여 대응 시간과 영향을 최소화하는 체계적이고 효율적인 프로세스입니다.

일반적인 안티 패턴:

- 복구 절차가 명확하지 않으면 중대한 인시던트가 발생했을 때 임시방편책으로 대응해야 합니다.
- 정의된 권한 및 소유권이 없으면 긴급 조치가 필요한 경우 지연이 발생합니다.
- 이해관계자와 고객에게는 기대에 부합하는 정보가 제공되지 않습니다.
- 중요한 결정이 지연됩니다.

이 모범 사례 확립의 이점:

- 사전 정의된 에스컬레이션 절차를 통해 인시던트 대응을 간소화합니다.
- 사전 승인된 조치와 명확한 소유권을 통해 가중 중지 시간을 줄입니다.
- 인시던트 심각도에 따라 리소스 할당 및 지원 수준 조정을 개선합니다.
- 이해관계자 및 고객과의 커뮤니케이션을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

적절하게 정의된 에스컬레이션 경로는 신속한 인시던트 대응에 매우 중요합니다. AWS Systems Manager Incident Manager에서는 인시던트 발생 시 적절한 조치를 취할 수 있도록 적절한 담당자에게 알림을 보내는 구조화된 에스컬레이션 계획 및 당직 일정을 설정할 수 있도록 지원합니다.

구현 단계

1. 에스컬레이션 프롬프트 설정: CloudWatch 경보를 설정하여 AWS Systems Manager Incident Manager에서 인시던트를 생성합니다.
2. 당직 일정 설정: Incident Manager에서 에스컬레이션 경로에 맞게 조정된 당직 일정을 생성합니다. 당직 근무 중인 직원에게 신속하게 조치를 취하는 데 필요한 권한과 도구를 제공합니다.
3. 상세 에스컬레이션 절차:
   - 인시던트를 에스컬레이션해야 하는 구체적인 조건을 결정합니다.
   - Incident Manager에서 에스컬레이션 계획을 생성합니다.
   - 에스컬레이션 채널은 연락처 또는 당직 일정으로 구성되어야 합니다.
   - 각 에스컬레이션 수준에서 팀의 역할과 책임을 정의합니다.
4. 완화 조치 사전 승인: 의사 결정권자와 협업하여 예상 시나리오에 대한 조치를 사전 승인합니다. Incident Manager와 통합된 Systems Manager Automation 런북을 사용하여 인시던트을 빠르게 해결합니다.
5. 소유권 지정: 에스컬레이션 경로의 각 단계에서 내부 소유자를 명확하게 식별합니다.
6. 서드파티 에스컬레이션에 대한 세부 정보:
   - 서드파티의 서비스 수준에 관한 계약(SLA)을 문서화하고 내부 목표에 맞게 조정합니다.
   - 인시던트 발생 시 공급업체 커뮤니케이션을 위한 명확한 프로토콜을 설정합니다.
   - 공급업체 연락처를 인시던트 관리 도구에 통합하여 직접 액세스할 수 있습니다.
   - 서드파티 대응 시나리오가 포함된 정기적인 훈련을 실시합니다.
   - 공급업체 에스컬레이션 정보를 체계적으로 문서화하고 쉽게 액세스할 수 있도록 합니다.
7. 에스컬레이션 계획 교육 및 연습: 에스컬레이션 프로세스에 대해 팀을 교육하고 정기적인 인시던트 대응 훈련 또는 게임 데이를 실시합니다. Enterprise Support 고객은 Incident Management 워크숍을 요청할 수 있습니다.
8. 지속적인 개선: 에스컬레이션 경로의 효과를 정기적으로 검토합니다. 인시던트 사후 분석 및 지속적인 피드백을 통해 학습한 교훈을 기반으로 프로세스를 업데이트합니다.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- OPS08-BP04 실행 가능한 알림 생성
- OPS10-BP02 알림별 프로세스 마련
- OPS11-BP02 인시던트 사후 분석 수행

관련 문서:

- AWS Systems Manager Incident Manager Escalation Plans
- Working with on-call schedules in Incident Manager
- 런북 생성 및 관리
- Temporary elevated access management with AWS IAM Identity Center
- Atlassian - Escalation policies for effective incident management

OPS10-BP05 서비스에 영향을 미치는 이벤트에 대한 고객 커뮤니케이션 계획 정의

서비스에 영향을 미치는 이벤트 발생 시 효과적인 커뮤니케이션은 고객과의 신뢰와 투명성을 유지하는 데 매우 중요합니다. 체계적으로 정의된 커뮤니케이션 계획을 통해 조직은 인시던트 발생 시 내부 및 외부에서 정보를 빠르고 명확하게 공유할 수 있습니다.

원하는 성과:

- 서비스에 영향을 미치는 이벤트 발생 시 고객과 이해관계자에게 효과적으로 정보를 제공하는 탄탄한 커뮤니케이션 계획.
- 신뢰를 구축하고 고객의 불안을 줄이기 위한 커뮤니케이션의 투명성.
- 서비스에 영향을 미치는 이벤트가 고객 경험 및 비즈니스 운영에 미치는 영향 최소화.

일반적인 안티 패턴:

- 부적절하거나 지연된 커뮤니케이션은 고객 혼란과 불만족으로 이어집니다.
- 지나치게 기술적이거나 모호한 메시지는 실제로 사용자에게 미치는 영향을 전달하지 못합니다.
- 사전 정의된 커뮤니케이션 전략이 없기 때문에 메시지가 일관되지 않고 반응성이 떨어집니다.

이 모범 사례 확립의 이점:

- 적극적이고 명확한 커뮤니케이션을 통해 고객 신뢰와 만족도를 개선합니다.
- 고객 문제를 선제적으로 해결하여 지원 팀의 부담을 완화합니다.
- 인시던트를 효과적으로 관리하고 복구하는 능력을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

서비스에 영향을 미치는 이벤트에 대한 포괄적인 커뮤니케이션 계획을 수립하려면 적절한 채널 선택부터 메시지 작성 및 어조 조정에 이르기까지 다양한 측면이 필요합니다. 계획은 조정 가능하고 확장 가능하며 다양한 중단 시나리오에 적합해야 합니다.

구현 단계

1. 역할과 책임 정의:
   - 주요 인시던트 관리자를 지정하여 인시던트 대응 활동을 감독합니다.
   - 모든 외부 및 내부 커뮤니케이션을 조정할 책임이 있는 커뮤니케이션 관리자를 지정합니다.
   - 지원 티켓을 통해 일관된 커뮤니케이션이 가능하도록 지원 관리자를 포함합니다.
2. 커뮤니케이션 채널 파악: 워크플레이스 채팅, 이메일, SMS, 소셜 미디어, 앱 내 알림, 상태 페이지와 같은 채널을 선택합니다. 이러한 채널은 복원력이 있어야 하며 서비스에 영향을 미치는 이벤트 발생 시 독립적으로 운영될 수 있어야 합니다.
3. 고객에게 빠르고 명확하게 정기적으로 커뮤니케이션 전달:
   - 단순성과 필수 세부 정보를 강조하여 다양한 서비스 장애 시나리오에 대한 템플릿을 개발합니다. 템플릿에 서비스 장애, 예상 해결 시간 및 영향에 대한 정보를 포함합니다.
   - Amazon Pinpoint를 사용하여 푸시 알림, 인앱 알림, 이메일, 문자 메시지, 음성 메시지 및 사용자 지정 채널을 통한 메시지를 사용하여 고객에게 알립니다.
   - Amazon Simple Notification Service(SNS)를 사용하여 프로그래밍 방식으로 또는 이메일, 모바일 푸시 알림 및 문자 메시지를 통해 구독자에게 알립니다.
   - Amazon CloudWatch 대시보드를 공개적으로 공유하여 대시보드를 통해 상태를 전달합니다.
   - 소셜 미디어 참여 장려:
     - 소셜 미디어를 적극적으로 모니터링하여 고객의 분위기를 파악합니다.
     - 소셜 미디어 플랫폼에 게시하여 공개 업데이트 및 커뮤니티 참여를 확인합니다.
     - 일관되고 명확한 소셜 미디어 커뮤니케이션을 위한 템플릿을 준비합니다.

4. 내부 커뮤니케이션 조정: 팀 조정 및 커뮤니케이션을 위해 채팅 애플리케이션 내 Amazon Q Developer 같은 도구를 사용하여 내부 프로토콜을 구현합니다. CloudWatch 대시보드를 사용하여 상태를 전달합니다.
5. 전용 도구 및 서비스를 사용하여 커뮤니케이션 조율:
   - 채팅 애플리케이션 내 Amazon Q Developer와 함께 AWS Systems Manager Incident Manager를 사용하여 인시던트 발생 시 실시간 내부 커뮤니케이션 및 조정을 위한 전용 채팅 채널을 설정합니다.
   - AWS Systems Manager Incident Manager 런북을 사용하여 인시던트 발생 시 Amazon Pinpoint, Amazon SNS 또는 소셜 미디어 플랫폼과 같은 서드파티 도구를 통해 고객 알림을 자동화합니다.
   - 런북에 승인 워크플로를 통합하여 선택적으로 전송 전에 모든 외부 커뮤니케이션을 검토하고 승인할 수 있습니다.
6. 연습 및 개선:
   - 커뮤니케이션 도구 및 전략의 사용에 대한 교육을 실시합니다. 팀이 인시던트 발생 시 시의적절하게 결정을 내릴 수 있도록 지원합니다.
   - 정기적인 훈련이나 게임 데이를 통해 커뮤니케이션 계획을 테스트합니다. 이 테스트를 사용하여 메시징을 구체화하고 채널의 효과를 평가합니다.
   - 피드백 메커니즘을 구현하여 인시던트 발생 시 커뮤니케이션 효과를 평가합니다. 피드백과 변화하는 요구 사항을 기반으로 커뮤니케이션 계획을 지속적으로 발전시킵니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- OPS07-BP03 런북을 사용한 절차 수행
- OPS10-BP06 대시보드를 통해 상태 전달
- OPS11-BP02 인시던트 사후 분석 수행

관련 문서:

- Atlassian - Incident communication best practices
- Atlassian - How to write a good status update
- PagerDuty - A Guide to Incident Communications

관련 비디오:

- Atlassian - Create your own incident communication plan: Incident templates

관련 예제:

- AWS Health Dashboard

OPS10-BP06 대시보드를 통해 상태 전달

대시보드를 전략적 도구로 사용하여 내부 기술팀, 경영진, 고객 등 다양한 대상에게 실시간 운영 상태 및 주요 지표를 전달합니다. 이러한 대시보드는 시스템 상태 및 비즈니스 성과를 중앙 집중식으로 시각적으로 표현하여 투명성과 의사 결정 효율성을 향상시킵니다.

원하는 성과:

- 대시보드는 다양한 이해관계자와 관련된 시스템 및 비즈니스 지표에 대한 포괄적인 보기를 제공합니다.
- 이해관계자가 운영 정보에 사전에 액세스할 수 있으므로 빈번히 상태를 요청하지 않아도 됩니다.
- 정상적인 운영 및 인시던트 발생 시 실시간 의사 결정이 향상됩니다.

일반적인 안티 패턴:

- 엔지니어가 인시던트 관리 통화에 참여하려면 빠른 진행을 위해 상태 업데이트가 필요합니다.
- 관리를 위해 수동 보고에 의존하기 때문에 지연이 발생하고 정확성이 떨어질 수 있습니다.
- 인시던트 발생 시 운영 팀은 상태 업데이트를 위해 빈번히 업무를 중단해야 합니다.

이 모범 사례 확립의 이점:

- 이해관계자가 중요한 정보에 즉시 액세스할 수 있도록 하여 정보에 입각한 의사 결정을 촉진합니다.
- 수동 보고 및 빈번한 상태 조회를 최소화하여 운영 비효율성을 완화합니다.
- 시스템 성능 및 비즈니스 지표에 대한 실시간 가시성을 통해 투명성과 신뢰도를 높입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

대시보드는 시스템 및 비즈니스 지표의 상태를 효과적으로 전달하며 다양한 대상 그룹의 요구에 맞게 조정할 수 있습니다. Amazon CloudWatch 대시보드 및 Amazon QuickSight와 같은 도구를 사용하면 시스템 모니터링 및 비즈니스 인텔리전스를 위한 대화형 실시간 대시보드를 만들 수 있습니다.

구현 단계

1. 이해관계자의 요구 사항 파악: 기술팀, 경영진, 고객 등 다양한 대상 그룹의 특정 정보 요구 사항을 결정합니다.
2. 적절한 도구 선택: 시스템 모니터링을 위한 Amazon CloudWatch 대시보드 및 대화형 비즈니스 인텔리전스를 위한 Amazon QuickSight와 같은 적절한 도구를 선택합니다. AWS Health는 AWS Health Dashboard에서 즉시 사용 가능한 환경을 제공하며, Amazon EventBridge 또는 AWS Health API를 통해 상태 이벤트를 사용하여 자체 대시보드를 보강할 수도 있습니다.
3. 효과적인 대시보드 설계:
   - 관련 지표와 KPI를 명확하게 제시하여 이해할 수 있고 실행 가능한 방식으로 대시보드를 설계합니다.
   - 필요에 따라 시스템 수준 및 비즈니스 수준 보기를 통합합니다.
   - 상위 수준(광범위한 개요용) 및 하위 수준(세부 분석용) 대시보드를 모두 포함합니다.
   - 대시보드 내에 자동 경보를 통합하여 중요한 문제를 강조 표시합니다.
   - 대시보드에 중요한 지표 임곗값 및 목표를 주석으로 추가하여 즉시 확인할 수 있습니다.
4. 데이터 소스 통합:
   - Amazon CloudWatch를 사용하여 다양한 AWS 서비스의 지표를 집계 및 표시하고 다른 데이터 소스의 지표를 쿼리하여 시스템의 상태 및 비즈니스 지표에 대한 통합된 보기를 생성합니다.
   - CloudWatch 로그 인사이트와 같은 기능을 사용하여 다양한 애플리케이션 및 서비스의 로그 데이터를 쿼리하고 시각화합니다.
   - AWS Health API 또는 Amazon EventBridge의 AWS Health 이벤트를 통해 AWS Health 이벤트를 사용하여 AWS 서비스의 운영 상태와 확인된 운영 문제에 대한 정보를 얻습니다.
5. 셀프 서비스 액세스 제공:
   - 셀프 서비스 정보에 액세스하도록 대시보드 공유 기능을 사용하여 관련 이해관계자와 CloudWatch 대시보드를 공유합니다.
   - 대시보드에 쉽게 액세스할 수 있도록 하고 실시간 최신 정보를 제공합니다.
6. 정기적으로 업데이트 및 개선:
   - 진화하는 비즈니스 요구 사항 및 이해관계자 피드백에 맞춰 대시보드를 지속적으로 업데이트하고 수정합니다.

- 대시보드를 정기적으로 검토하여 필요한 정보를 전달하는 데 적합하고 효과적인지 확인합니다.

리소스

관련 모범 사례:

- OPS08-BP05 대시보드 만들기

관련 문서:

- 운영 가시성을 위한 대시보드 구축
- Amazon CloudWatch 대시보드 사용
- 대시보드 변수를 사용하여 유연한 대시보드 생성
- CloudWatch 대시보드 공유
- 다른 데이터 소스의 쿼리 지표
- CloudWatch 대시보드에 사용자 지정 위젯 추가

관련 예제:

- One Observability 워크숍 - 대시보드

OPS10-BP07 이벤트 대응 자동화

이벤트 대응 자동화는 빠르고 일관되며 오류 없는 운영 처리를 위한 핵심 비결입니다. 간소화된 프로세스를 만들고 도구를 사용하여 이벤트를 자동으로 관리하고 대응하여 수동 개입을 최소화하고 운영 효율성을 개선하세요.

원하는 성과:

- 자동화를 통한 인적 오류 감소 및 해결 시간 단축.
- 일관되고 신뢰할 수 있는 운영 이벤트 처리.
- 운영 효율성 및 시스템 신뢰성 향상.

일반적인 안티 패턴:

- 수동으로 이벤트를 처리하면 지연과 오류가 발생합니다.

- 반복적이고 중요한 작업에서 자동화가 간과됩니다.
- 반복적인 수동 작업으로 인해 알림에 대한 피로감이 쌓이고 중요한 문제가 누락됩니다.

이 모범 사례 확립의 이점:

- 이벤트 대응 가속화를 통한 시스템 가동 중지 감소.
- 자동화되고 일관된 이벤트 처리를 통한 신뢰할 수 있는 운영.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

자동화를 통합하여 효율적인 운영 워크플로를 만들고 수동 개입을 최소화합니다.

구현 단계

1. 자동화 기회 파악: 문제 해결, 티켓 강화, 용량 관리, 규모 조정, 배포 및 테스트와 같은 자동화를 위한 반복 작업을 결정합니다.
2. 자동화 프롬프트 확인:
   - 이 단계에서는 Amazon CloudWatch 작업을 사용하여 자동 응답을 개시하는 특정 조건이나 지표를 평가 및 정의합니다.
   - Amazon EventBridge를 사용하여 AWS 서비스, 사용자 지정 워크로드, SaaS 애플리케이션의 이벤트에 응답합니다.
   - AWS 리소스에서 특정 로그 항목, 성과 지표 임곗값 또는 상태 변경 등의 시작 이벤트를 고려해 보세요.
3. 이벤트 기반 자동화 구현:
   - AWS Systems Manager 자동화 런북을 사용하여 유지 관리, 배포 및 수정 작업을 간소화합니다.
   - Incident Manager에서 인시던트를 생성하면 AWS 관련 리소스에 대한 세부 정보를 자동으로 수집하고 인시던트에 추가할 수 있습니다.
   - Quota Monitor for AWS를 사용하여 할당량을 사전에 모니터링합니다.
   - 가용성과 성능을 유지하기 위해 AWS Auto Scaling을 사용하여 용량을 자동으로 조정합니다.
   - Amazon CodeCatalyst를 사용하여 개발 파이프라인을 자동화합니다.
   - 가상 모니터링을 사용하여 엔드포인트 및 API를 스모크 테스트하거나 지속적으로 모니터링합니다.
4. 자동화를 통한 위험 완화 수행:

- 위험을 신속하게 해결하기 위해 자동화된 보안 대응을 구현합니다.
- AWS Systems Manager State Manager를 사용하여 구성 편차를 줄입니다.
- AWS Config 규칙을 사용하여 규정 미준수 리소스를 수정합니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- OPS08-BP04 실행 가능한 알림 생성
- OPS10-BP02 알림별 프로세스 마련

관련 문서:

- Using Systems Manager Automation runbooks with Incident Manager
- Creating incidents in Incident Manager
- AWS Service Quotas
- Monitor resource usage and send notifications when approaching quotas
- AWS Auto Scaling
- What is Amazon CodeCatalyst?
- Amazon CloudWatch 경보 사용
- Amazon CloudWatch 경보 작업 사용
- Remediating Noncompliant Resources with AWS Config 규칙
- Creating metrics from log events using filters
- AWS Systems Manager State Manager

관련 비디오:

- Create Automation Runbooks with AWS Systems Manager
- How to automate IT Operations on AWS
- AWS Security Hub automation rules
- Start your software project fast with Amazon CodeCatalyst blueprints

관련 예제:

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- One Observability 워크숍
- Respond to incidents using Incident Manager

개선

질문

- OPS 11. 귀사는 어떻게 운영을 지속적으로 개선하고 있나요?

OPS 11. 귀사는 어떻게 운영을 지속적으로 개선하고 있나요?

시간과 리소스를 할애하여 점진적 개선을 거의 지속적으로 수행하면 운영의 효과와 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 학습한 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 평가하세요. 의도적인 워크로드 검토를 자주 실시하세요. 소프트웨어 개발 단계에서 개선 기회의 우선순위를 지정하세요.

원하는 성과:

- 아키텍처 모범 사례를 기준으로 워크로드를 자주 분석합니다.
- 소프트웨어 개발 프로세스의 기능과 개선 기회에 동등한 우선순위를 부여합니다.

일반적인 안티 패턴:

- 몇 년 전에 배포된 이후 워크로드에 대한 아키텍처 검토를 수행한 적이 없습니다.
- 개선 기회에 더 낮은 우선순위를 부여합니다. 새로운 기능에 비해 개선 기회를 계속 뒷전으로 두고 있습니다.
- 조직에 맞춰 모범 사례를 수정하기 위한 표준이 없습니다.

이 모범 사례 확립의 이점:

- 워크로드가 최신 아키텍처 모범 사례에 맞춰 유지됩니다.
- 의도적인 방식으로 워크로드를 발전시킵니다.
- 조직의 모범 사례를 활용하여 모든 워크로드를 개선할 수 있습니다.
- 개별적으로는 작지만 모이면 큰 영향을 미치는 이익을 얻을 수 있어 효율성의 심도가 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드에 대한 아키텍처 검토를 자주 수행하세요. 내부 및 외부 모범 사례를 사용하여 워크로드를 평가하고 개선 기회를 식별하세요. 소프트웨어 개발 단계에서 개선 기회의 우선순위를 지정하세요.

구현 단계

1. 합의된 빈도로 프로덕션 워크로드에 대한 정기적인 아키텍처 검토를 수행합니다. AWS 관련 모범 사례를 포함한 문서화된 아키텍처 표준을 사용합니다.
   a. 이러한 검토에는 내부적으로 정의된 표준을 사용합니다. 내부 표준이 없는 경우에는 AWS Well-Architected Framework를 사용합니다.
   b. AWS Well-Architected Tool을 사용하여 내부 모범 사례의 사용자 지정 렌즈를 생성하고 아키텍처 검토를 수행합니다.
   c. AWS Solution Architect 또는 Technical Account Manager에게 문의하여 워크로드에 대한 가이드식 Well-Architected Framework 검토를 수행합니다.
2. 소프트웨어 개발 프로세스 중 검토 과정에서 식별된 개선 기회를 우선시합니다.

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
- OPS11-BP08 파악한 내용 문서화 및 공유
- OPS04 - 관찰성 구현

관련 문서:

- AWS Well-Architected Tool - Custom lenses
- AWS Well-Architected 백서 - 검토 프로세스
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

관련 비디오:

- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

관련 예제:

- 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.

원하는 성과:

- 인시던트 사후 분석을 포함하는 인시던트 관리 프로세스를 수립했습니다.
- 이벤트에 대한 데이터를 수집하기 위한 관찰성 계획이 마련되어 있습니다.
- 이 데이터를 통해 인시던트 사후 분석 프로세스를 지원하는 지표를 이해하고 수집할 수 있습니다.

- 인시던트로부터 교훈을 얻어 미래의 결과를 개선합니다.

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 확립의 이점:

- 인시던트에 기여한 구성 요소, 조건, 작업 및 이벤트를 결정하기 위해 사전 정의된 프로세스를 사용하면 개선 기회를 파악할 수 있습니다.
- 인시던트 사후 분석에서 얻은 데이터를 사용하여 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

발생 요인을 확인하는 프로세스를 사용합니다. 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. 인시던트의 근본 원인을 적절하게 전달하고 대상 고객에 맞게 커뮤니케이션을 조정합니다. 조직 내에서 학습한 내용을 공개적으로 공유합니다.

구현 단계

1. 배포 변경, 구성 변경, 인시던트 시작 시간, 경보 시간, 참여 시간, 완화 시작 시간, 인시던트 해결 시간과 같은 지표를 수집합니다.
2. 인시던트 발생 상황을 파악하기 위해 타임라인에 주요 시점을 표시합니다.
3. 다음과 같이 질문하세요.
   a. 감지 시간을 단축할 수 있나요?
   b. 인시던트를 더 빨리 감지할 수 있는 지표 및 경보 업데이트가 있습니까?

c. 진단 시간을 개선할 수 있나요?
d. 대응 계획이나 에스컬레이션 계획에 올바른 대응 담당자를 더 빨리 투입할 수 있는 업데이트가 있습니까?
e. 완화 시간을 단축할 수 있나요?
f. 추가하거나 개선할 수 있는 런북 또는 플레이북 단계가 있나요?
g. 향후 인시던트 발생을 방지할 수 있나요?

4. 체크리스트와 작업을 생성합니다. 모든 작업을 추적하고 전달합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- OPS11-BP01 지속적인 개선을 위한 프로세스 마련
- OPS 4 - 관찰성 구현

관련 문서:

- Performing a post-incident analysis in Incident Manager
- 운영 준비 상태 검토

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 루프는 즉각적 피드백 및 후행 분석과 같은 두 가지 범주로 구분됩니다. 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 성과: 즉각적 피드백 및 후행 분석을 사용하여 개선을 추진할 수 있습니다. 사용자 및 팀원의 피드백을 얻을 수 있는 메커니즘이 있습니다. 후행 분석은 개선을 추진하는 추세를 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.
- 추세를 사용하여 개선 기회를 파악할 수 있습니다.
- 후행 분석을 통해 워크로드 및 운영에 대한 투자를 검증할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 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 기간에 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.
  - AWS Systems Manager OpsCenter를 사용하여 이러한 개선 사항을 OpsIms로 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 이해관계자를 모읍니다.
   - 화이트보드나 스프레드시트에 중지, 시작 및 유지라는 세 개의 열을 만듭니다.
     - 중지는 팀에서 수행을 중지하고자 하는 항목입니다.
     - 시작은 시작하고자 하는 아이디어입니다.
     - 유지는 계속하고자 하는 항목입니다.
   - 회의실을 한 바퀴 돌며 이해관계자들의 피드백을 수렴합니다.
   - 피드백의 우선순위를 정합니다. 모든 시작 또는 유지 항목에 대한 활동 및 이해관계자를 할당합니다.
   - 소프트웨어 개발 프로세스에 해당 활동을 추가하고 개선 작업을 수행할 때 이해관계자에게 상태 업데이트를 전달합니다.

구현 계획의 작업 수준: 중간. 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 이해관계자는 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 추세와 영역을 파악합니다.

관련 문서:

- 7 Pitfalls to Avoid When Building a CCOE
- Atlassian Team Playbook - Retrospectives
- Email Definitions: Feedback Loops
- Establishing Feedback Loops Based on the AWS Well-Architected Framework Review
- IBM Garage Methodology - Hold a retrospective
- Investopedia – The PDCS Cycle
- Maximizing Developer Effectiveness by Tim Cochran
- Operations Readiness Reviews (ORR) Whitepaper - Iteration
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

관련 비디오:

- Building Effective Customer Feedback Loops

관련 예제:

- Astuto - Open source customer feedback tool
- AWS 솔루션 - QnABot on AWS
- Fider - A platform to organize customer feedback

관련 서비스:

- AWS Systems Manager OpsCenter

OPS11-BP04 지식 관리 수행

지식 관리는 팀원이 업무 수행에 필요한 정보를 찾는 데 도움이 됩니다. 학습하는 조직에서는 개인에게 유용한 정보가 자유롭게 공유됩니다. 정보가 찾거나 검색할 수 있습니다. 정보가 정확하며 최신 상태입

니다. 새로운 정보를 생성하고, 기존 정보를 업데이트하며, 오래된 정보를 보관하는 메커니즘이 있습니다. 지식 관리 플랫폼의 가장 일반적인 예로는 Wiki와 같은 콘텐츠 관리 시스템을 들 수 있습니다.

원하는 성과:

- 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. 팀원은 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 팀원이 더 빨리 온보딩됩니다.
- 정보는 시의적절하고 정확하며 실행 가능합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 이해관계자로부터 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. 이해관계자와 협력하여 추가되는 지식의 우선순위를 정합니다.
4. 주기적으로 이해관계자와 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 학습한 내용 문서화 및 공유 - 지식 관리는 학습한 내용에 대한 정보 공유를 용이하게 합니다.

관련 문서:

- Atlassian - Knowledge Management

관련 예제:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 개선 추진 요인 정의

개선 기회를 평가하고 우선순위를 지정할 수 있도록 데이터와 피드백 루프를 바탕으로 개선 추진 요인을 파악합니다. 시스템과 프로세스의 개선 기회를 탐색하고 적절한 경우 자동화합니다.

원하는 성과:

- 환경 전반에서 데이터를 추적합니다.
- 이벤트 및 활동과 비즈니스 성과의 상관관계를 파악합니다.
- 환경과 시스템을 비교하고 대조할 수 있습니다.
- 배포 및 결과에 대한 자세한 활동 기록을 유지 관리합니다.
- 보안 태세를 뒷받침하기 위해 데이터를 수집합니다.

일반적인 안티 패턴:

- 전체 환경에서 데이터를 수집하지만 이벤트와 활동의 상관관계를 파악하지는 않습니다.
- 환경 전체에서 상세한 데이터를 수집하여 Amazon CloudWatch 및 AWS CloudTrail 활동과 비용이 많이 발생합니다. 그러나 이 데이터를 의미 있게 사용하지는 않습니다.
- 개선 추진 요인을 정의할 때 비즈니스 성과를 고려하지 않습니다.
- 새 기능의 효과를 평가하지 않습니다.

이 모범 사례 확립의 이점:

- 개선 기준을 결정하여 이벤트 기반 동기 또는 감정적 에너지 소모의 영향을 최소화합니다.
- 기술 이벤트뿐만 아니라 비즈니스 이벤트에도 대응합니다.
- 환경을 평가하여 개선이 필요한 영역을 식별합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

- 개선 추진 요인 파악: 원하는 성과가 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 취약성을 평가합니다. 규모 조정 옵션을 추적하고 최적화 기회를 모색합니다.
    - AWS 최신 보안 공지
    - AWS Trusted Advisor
    - Cloud Intelligence Dashboards
  - 규정 준수 요건: 개선 기회를 검토할 때 규정과 정책 준수 상태를 유지하거나 서드파티의 지원을 계속 받으려는 데 필요한 업데이트와 변경 사항을 평가합니다.

- AWS 규정 준수
- AWS 규정 준수 프로그램
- AWS 규정 준수 최신 뉴스

리소스

관련 모범 사례:

- OPS01 조직 우선순위
- OPS02 관계 및 소유권
- OPS04-BP01 핵심 성과 지표 파악
- OPS08 워크로드 관찰성 활용
- OPS09 운영 상태 파악
- OPS11-BP03 피드백 루프 구현

관련 문서:

- Amazon Athena
- QuickSight
- AWS 규정 준수
- AWS 규정 준수 최신 뉴스
- AWS 규정 준수 프로그램
- AWS Glue
- AWS 최신 보안 공지
- AWS Trusted Advisor
- Export your log data to Amazon S3
- AWS의 새로운 소식
- 고객 중심 혁신의 필요성
- Digital Transformation: Hype or a Strategic Necessity?

관련 비디오

- AWS re:Invent 2023 - Improve operational efficiency and resilience with 지원 (SUP310)

OPS11-BP06 인사이트 검증

여러 부문의 팀 및 비즈니스 소유자와 함께 분석 결과와 응답을 검토합니다. 이러한 검토에서는 개선 가능성을 공통적으로 파악하고, 추가적인 영향을 확인하며, 조치 과정을 결정할 수 있습니다. 필요에 따라 대응 내용을 조정합니다.

원하는 성과:

- 정기적으로 비즈니스 소유자와 함께 인사이트를 검토합니다. 비즈니스 소유자는 새로 얻은 인사이트에 대한 추가 컨텍스트를 제공합니다.
- 인사이트를 검토하고 기술 부문의 동료에게 피드백을 요청하며 팀 간에 학습한 내용을 공유합니다.
- 다른 기술 및 비즈니스 팀이 검토할 수 있도록 데이터와 인사이트를 게시합니다. 학습한 내용을 다른 부서의 새로운 업무 방식에 반영합니다.
- 시니어 리더와 함께 새로운 인사이트를 요약하고 검토합니다. 시니어 리더는 새로운 인사이트를 사용하여 전략을 정의합니다.

일반적인 안티 패턴:

- 새 기능을 릴리스합니다. 이 기능은 고객 행동 중 일부를 변화시킵니다. 관찰성에 이러한 변경 사항을 고려하지 않습니다. 이러한 변경으로 인한 이점을 수량화하지 않습니다.
- 새 업데이트를 푸시하고 CDN 새로 고침을 소홀히 합니다. CDN 캐시가 최신 릴리스와 더 이상 호환되지 않습니다. 오류가 있는 요청의 비율을 측정합니다. 모든 사용자가 백엔드 서버와 통신할 때 HTTP 400 오류를 보고합니다. 클라이언트 오류를 조사한 결과 차원을 잘못 측정했기 때문에 시간이 낭비되었다는 것을 알게 됩니다.
- 서비스 수준에 관한 계약(SLA)에는 가동 시간이 99.9%라고 명시되어 있으며 Recovery Point Objective는 4시간입니다. 서비스 소유자는 시스템 가동 중지 시간이 전혀 없다고 주장합니다. 비용이 많이 들고 복잡한 복제 솔루션을 구축하여 시간과 비용이 낭비됩니다.

이 모범 사례 확립의 이점:

- 비즈니스 소유자 및 주제 전문가와 함께 인사이트를 검증하면 공통된 이해를 확립하고 개선에 더 효과적으로 반영할 수 있습니다.
- 숨겨진 문제를 발견하고 이를 향후 의사 결정에 반영합니다.
- 기술적 성과에서 비즈니스 성과로 초점이 옮겨집니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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.

리소스

관련 모범 사례:

- OPS01-BP06 이점과 위험을 관리하면서 장단점 평가
- OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임
- OPS11-BP03 피드백 루프 구현

관련 문서:

- Designing a Cloud Center of Excellence (CCOE)

관련 비디오:

- Building observability to increase resiliency

OPS11-BP07 운영 지표 검토 수행

다양한 실무 영역의 여러 팀원과 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 개선 기회를 모색해야 합니다.

원하는 성과:

- 비즈니스에 영향을 미치는 지표 자주 검토
- 관찰성 기능을 통해 이상 징후 감지 및 검토
- 데이터를 사용하여 비즈니스 성과 및 목표 지원

일반적인 안티 패턴:

- 유지 관리 기간으로 인해 중요한 소매 프로모션이 중단됩니다. 기업에서는 비즈니스에 영향을 미치는 다른 이벤트가 있는 경우 지연될 수 있는 표준 유지 관리 기간이 있음을 모릅니다.
- 조직에서 오래된 라이브러리를 일반적으로 사용하기 때문에 운영 중단이 오래 지속되었습니다. 이후 지원되는 라이브러리로 마이그레이션했습니다. 조직의 다른 팀은 위험에 처해 있다는 것을 알지 못합니다.
- 고객 SLA 달성을 정기적으로 검토하지 않습니다. 고객 SLA를 충족하지 못하는 추세입니다. 고객 SLA를 충족하지 못할 경우 재정적 징벌이 부과될 수 있습니다.

이 모범 사례 확립의 이점:

- 정기적으로 만나 운영 지표, 이벤트 및 인시던트를 검토하면 팀 간에 공통된 이해를 유지할 수 있습니다.
- 팀은 정기적으로 회의를 통해 지표와 인시던트를 검토하며, 이를 통해 위험에 대한 조치를 취하고 고객 SLA를 인식할 수 있습니다.
- 파악한 내용을 공유하여 비즈니스 성과에 대한 우선순위 지정 및 목표 개선을 위한 데이터를 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

- 다양한 실무 영역의 여러 팀원과 함께 운영 지표 후행 분석을 정기적으로 수행합니다.
- 실무 팀, 개발 팀, 운영 팀 등의 이해관계자와 함께 즉각적인 피드백 및 후행 분석에서 발견된 사항을 확인하고 파악한 내용을 공유합니다.
- 그리고 이러한 인사이트를 활용하여 개선 기회와 진행 가능한 조치 과정을 확인합니다.

리소스

관련 모범 사례:

- OPS08-BP05 대시보드 만들기
- OPS09-BP03 운영 지표 검토 및 개선 우선순위 지정
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용

관련 문서:

- Amazon CloudWatch
- Amazon CloudWatch 지표 및 차원 참조
- 사용자 지정 지표 게시
- Amazon CloudWatch 지표 사용
- Dashboards and visualizations with CloudWatch

OPS11-BP08 학습한 내용 문서화 및 공유

운영 활동 과정에서 파악한 내용을 문서화하고 공유하여 내부적으로 그리고 여러 팀 간에 사용할 수 있도록 합니다. 조직 전체에서 관련 이점을 더욱 효율적으로 활용하려면 팀에서 학습한 내용을 공유해야 합니다. 피할 수 있는 오류를 방지하고 개발 작업을 쉽게 수행하기 위해 정보와 리소스를 공유하고 원하는 기능을 제공하는 데 집중하세요.

AWS Identity and Access Management(IAM)를 사용하여 계정 내에서와 계정 간에 공유할 리소스 액세스를 제어할 수 있는 권한을 정의합니다.

원하는 성과:

- 버전 관리 리포지토리를 사용하여 애플리케이션 라이브러리, 스크립팅된 절차, 절차 설명서 및 기타 시스템 설명서를 공유합니다.
- 인프라 표준을 AWS CloudFormation 템플릿(버전 관리됨)으로 공유합니다.
- 팀 전체에서 학습한 내용을 검토합니다.

일반적인 안티 패턴:

- 조직에서 일반적으로 버그가 있는 라이브러리를 사용하기 때문에 운영 중단이 오래 지속되었습니다. 이후 신뢰할 수 있는 라이브러리로 마이그레이션했습니다. 조직의 다른 팀들은 그들이 위험에 처해 있다는 것을 알지 못합니다. 아무도 이 라이브러리에 대한 경험을 문서화하고 공유하지 않으며 위험을 인식하지 못합니다.
- 내부적으로 공유된 마이크로서비스에서 세션 중단을 일으키는 엣지 사례를 발견했습니다. 이 엣지 사례를 방지하기 위해 서비스에 대한 직접 호출을 업데이트했습니다. 조직의 다른 팀은 위험에 처해 있다는 것을 알지 못합니다.
- 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 모범 사례 확립의 이점: 개선을 지원하고 경험의 이점을 극대화하기 위해 학습한 교훈을 공유합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

- 학습한 내용 문서화 및 공유: 운영 활동을 통해 파악한 내용과 후행 분석 결과를 문서화하는 절차를 마련하여 다른 팀에서도 사용할 수 있도록 합니다.
- 학습한 내용 공유: 학습한 내용 및 관련 아티팩트를 여러 팀에서 공유하는 절차를 마련합니다. 예를 들어, 접속 가능한 Wiki를 통해 새로워진 절차, 지침, 거버넌스 및 모범 사례를 공유합니다. 스크립트, 코드 및 라이브러리는 공동 리포지토리를 통해 공유할 수 있습니다.
  - AWS re:Post Private을 지식 서비스로 활용하여 조직 내 협업 및 지식 공유를 간소화합니다.

리소스

관련 모범 사례:

- OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임
- OPS05-BP01 버전 관리 사용
- OPS05-BP06 설계 표준 공유
- OPS11-BP03 피드백 루프 구현
- OPS11-BP07 운영 지표 검토 수행

관련 문서:

- AWS re:Post Private을 사용하여 협업을 강화하고 클라우드 관련 지식을 안전하게 공유
- Reduce project delays with a docs-as-code solution

관련 비디오:

- AWS re:Invent 2,023 - Collaborate within your company and with AWS using AWS re:Post Private
- 지원s You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 개선을 위한 시간 할애

프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

원하는 성과:

- 실험과 테스트의 위험, 작업량 및 비용을 줄일 수 있도록 환경의 임시 복제본을 생성합니다.
- 이렇게 복제된 환경을 사용하여 분석의 결론을 테스트하고, 실험을 진행하며, 계획된 향상 내용을 개발 및 테스트할 수 있습니다.
- 게임 데이를 운영하고 결함 주입 서비스(FIS)를 통해 팀이 프로덕션과 유사한 환경에서 실험을 실행하는 데 필요한 제어 및 가드레일을 제공합니다.

일반적인 안티 패턴:

- 애플리케이션 서버에 알려진 성능 문제가 있습니다. 이는 계획된 모든 기능 구현 뒤의 백로그에 추가됩니다. 추가되는 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.
- 운영 승인이 완료되었으며 운영 사례를 다시 테스트하지 않습니다.

이 모범 사례 확립의 이점: 프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

- 개선을 위한 시간 할애: 프로세스 내에서 전담 리소스와 시간을 할애하여 점진적 개선을 지속적으로 수행합니다.
- 변경 사항을 적용하여 결과를 개선하고, 평가를 통하여 성공 여부를 확정합니다.
- 결과가 목표에 미치지 못하지만 여전히 개선을 우선해야 한다면 다른 대안을 찾아서 진행합니다.
- 게임 데이 내내 프로덕션 워크로드를 시뮬레이션하고 이러한 시뮬레이션에서 파악한 내용을 활용하여 개선합니다.

리소스

관련 모범 사례:

- OPS05-BP08 여러 환경 사용

관련 비디오:

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

보안

보안 원칙에는 클라우드 기술을 활용하여 보안을 강화하고 데이터, 시스템 및 자산을 보호하는 능력이 포함됩니다. 구현에 대한 권장 가이드는 보안 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 보안 기초
- 자격 증명 및 액세스 관리
- 탐지
- 인프라 보호
- 데이터 보호
- 인시던트 대응
- 애플리케이션 보안

보안 기초

질문

- SEC 1. 워크로드는 어떻게 안전하게 운영하나요?

SEC 1. 워크로드는 어떻게 안전하게 운영하나요?

워크로드를 안전하게 운영하려면 모든 보안 영역에 중요한 모범 사례를 적용해야 합니다. 운영 우수성에 대해 조직 및 워크로드 수준에서 정의한 요구 사항 및 프로세스를 모든 영역에 적용하세요. AWS, 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 보안 위협 및 권장 사항에 대한 최신 정보 파악
- SEC01-BP05 보안 관리 범위 축소
- SEC01-BP06 표준 보안 제어의 배포 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 성과: 클라우드 운영, 관련 없는 워크로드 및 환경을 별도의 계정으로 격리하여 클라우드 인프라 전반의 보안을 강화하는 계정 구조.

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 범위가 감소합니다.
- AWS 서비스, 리소스 및 리전에 대한 액세스 권한의 중앙 거버넌스.
- 보안 서비스의 정책 및 중앙 집중식 관리를 통해 클라우드 인프라의 보안을 유지 관리합니다.
- 자동화된 계정 생성 및 유지 관리 프로세스.
- 규정 준수 및 규제 요구 사항에 대한 인프라의 중앙 집중식 감사.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS 계정은 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 지침은Organizing Your AWS Environment Using Multiple Accounts를 참조하세요.

중앙 관리 하에 여러 AWS 계정이 있는 경우 조직 단위(OU) 계층으로 정의된 계층 구조로 계정을 구성해야 합니다. 그런 다음 보안 제어를 구성하고 OU 및 구성원 계정에 적용하여 조직의 구성원 계정에 일관된 예방적인 제어를 설정할 수 있습니다. 보안 제어는 상속되므로 OU 계층 구조의 하위 수준에 있는 구성원 계정이 사용 가능한 권한을 필터링할 수 있습니다. 우수한 설계는 이 상속을 활용하여 각 구성원 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 구성원 계정이 포함되어 있습니다. 서비스 제어 정책(SCP)을 사용하면 조직 관리자가 구성원 계정에 대한 세분화된 예방 제어를 설정할 수 있으며, AWS Config를 사용하면 구성원 계정에 대한 사전 예방 및 탐지 제어를 설정할 수 있습니다. 많은 AWS 서비스가 AWS Organizations와 통합되어 위임된 관리 제어를 제공하고 조직의 모든 구성원 계정 전체의 서비스별 작업을 수행합니다.

AWS Organizations 위에 계층화된 AWS Control Tower에서는 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 제어.
- 주어진 OU 세트에 대해 활성화 또는 비활성화할 수 있는 선택적 제어.
- AWS Control Tower Account Factory는 조직 내에서 미리 승인된 기준과 구성 옵션을 포함하는 계정의 자동화된 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 Towerhttps://docs.aws.amazon.com/controltower/latest/userguide/mandatory-controls.html는 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 계정 및 리전에 배포할 수 있습니다.

리소스

관련 모범 사례:

- SEC02-BP04 중앙 집중식 ID 공급업체 사용

관련 문서:

- AWS Control Tower
- AWS 보안 감사 지침
- IAM 모범 사례
- Use CloudFormation StackSets to provision resources across multiple AWS 계정 and regions
- Organizations FAQ
- AWS Organizations 용어 및 개념
- Best Practices for Service Control Policies in an AWS Organizations Multi-Account Environment
- AWS Account Management 참조 안내서
- Organizing Your AWS Environment Using Multiple Accounts

관련 비디오:

- Enable AWS adoption at scale with automation and governance

- Security Best Practices the Well-Architected Way
- Building and Governing Multiple Accounts using AWS Control Tower
- Enable Control Tower for Existing Organizations

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 자격 증명이 의도치 않게 노출되어 클라우드 환경이 손상될 위험을 줄일 수 있습니다.

원하는 성과: 루트 사용자를 보호하면 루트 사용자 자격 증명 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 자격 증명이 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 계정을 처음 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 자격 증명을 생성하는 것으로 시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 합니다. 계정을 생성할 때 사용한 이메일 주소와 암호를 입력하여 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 자격 증명은 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 계정이 있습니다. 하나의 계정이 관리 계정으로 지정되면 여러 계층의 구성원 계정을 관리 계정 아래 추가할 수 있습니다. 관리 계정의 루트 사용자 보호에 우선순위를 지정한 다음 구성원 계정 루트 사용자의 주소를 기재합니다. 관리 계정의 루트 사용자를 보호하기 위한 전략은 구성원 계정 루트 사용자와 다를 수 있으며, 구성원 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 benchmark version 1.4.0에서 교차 참조됩니다. 이러한 단계 외에도 AWS 계정 및 리소스 보호를 위해 AWS 모범 사례 지침을 참조하세요.

예방 제어

1. 계정의 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 다른 사람과 공유해서는 안 됩니다.
2. 루트 사용자의 액세스 키를 생성하지 않습니다. 액세스 키가 있으면 제거합니다(CIS 1.4).
   a. 루트 사용자에 대한 장기 프로그래밍 방식 자격 증명(액세스 및 보안 암호 키)을 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 자격 증명을 저장해야 하는지 여부를 결정합니다.

   a. AWS Organizations를 사용하여 새 구성원 계정을 생성하는 경우 새 구성원 계정에 대한 루트 사용자의 초기 암호가 사용자에게 노출되지 않는 임의의 값으로 설정됩니다. 필요한 경우 AWS 조직 관리 계정의 암호 재설정 흐름을 사용하여 구성원 계정에 대한 액세스 권한을 얻는 것이 좋습니다.

   b. 독립 실행형 AWS 계정 또는 관리 AWS 조직 계정의 경우 루트 사용자에 대한 자격 증명을 생성하고 안전하게 저장하는 것이 좋습니다. 루트 사용자에 대해 MFA를 사용합니다.

4. AWS 다중 계정 환경에서 구성원 계정 루트 사용자에 대한 예방 제어를 활성화합니다.

   a. 구성원 계정의 경우 루트 사용자에 대한 루트 액세스 키 생성 금지 예방 가드레일 사용을 고려하세요.

   b. 구성원 계정의 경우 루트 사용자로 작업 금지 예방 가드레일 사용을 고려하세요.

5. 루트 사용자에 대한 자격 증명이 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 사용자용 MFA를 구성하려면 가상 MFA 또는 하드웨어 MFA 디바이스를 생성하는 방법에 대한 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.

   - 자격 증명 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 자격 증명의 사용을 감지하는 경보를 생성합니다(CIS 1.7). Amazon GuardDuty에서는 RootCredentialUsage 조사 결과를 통해 루트 사용자 API 자격 증명 사용을 모니터링하고 알림을 제공할 수 있습니다.
- AWS Well-Architected Security Pillar conformance pack for AWS Config에 포함된 탐지 제어를 평가 및 구현합니다. 또는 AWS Control Tower를 사용하는 경우 Control Tower 내에서 사용 가능한 제어 기능이 강력히 권장됩니다.

운영 지침

- 조직에서 루트 사용자 자격 증명에 대한 액세스 권한을 갖는 담당자를 결정합니다.
  - 루트 사용자 액세스 권한을 획득하는 데 필요한 모든 자격 증명 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 자격 증명을 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. `<abuse@amazon.com>`에서 수신된 보안 알림이 있는지 이러한 이메일 받은 편지함을 모니터링합니다. 또한 계정과 연결된 모든 전화번호가 작동하는지 확인합니다.
- 루트 계정 남용에 대응하기 위한 인시던트 대응 절차를 준비합니다. AWS 계정에서 인시던트 대응 전략 구축에 대한 자세한 내용은 AWS Security Incident Response Guide 및 보안 원칙 백서의 인시던트 대응 섹션에 나온 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 – root credential usage alert
- Step-by-step guidance on monitoring for root credential use through CloudTrail
- AWS에서 사용하도록 승인된 MFA 토큰
- AWS에서 break glass access 구현
- Top 10 security items to improve in your AWS 계정
- AWS 계정에서 승인되지 않은 활동이 감지되면 어떻게 해야 하나요?

관련 비디오:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM

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.

원하는 성과: 비즈니스의 보안 제어 목표가 잘 정의되고 규정 준수 요구 사항에 맞게 조정됩니다. 제어는 자동화와 정책을 통해 구현 및 시행되며 목표 달성의 효율성 측면이 지속적으로 평가됩니다. 특정 시점과 일정 기간의 효과 증명을 감사자에게 쉽게 보고할 수 있습니다.

일반적인 안티 패턴:

- 비즈니스와 관련하여 확실한 보안에 대한 규제 요구 사항, 시장 기대치 및 업계 표준을 제대로 이해하고 있지 않습니다.
- 사이버 보안 프레임워크와 제어 목표가 비즈니스 요구 사항에 맞지 않습니다.
- 제어 구현이 측정 가능한 방식으로 제어 목표와 밀접하게 일치하지 않습니다.
- 자동화를 통해 제어의 효과를 보고하지 않습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

보안 제어 목표의 기초를 형성할 수 있는 일반적인 사이버 보안 프레임워크가 많이 있습니다. 비즈니스에 대한 규제 요구 사항, 시장 기대치, 업계 표준을 고려하여 요구 사항을 가장 잘 지원하는 프레임워크를 결정하세요. 예를 들어, AICPA SOC 2, HITRUST, PCI-DSS, ISO 27001, NIST SP 800-53이 포함됩니다.

제어 목표를 파악한 경우, 사용하는 AWS 서비스가 해당 목표를 달성하는 데 어떤 도움이 되는지 이해해야 합니다. AWS Artifact를 사용하여 AWS에서 포괄하는 책임 범위를 설명하는 대상 프레임워크에 맞게 조정된 문서와 보고서 그리고 사용자 책임에 속하는 나머지 범위에 대한 지침을 찾아보세요. 다양한 프레임워크 제어 설명에 부합하는 추가적인 서비스별 지침은 AWS Customer Compliance Guides를 참조하세요.

목표 달성을 위한 제어를 정의할 때 예방적 제어를 바탕으로 실행을 체계화하고 감지 제어를 사용하여 완화를 자동화하세요. 서비스 제어 정책(SCP)을 사용하여 AWS Organizations에서 리소스 구성 및 작업이 규정을 준수하지 못하는 일이 없도록 방지할 수 있습니다. 규정 미준수 리소스를 모니터링하고 보고한 다음, 해당 동작이 확실해지면 규칙을 실행 모델로 전환하도록 AWS Config에서 규칙을 구현합니다. 사이버 보안 프레임워크에 맞게 미리 정의된 관리형 규칙 세트를 배포하려면 첫 번째 옵션으로 AWS Security Hub 표준 사용을 평가합니다. AWS Foundational Security Best Practices(FSBP) 표준과 CIS AWS Foundations Benchmark는 여러 표준 프레임워크에서 공유되는 많은 목표에 부합하는 제어 기능을 갖추고 있어 좋은 출발점이 될 수 있습니다. Security Hub에 기본적으로 원하는 제어 감지 기능이 없는 경우 AWS Config 규정 준수 팩을 사용하여 보완할 수 있습니다.

AWS 글로벌 보안 및 규정 준수 가속화(GSCA) 팀이 권장하는 APN 파트너 번들을 사용하여 보안 고문, 컨설팅 기관, 증거 수집 및 보고 시스템, 감사자 및 필요한 경우 기타 보완 서비스의 도움을 받을 수 있습니다.

구현 단계

1. 일반적인 사이버 보안 프레임워크를 평가하고 선택한 목표에 맞게 제어 목표를 조정합니다.
2. AWS Artifact를 사용하는 프레임워크에 대한 지침 및 책임 관련 문서를 얻습니다. 공동 책임 모델에서 AWS가 부담하는 규정 준수 부분과 사용자의 책임인 부분을 이해합니다.
3. SCP, 리소스 정책, 역할 신뢰 정책 및 기타 가드레일을 사용하여 리소스 구성 및 작업이 규정을 준수하지 못하는 일이 없도록 합니다.
4. 제어 목표에 맞는 Security Hub 표준 및 AWS Config 규정 준수 팩 배포를 평가합니다.

리소스

관련 모범 사례:

- SEC03-BP01 액세스 요구 사항 정의
- SEC04-BP01 서비스 및 애플리케이션 로깅 구성
- SEC07-BP01 데이터 분류 체계 이해
- OPS01-BP03 거버넌스 요구 사항 평가
- OPS01-BP04 규정 준수 요구 사항 평가
- PERF01-BP05 정책 및 참조 아키텍처 사용
- COST02-BP01 조직 요구 사항에 따라 정책 개발

관련 문서:

- AWS Customer Compliance Guides

관련 도구:

- AWS Artifact

SEC01-BP04 보안 위협 및 권장 사항에 대한 최신 정보 파악

업계 위협 인텔리전스 간행물과 업데이트를 위한 데이터 피드를 모니터링하여 최신 위협 및 방어 조치를 파악하세요. 최신 위협 데이터를 기반으로 자동 업데이트되는 관리형 서비스 제품을 평가합니다.

원하는 성과: 업계 간행물에 최신 위협 및 권장 사항이 업데이트되므로, 지속적으로 정보를 얻을 수 있습니다. 자동화를 사용하여 새로운 위협을 식별할 때 잠재적 취약성과 노출을 탐지합니다. 이러한 위

협에 대해 완화 조치를 취합니다. 최신 위협 인텔리전스로 자동 업데이트되는 AWS 서비스를 채택합니다.

일반적인 안티 패턴:

- 최신 위협 인텔리전스에 대한 정보를 지속적으로 파악할 수 있는 안정적이고 반복 가능한 메커니즘이 없습니다.
- 잠재적 취약성 및 노출에 대해 인적 검토가 필요한 기술 포트폴리오, 워크로드, 종속성에 대한 인벤토리를 수동으로 유지 관리합니다.
- 워크로드 및 종속성을 알려진 위협 완화 기능을 제공하는 최신 지원 버전으로 업데이트할 수 있는 메커니즘이 없습니다.

이 모범 사례 확립의 이점: 위협 인텔리전스 소스를 사용하여 최신 상태를 유지하면 비즈니스에 영향을 미칠 수 있는 위협 환경의 중요한 변화를 놓칠 위험이 줄어듭니다. 워크로드에 잠재적인 취약성이나 노출이 존재하는 부분과 그 종속성을 스캔, 탐지 및 해결하기 위한 자동화 기능을 마련하면 수동 대안에 비해 위험을 빠르고 예측 가능하게 완화하는 데 도움이 될 수 있습니다. 이를 통해 취약성 완화와 관련된 시간과 비용을 제어할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

신뢰할 수 있는 위협 인텔리전스 발행물을 검토하여 위협 상황을 파악합니다. 알려진 적대적 전술, 기법 및 절차(TTP)에 대한 문서는 MITRE ATT&CK 기술 자료를 참조하세요. MITRE의 Common Vulnerabilities and Exposures 목록을 검토하여 사용 중인 제품의 알려진 취약성에 대한 최신 정보를 확인합니다. 널리 알려진 OWASP(Open Worldwide Application Security Project)의 인기 OWASP Top 10 프로젝트를 통해 웹 애플리케이션에 대한 중대한 위험을 이해합니다.

CVE용 AWS 보안 공지를 참조하여 AWS 보안 이벤트 및 권장되는 수정 단계에 대한 최신 정보를 확인합니다.

최신 상태를 유지하는 데 드는 전반적인 수고를 덜고 오버헤드를 줄이려면 시간이 지남에 따라 새로운 위협 인텔리전스를 자동으로 통합하는 AWS 서비스를 사용하는 것이 좋습니다. 예를 들어, Amazon GuardDuty에서는 계정 내에서 비정상적인 동작과 위협 서명을 탐지하기 위한 업계 위협 인텔리전스를 최신 상태로 유지할 수 있습니다. Amazon Inspector에서는 연속 스캔 기능에 사용하는 CVE의 데이터베이스를 자동으로 최신 상태로 유지합니다. AWS WAF 및 AWS Shield Advanced 모두 새로운 위협이 발생하면 자동으로 업데이트되는 관리형 규칙 그룹을 제공합니다.

자동화된 플릿 관리 및 패치 적용의 경우 Well-Architected 운영 우수성 원칙을 검토하세요.

구현 단계

- 비즈니스 및 업계와 관련된 위협 인텔리전스 간행물의 최신 소식을 구독합니다. AWS 보안 공지를 구독합니다.
- Amazon GuardDuty 및 Amazon Inspector와 같이 새로운 위협 인텔리전스를 자동으로 통합하는 서비스 채택을 고려해 보세요.
- Well-Architected 운영 우수성 원칙의 모범 사례에 부합하는 플릿 관리 및 패치 적용 전략을 배포합니다.

리소스

관련 모범 사례:

- SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정
- OPS01-BP05 위협 환경 평가
- OPS11-BP01 지속적인 개선을 위한 프로세스 마련

SEC01-BP05 보안 관리 범위 축소

특정 제어 기능의 관리를 AWS(관리형 서비스)로 전환하는 AWS 서비스를 사용하여 보안 범위를 줄일 수 있는지를 결정합니다. 이러한 서비스는 인프라 프로비저닝, 소프트웨어 설정, 패치 적용, 백업과 같은 보안 유지 관리 작업을 줄이는 데 도움이 될 수 있습니다.

원하는 성과: 워크로드에 맞는 AWS 서비스를 선택할 때 보안 관리 범위를 고려합니다. 관리 오버헤드 및 유지 관리 작업에 드는 비용(총 소유 비용(TCO))은 다른 Well-Architected 고려 사항 외에도 선택한 서비스 비용을 기준으로 산정됩니다. AWS 제어 및 규정 준수 설명서를 제어 평가 및 검증 절차에 통합합니다.

일반적인 안티 패턴:

- 선택한 서비스에 대한 공동 책임 모델을 완전히 이해하지 않고 워크로드를 배포합니다.
- 상응하는 관리형 서비스를 평가하지 않고 가상 머신에서 데이터베이스 및 기타 기술을 호스팅합니다.
- 관리형 서비스 옵션과 비교할 때 보안 관리 작업을 가상 머신의 호스팅 기술에 대한 총 소유 비용에 포함하지 않습니다.

이 모범 사례 확립의 이점: 관리형 서비스를 사용하면 운영 보안 제어를 관리하는 데 따르는 전반적인 부담을 낮추고 보안 위험과 총 소유 비용을 줄일 수 있습니다. 특정 보안 작업에 걸리는 시간을 비즈니스에 더 많은 가치를 제공하는 작업에 재투자할 수 있습니다. 또한, 관리형 서비스는 일부 제어 요구 사항을 AWS로 전환하여 규정 준수 요구 사항의 범위를 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS 기반 워크로드의 구성 요소를 통합하는 방법에는 여러 가지가 있습니다. Amazon EC2 인스턴스에 기술을 설치하고 실행하려면 사용자가 전체 보안 책임의 가장 큰 부분을 맡아야 하는 경우가 많습니다. 특정 제어 기능을 운영하는 데 따르는 부담을 줄이려면 사용자가 지는 공동 책임 모델의 범위를 줄이는 AWS 관리형 서비스를 식별하고, 기존 아키텍처에서 이를 사용하는 방법을 이해해야 합니다. 예를 들어 데이터베이스 배포에 Amazon Relational Database Service(RDS) 사용, 컨테이너 오케스트레이션에 Amazon Elastic Kubernetes Service(Amazon EKS) 또는 Amazon Elastic Container Service(Amazon ECS) 사용 또는 서버리스 옵션 사용이 포함됩니다. 새 애플리케이션을 구축할 때는 보안 제어를 구현하고 관리하는 데 필요한 시간과 비용을 줄이는 데 어떤 서비스가 도움이 될 수 있는지 생각해 보세요.

규정 준수 요구 사항도 서비스를 선택할 때 고려할 요인이 될 수 있습니다. 관리형 서비스는 일부 요구 사항의 규정 준수를 AWS로 전환할 수 있습니다. 운영 및 관리하는 서비스의 양상을 감사하고 관련 AWS 감사 보고서의 제어 설명을 수락하는 데 있어 규정 준수 팀이 어느 정도 편안함을 느끼는지 규정 준수 팀과 논의하세요. AWS Artifact에서 발견된 감사 아티팩트를 감사 기관이나 규제 기관에 AWS 보안 통제의 증거로 제공할 수 있습니다. 또한 일부 AWS 감사 아티팩트에서 제공하는 책임 지침을 AWS Customer Compliance Guides와 함께 사용하여 아키텍처를 설계할 수 있습니다. 이 지침은 시스템의 특정 사용 사례를 지원하기 위해 적용해야 하는 추가 보안 제어를 결정하는 데 도움이 됩니다.

관리형 서비스를 사용할 경우 리소스를 최신 버전으로 업데이트하는 프로세스를 숙지해야 합니다(예: Amazon RDS에서 관리하는 데이터베이스 버전 또는 AWS Lambda 함수의 프로그래밍 언어 런타임 업데이트). 관리형 서비스가 이 작업을 대신 수행할 수도 있지만, 업데이트 시기를 구성하고 운영에 미치는 영향을 이해하는 것은 사용자의 책임입니다. AWS Health와 같은 도구를 사용하면 환경 전체에서 이러한 업데이트를 추적하고 관리할 수 있습니다.

구현 단계

1. 관리형 서비스로 대체할 수 있는 워크로드 구성 요소를 평가하세요.
    a. 워크로드를 AWS로 마이그레이션하는 경우 워크로드를 리호스팅, 리팩터링, 리플랫포밍, 재구축 또는 교체해야 하는지 평가할 때 관리에 드는 시간 및 비용 절감과 위험 감소를 고려해야 합니다.

때로는 마이그레이션을 시작할 때 추가적으로 투자하여 장기적으로 상당한 비용을 절감할 수 있습니다.

2. 자체 기술 배포를 설치하고 관리하는 대신 관리형 서비스(예: Amazon RDS)를 구현하는 것을 고려해 보세요.
3. AWS Artifact의 책임 지침을 참조하여 워크로드에 적용해야 하는 보안 제어를 결정하세요.
4. 사용 중인 리소스의 인벤토리를 유지하고 최신 서비스와 접근 방식을 최신 상태로 유지하여 범위를 줄일 수 있는 새로운 기회를 찾아보세요.

리소스

관련 모범 사례:

- PERF02-BP01 워크로드에 가장 적합한 컴퓨팅 옵션 선택
- PERF03-BP01 데이터 액세스 및 스토리지 요구 사항을 가장 잘 지원하는 목적별 데이터 스토어 사용
- SUS05-BP03 관리형 서비스 사용

관련 문서:

- Planned lifecycle events for AWS Health

관련 도구:

- AWS Health
- AWS Artifact
- AWS Customer Compliance Guides

관련 비디오:

- How do I migrate to an Amazon RDS or Aurora MySQL DB instance using AWS DMS?
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health

SEC01-BP06 표준 보안 제어의 배포 자동화

AWS 환경 전반에서 표준인 보안 제어 기능을 개발하고 배포할 때 최신 DevOps 사례를 적용하세요. 코드형 인프라(IaC) 템플릿을 사용하여 표준 보안 제어 및 구성을 정의하고, 버전 제어 시스템에서 변경 사항을 캡처하고, CI/CD 파이프라인의 일부로 변경 사항을 테스트하며, AWS 환경에 변경 사항을 자동으로 배포할 수 있습니다.

원하는 성과: IaC 템플릿이 표준화된 보안 제어를 캡처하고 이를 버전 제어 시스템에 적용합니다. CI/CD 파이프라인은 변경 사항을 탐지하고 AWS 환경 테스트 및 배포를 자동화하는 곳에 마련되어 있습니다. 배포를 진행하기 전에 템플릿의 구성 오류를 탐지하고 알림을 보내기 위한 가드레일이 있습니다. 워크로드는 표준 제어 기능이 적용되는 환경에 배포됩니다. 팀은 셀프 서비스 메커니즘을 통해 승인된 서비스 구성을 배포할 수 있습니다. 제어 구성, 스크립트 및 관련 데이터를 위한 안전한 백업 및 복구 전략이 마련되어 있습니다.

일반적인 안티 패턴:

- 웹 콘솔 또는 명령줄 인터페이스를 통해 표준 보안 제어 기능을 수동으로 변경합니다.
- 개별 워크로드 팀에 의존하여 중앙 팀이 정의한 제어 기능을 수동으로 구현합니다.
- 워크로드 팀의 요청에 따라 중앙 보안 팀에 의존하여 워크로드 수준 제어 기능을 배포합니다.
- 동일한 개인 또는 팀이 적절히 업무를 분담하거나 점검하며 균형을 맞추지 않고 보안 제어 자동화 스크립트를 개발, 테스트 및 배포할 수 있습니다.

이 모범 사례 확립의 이점: 템플릿을 사용하여 표준 보안 제어를 정의하면 버전 제어 시스템을 통해 시간 경과에 따른 변경 사항을 추적하고 비교할 수 있습니다. 자동화를 사용하여 변경 사항을 테스트하고 배포하면 표준화와 예측 가능성을 높여 배포가 성공할 확률이 커지고 수동 반복 작업을 줄일 수 있습니다. 워크로드 팀이 승인된 서비스 및 구성을 배포할 수 있는 셀프 서비스 메커니즘을 제공하면 구성 오류 및 오용의 위험을 줄일 수 있습니다. 또한, 개발 프로세스 초기에 제어 기능을 통합하는 데도 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

SEC01-BP01 계정을 사용하여 워크로드 분리에서 설명한 사례에 따라 AWS Organizations를 사용하여 관리하는 여러 환경에 여러 AWS 계정이 필요합니다. 이러한 각 환경과 워크로드에는 별도의 보안 제어가 필요할 수 있지만, 조직 전체에서 일부 보안 제어를 표준화할 수 있습니다. 예로는 중앙 집중식 ID 제공업체 통합, 네트워크 및 방화벽 정의, 로그 저장 및 분석을 위한 표준 위치 구성 등이 있습니다. 코드형 인프라(IaC)를 사용하여 인프라 프로비저닝에 동일하게 엄격한 애플리케이션 코드 개발을 적용할 수 있는 것과 마찬가지로, IaC를 통해 표준 보안 제어를 정의하고 배포할 수도 있습니다.

가능하면 AWS CloudFormation과 같은 선언적인 방법으로 보안 제어를 정의하고 소스 제어 시스템에 저장합니다. DevOps 관행을 바탕으로 보다 예측 가능한 릴리스를 위해 제어 기능을 자동으로 배포하고, AWS CloudFormation Guard와 같은 도구를 사용하여 자동화된 테스트를 수행하며, 배포된 제어 기능과 원하는 구성 간의 차이를 탐지합니다. AWS CodePipeline, AWS CodeBuild, AWS CodeDeploy와 같은 서비스를 사용하여 CI/CD 파이프라인을 구성할 수 있습니다. Organizing Your AWS Environment Using Multiple Accounts의 지침 참조하여 다른 배포 파이프라인과 분리된 자체 계정에서 이러한 서비스를 구성하세요.

템플릿을 정의하여 AWS 계정, 서비스 및 구성을 정의하고 배포하는 것을 표준화할 수도 있습니다. 이 기술을 사용하면 중앙 보안 팀에서 정의를 관리하고 셀프 서비스 접근 방식을 통해 워크로드 팀에 제공할 수 있습니다. 이를 달성하는 한 가지 방법은 워크로드 팀이 자체 파이프라인 배포에 통합 가능한 제품으로 템플릿을 게시할 수 있는 Service Catalog를 사용하는 것입니다. AWS Control Tower를 사용하는 경우 일부 템플릿과 제어 기능을 시작점으로 삼을 수 있습니다. 또한 Control Tower는 Account Factory 기능을 제공하여 워크로드 팀이 정의한 표준을 통해 새로운 AWS 계정 계정을 만들 수 있도록 합니다. 이 기능을 사용하면 중앙 팀에 종속되지 않고 워크로드 팀에서 필요하다고 판단한 경우 새 계정을 승인하고 생성할 수 있습니다. 이러한 계정은 제공하는 기능, 처리 중인 데이터의 민감도 또는 동작과 같은 이유에 따라 다양한 워크로드 구성 요소를 분리하는 데 필요할 수 있습니다.

구현 단계

1. 버전 관리 시스템에서 템플릿을 저장하고 유지 관리하는 방법을 결정합니다.
2. CI/CD 파이프라인을 생성하여 템플릿을 테스트하고 배포합니다. 잘못된 구성이 있는지 점검하고 템플릿이 기업 표준을 준수하는지 확인하는 테스트를 정의합니다.
3. 워크로드 팀이 요구 사항에 따라 AWS 계정을 배포하고 서비스할 수 있도록 표준화된 템플릿 카탈로그를 구축합니다.
4. 제어 구성, 스크립트 및 관련 데이터에 대한 안전한 백업 및 복구 전략을 구현합니다.

리소스

관련 모범 사례:

- OPS05-BP01 버전 관리 사용
- OPS05-BP04 구축 및 배포 관리 시스템 사용
- REL08-BP05 자동화를 통한 변경 사항 배포
- SUS06-BP01 지속 가능성 개선을 신속하게 도입할 수 있는 방법 채택

관련 문서:

- Organizing Your AWS Environment Using Multiple Accounts

관련 예제:

- Automate account creation, and resource provisioning using Service Catalog, AWS Organizations, and AWS Lambda
- Strengthen the DevOps pipeline and protect data with AWS Secrets Manager, AWS KMS, and AWS Certificate Manager

관련 도구:

- AWS CloudFormation Guard
- Landing Zone Accelerator on AWS

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

위협 모델링이란 무엇인가요?

"위협 모델링은 가치 있는 것을 보호한다는 맥락에서 위협과 완화 조치를 식별, 전달 및 이해하기 위해 작동합니다." – The Open Web Application Security Project (OWASP) Application Threat Modeling

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 시프트-레프트 보안이라는 업계 원칙과 일치합니다. 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 어떻게 수행할 수 있나요?

위협 모델링을 수행하는 방법에는 여러 가지가 있습니다. 프로그래밍 언어와 마찬가지로 각각 장단점이 있으므로 자신에게 가장 적합한 방법을 선택해야 합니다. 한 가지 접근 방식은 Shostack's 4 Question Frame for Threat Modeling으로 시작하는 것입니다. 여기에서는 위협 모델링 실습에 대한 구조를 제공하기 위해 개방형 질문을 제시합니다.

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 가령 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. 잘못되면 어떻게 되나요?

   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 내 모든 개인과 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.

   보안과 규정 준수는 AWS와 고객의 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.

   공동 책임에서 AWS가 맡은 부분의 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. AWS 고객은 AWS Artifact에서 이러한 프로그램의 감사 보고서를 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 장이 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 기능에 해당될 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 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 게시물을 참조하세요.

Threat Composer

위협 모델링을 수행하는 데 도움과 안내를 받으려면 위협 모델링 시 가치 창출 시간을 단축하는 것을 목표로 하는 Threat Composer 도구를 사용하는 것이 좋습니다. 이 도구는 다음과 같은 작업을 수행하는 데 도움이 됩니다.

- 자연스러운 비선형 워크플로우에서 작동하는 위협 문법에 맞게 유용한 위협 설명을 작성합니다.
- 사람이 읽을 수 있는 위협 모델을 생성합니다.
- 기계가 읽을 수 있는 위협 모델을 생성하여 위협 모델을 코드로 취급할 수 있습니다.
- Insights Dashboard를 사용하여 품질 및 커버리지 개선 영역을 빠르게 식별할 수 있도록 도와줍니다.

자세한 내용을 보려면 Threat Composer를 방문하여 시스템 정의 Example Workspace로 전환합니다.

리소스

관련 모범 사례:

- SEC01-BP03 제어 목표 파악 및 검증
- SEC01-BP04 보안 위협 및 권장 사항에 대한 최신 정보 파악
- SEC01-BP05 보안 관리 범위 축소
- SEC01-BP08 새로운 보안 서비스 및 기능을 정기적으로 평가 및 구현

관련 문서:

- How to approach threat modeling (AWS Security Blog)
- NIST: Guide to Data-Centric System Threat Modelling

관련 비디오:

- AWS Summit ANZ 2021 - How to approach threat modelling
- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery

관련 교육:

- Threat modeling the right way for builders – AWS Skill Builder virtual self-paced training
- Threat modeling the right way for builders – AWS 워크숍

관련 도구:

- Threat Composer

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.

원하는 성과: AWS 및 AWS 파트너가 출시한 새로운 기능 및 서비스를 알려주는 표준 관행이 마련되어 있습니다. 이러한 최신 기능이 환경과 워크로드에 대한 현재 제어 설계와 새로운 제어 설계에 어떤 영향을 미치는지 평가합니다.

일반적인 안티 패턴:

- AWS 블로그와 RSS 피드를 구독하여 관련 새 기능 및 서비스를 신속하게 알아보지 않습니다.
- 2차 소스에서 제공하는 보안 서비스 및 기능에 대한 소식 및 업데이트를 적극 활용합니다.
- 조직의 AWS 사용자에게 최신 업데이트에 대한 새로운 정보를 계속 파악하도록 권장하지 않습니다.

이 모범 사례 확립의 이점: 새로운 보안 서비스 및 기능을 잘 파악하면 클라우드 환경 및 워크로드의 제어 구현에 대해 정보에 입각한 결정을 내릴 수 있습니다. 이러한 소스는 진화하는 보안 환경과 새롭게 등장하는 위협으로부터 AWS 서비스를 보호하는 데 사용할 수 있는 방법에 대한 인식을 높이는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

AWS는 다음과 같은 여러 채널을 통해 고객에게 새로운 보안 서비스 및 기능을 안내합니다.

- AWS의 새로운 소식
- AWS 뉴스 블로그
- AWS 보안 블로그
- AWS 보안 공지
- AWS 문서 개요

Amazon Simple Notification Service(SNS)를 통해 AWS 일간 기능 업데이트 주제를 구독하여 업데이트에 대한 포괄적인 일일 요약 정보를 확인할 수 있습니다. Amazon GuardDuty, AWS Security Hub와 같은 일부 보안 서비스는 해당 서비스에 대한 새로운 표준, 조사 결과 및 기타 업데이트 관련 정보를 파악할 수 있도록 자체 SNS 주제를 제공합니다.

나아가 매년 전 세계에서 개최되는 컨퍼런스, 이벤트, 웨비나를 통해 새로운 서비스와 기능에 대해 자세히 발표하고 설명합니다. 특히 주목할 것은 연례 AWS re:Inforce 보안 컨퍼런스와 보다 일반적인 AWS re:Invent 컨퍼런스입니다. 앞서 언급한 AWS 뉴스 채널은 이러한 컨퍼런스에서 보안 및 기타 서비스에 대해 발표한 내용을 공유합니다. YouTube의 AWS Events 채널에서 온라인으로 심층 분석 교육 브레이크아웃 세션을 볼 수 있습니다.

또한 AWS 계정 팀에 최신 보안 서비스 업데이트 및 권장 사항에 대해 문의할 수 있습니다. 직접 연결되는 연락처 정보가 없는 경우 영업 지원 양식을 통해 팀에 문의할 수 있습니다. 마찬가지로, AWS Enterprise Support를 구독한 경우 TAM(Technical Account Manager)으로부터 매주 업데이트를 받고 정기적인 검토 회의 일정을 잡을 수 있습니다.

구현 단계

1. 좋아하는 RSS 리더와 함께 다양한 블로그와 게시판을 구독하거나 일간 기능 업데이트 SNS 주제를 구독합니다.
2. 어떤 AWS 이벤트에 참석해야 하는지 평가하여 새로운 기능과 서비스에 대해 직접 알아보세요.
3. 보안 서비스 및 기능 업데이트에 관한 질문이 있으면 AWS 계정 팀과 회의를 진행하세요.
4. Enterprise Support를 구독하여 Technical Account Manager(TAM)와 정기적으로 상담하는 것을 고려해 보세요.

리소스

관련 모범 사례:

- PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해
- COST01-BP07 새로운 서비스 릴리스로 최신 상태 유지

자격 증명 및 액세스 관리

Questions

- SEC 2. 사람과 시스템에 대한 인증은 어떻게 관리하나요?
- SEC 3. 사람과 시스템에 대한 권한은 어떻게 관리하나요?

SEC 2. 사람과 시스템에 대한 인증은 어떻게 관리하나요?

보안 AWS 워크로드를 운영할 때는 두 가지 유형의 ID를 관리해야 합니다.

- 인적 자격 증명: AWS 환경 및 애플리케이션에 액세스해야 하는 인적 자격 증명은 인력, 서드파티 및 사용자라는 세 그룹으로 분류할 수 있습니다.

  인력 그룹에는 조직의 구성원인 관리자, 개발자 및 운영자가 포함됩니다. 이들이 AWS 리소스를 관리, 구축 및 운영하려면 액세스 권한이 필요합니다.

  서드파티는 계약업체, 공급업체 또는 파트너와 같은 외부 협력자입니다. 이들은 계약의 일환으로 AWS 리소스와 상호 작용합니다.

  사용자는 애플리케이션의 소비자입니다. 이들은 웹 브라우저, 클라이언트 애플리케이션, 모바일 앱 또는 대화형 명령줄 도구를 통해 AWS 리소스에 액세스합니다.
- 머신 자격 증명: 워크로드 애플리케이션, 운영 도구 및 구성 요소에서 AWS 서비스에 요청을 하려면 (예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 자격 증명에는 AWS 환경에서 실행되는 머신이 포함됩니다(예: Amazon EC2 인스턴스 또는 AWS Lambda 함수). 외부 당사자 또는 AWS 환경에 액세스해야 하는 AWS 외부 머신의 머신 자격 증명을 관리할 수도 있습니다.

모범 사례

- SEC02-BP01 강력한 로그인 메커니즘 사용
- SEC02-BP02 임시 자격 증명 사용
- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 ID 공급업체 사용
- SEC02-BP05 정기적으로 자격 증명 감사 및 교체
- SEC02-BP06 사용자 그룹 및 속성 사용

SEC02-BP01 강력한 로그인 메커니즘 사용

로그인(로그인 자격 증명을 사용한 인증)은 다중 인증(MFA)과 같은 메커니즘을 사용하지 않을 때, 특히 로그인 자격 증명이 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 성과: AWS Identity and Access Management(IAM) 사용자, AWS 계정 루트 사용자, AWS IAM Identity Center, 서드파티 ID 제공업체에 대한 강력한 로그인 메커니즘을 사용하여 AWS의 자격 증명

에 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 자격 증명을 공유합니다.
- 의심스러운 로그인에 대한 탐지 제어를 사용하지 않습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

인적 자격 증명으로 AWS에 로그인하는 방법에는 몇 가지가 있습니다. AWS에 인증할 때 페더레이션(AWS IAM과 중앙 집중화된 IdP 간의 직접 SAML 2.0 페더레이션 또는 AWS IAM Identity Center 사용)을 사용하는 중앙 집중식 ID 제공업체를 사용하는 것이 AWS 모범 사례입니다. 이 경우 ID 제공업체 또는 Microsoft Active Directory를 사용하여 보안 로그인 프로세스를 설정합니다.

AWS 계정을 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 계정을 개설한 직후에는 계정 루트 사용자에 대해 다중 인증(MFA)을 활성화하고 AWS 모범 사례 가이드를 사용하여 루트 사용자를 보호하는 것이 중요합니다.

AWS IAM Identity Center는 인력 사용자를 위해 설계되었으며 서비스 내에서 사용자 ID를 생성 및 관리하고 MFA로 로그인 프로세스를 보호할 수 있습니다. 반면 AWS Cognito는 애플리케이션의 외부 사용자 ID에 대한 사용자 풀 및 ID 제공업체를 제공하는 고객 ID 및 액세스 관리(CIAM)용으로 설계되었습니다.

AWS IAM Identity Center에서 사용자를 생성하는 경우 해당 서비스에서 로그인 프로세스를 보호하고 MFA를 켭니다. 애플리케이션의 외부 사용자 ID의 경우 Amazon Cognito 사용자 풀을 사용하고 해당 서비스에서 로그인 프로세스를 보호하거나 Amazon Cognito 사용자 풀이 지원하는 ID 제공업체 중 하나를 사용할 수 있습니다.

또한 AWS IAM Identity Center의 사용자의 경우 AWS 리소스에 대한 액세스 권한을 부여하기 전에 AWS Verified Access를 사용하여 사용자의 자격 증명 및 디바이스 태세를 확인하여 추가 보안 계층을 제공할 수 있습니다.

AWS Identity and Access Management(IAM) 사용자를 사용하는 경우 IAM을 사용하여 로그인 프로세스를 보호합니다.

AWS IAM Identity Center와 직접 IAM 페더레이션을 동시에 사용하여 AWS에 대한 액세스를 관리할 수 있습니다. IAM 페더레이션을 사용하여 AWS Management Console 및 서비스에 대한 액세스를 관리하고 IAM Identity Center를 사용하여 QuickSight 또는 Amazon Q Business와 같은 비즈니스 애플리케이션에 대한 액세스를 관리할 수 있습니다.

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.

- MFA 필수(Require MFA). 사람의 ID 및 워크로드에 MFA를 요구하는 것이 IAM 모범 사례입니다. MFA를 활성화하면 사용자가 로그인 자격 증명과 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.
- 사용자에게 자신의 암호를 변경할 수 있도록 허용
- 공유 자격 증명 대신 개별 자격 증명을 생성합니다. 개별 자격 증명을 생성하여 각 사용자에게 고유한 보안 자격 증명 세트를 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 미리 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 사용자 풀 디렉터리 권장 사항:

- 암호 강도 설정을 구성합니다.
- 사용자에게 MFA를 요청합니다.
- Amazon Cognito 사용자 풀의 고급 보안 설정을 사용하여 의심되는 로그인을 차단할 수 있는 적응형 인증과 같은 기능을 사용합니다.

IAM 사용자 권장 사항:

- IAM Identity Center 또는 직접 페더레이션을 사용하는 것이 좋습니다. 그러나 IAM 사용자가 필요할 수 있습니다. 이 경우 IAM 사용자에 대한 암호 정책을 설정합니다. 암호 정책을 사용하여 최소 길이 또는 알파벳 이외 문자 포함 여부 등과 같은 요구 사항을 정의할 수 있습니다.
- MFA 로그인을 적용하도록 IAM 정책을 생성합니다. 그러면 사용자가 자신의 암호와 MFA 디바이스를 관리할 수 있습니다.

리소스

관련 모범 사례:

- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 ID 공급업체 사용
- SEC03-BP08 안전하게 조직과 리소스 공유

관련 문서:

- AWS IAM Identity Center 암호 정책
- IAM 사용자의 암호 정책
- AWS 계정 루트 사용자 암호 설정
- Amazon Cognito password policy
- AWS 보안 인증 정보
- IAM 보안 모범 사례

관련 비디오:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP02 임시 자격 증명 사용

모든 유형의 인증을 수행할 때 자격 증명이 의도치 않게 공개되거나 공유되거나 도난당하는 위험을 줄이거나 제거하기 위해 장기 자격 증명 대신 임시 자격 증명을 사용하는 것이 가장 좋습니다.

원하는 성과: 장기 자격 증명의 위험을 줄이려면 가능한 한 인적 자격 증명과 시스템 자격 증명 모두에 대해 임시 자격 증명을 사용합니다. 장기 자격 증명은 퍼블릭 리포지토리에 대한 업로드를 통한 노출과 같은 많은 위험을 초래합니다. 임시 자격 증명을 사용하면 자격 증명이 손상될 가능성이 크게 줄어듭니다.

일반적인 안티 패턴:

- 개발자가 페더레이션을 사용하여 CLI에서 임시 자격 증명을 획득하는 대신 IAM 사용자의 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 자격 증명을 사용할 수 있는 경우에도 시스템 자격 증명에 장기 액세스 키를 사용합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

모든 AWS API 및 CLI 요청에 대해 장기 자격 증명 대신 임시 자격 증명을 사용합니다. AWS 서비스에 대한 API 및 CLI 요청은 거의 모든 경우에 AWS 액세스 키를 사용하여 서명해야 합니다. 이러한 요청은 임시 또는 장기 자격 증명으로 서명할 수 있습니다. 장기 액세스 키라고도 하는 장기 자격 증명을 사용해야 하는 유일한 경우는 IAM 사용자 또는 AWS 계정 루트 사용자를 사용하는 경우입니다. 다른 방법을 통해 AWS에 페더레이션하거나 IAM 역할을 수임할 때 임시 자격 증명이 생성됩니다. 로그인 자격 증명을 사용하여 AWS Management Console에 액세스하는 경우에도 AWS 서비스를 직접 호출할 수 있도록 임시 자격 증명이 생성됩니다. 장기 자격 증명이 필요한 경우는 거의 없으며 임시 자격 증명을 사용하여 대부분의 작업을 수행할 수 있습니다.

임시 자격 증명을 선호하여 장기 자격 증명 사용을 피하는 것은 페더레이션 및 IAM 역할과 함께 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

인적 자격 증명

직원, 관리자, 개발자, 운영자와 같은 인적 자격 증명의 경우:

- 중앙 집중식 ID 제공업체를 사용해야 하며 인적 사용자가 임시 자격 증명을 사용하여 AWS에 액세스하려면 ID 제공업체와의 페더레이션을 사용해야 합니다. 사용자에 대한 페더레이션은 각 AWS 계정에 대한 직접 페더레이션을 사용하거나 AWS IAM Identity Center 및 선택한 ID 제공업체를 사용하여 수행할 수 있습니다. 페더레이션은 장기 자격 증명을 제거하는 것 외에도 IAM 사용자를 사용하는 것보다 많은 이점을 제공합니다. 사용자는 직접 페더레이션을 위해 명령줄에서 또는 IAM Identity Center를 사용하여 임시 자격 증명을 요청할 수도 있습니다. 즉, IAM 사용자 또는 사용자에 대한 장기 자격 증명이 필요한 사용 사례가 거의 없습니다.

서드파티 자격 증명의 경우:

- 서비스형 소프트웨어(SaaS) 공급자와 같은 서드파티에 AWS 계정의 리소스에 대한 액세스 권한을 부여할 때 크로스 계정 역할 및 리소스 기반 정책을 사용할 수 있습니다. 또한 B2B SaaS 고객 또는 파트너를 위한 Amazon Cognito OAuth 2.0 권한 부여 클라이언트 자격 증명 흐름을 사용할 수 있습니다.

웹 브라우저, 클라이언트 애플리케이션, 모바일 앱 또는 대화형 명령줄 도구를 통해 AWS 리소스에 액세스하는 사용자 자격 증명:

- 소비자 또는 고객에게 AWS 리소스에 대한 액세스 권한을 부여해야 하는 경우 Amazon Cognito 자격 증명 풀 또는 Amazon Cognito 사용자 풀을 사용하여 임시 자격 증명을 제공할 수 있습니다. 자격 증명의 권한은 IAM 역할을 통해 제어됩니다. 인증되지 않은 게스트 사용자의 권한이 제한된 개별 IAM 역할을 정의할 수도 있습니다.

기계 자격 증명

시스템 자격 증명의 경우 장기 자격 증명을 사용해야 할 수 있습니다. 이 경우 AWS에 액세스하려면 워크로드에 IAM 역할이 있는 임시 자격 증명을 사용하도록 요구해야 합니다.

- Amazon Elastic Compute Cloud(Amazon EC2)의 경우 Amazon EC2의 역할을 사용할 수 있습니다.
- https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.htmlAWS를 사용하면 임시 자격 증명을 사용하여 AWS Lambda 작업을 수행할 수 있는 서비스 권한을 부여하도록 Lambda 실행 역할을 구성할 수 있습니다. IAM 역할을 사용하여 임시 자격 증명을 부여하는 AWS 서비스에 대한 다른 유사한 모델이 많이 있습니다.
- IoT 디바이스의 경우 AWS IoT Core 자격 증명 공급자를 사용하여 임시 자격 증명을 요청할 수 있습니다.

- AWS 리소스에 액세스해야 하는 AWS 외부에서 실행되는 시스템 또는 온프레미스 시스템의 경우 IAM Roles Anywhere를 사용할 수 있습니다.

임시 자격 증명이 지원되지 않는 시나리오가 있으며, 이 경우 장기 자격 증명을 사용해야 합니다. 이러한 상황에서는 자격 증명을 주기적으로 감사 및 교체하고 정기적으로 액세스 키를 교체합니다. 매우 제한된 IAM 사용자 액세스 키의 경우 다음과 같은 추가 보안 조치를 고려하세요.

- 매우 제한된 권한 부여:
  - 최소 권한 원칙을 준수합니다(작업, 리소스 및 조건을 구체적으로 지정).
  - IAM 사용자에게 하나의 특정 역할에 대한 AssumeRole 작업만 부여하는 것을 고려합니다. 온프레미스 아키텍처에 따라 이 접근 방식은 장기 IAM 자격 증명을 격리하고 보호하는 데 도움이 됩니다.
- IAM 역할 신뢰 정책에서 허용되는 네트워크 소스 및 IP 주소를 제한합니다.
- 사용량을 모니터링하고 미사용 권한 또는 오용에 대한 알림을 설정합니다(AWS CloudWatch Logs 지표 필터 및 경보 사용).
- 권한 경계를 적용합니다(서비스 제어 정책(SCP) 및 권한 경계가 서로를 보완함 - SCP는 개괄적이지만 권한 경계는 세분화됨).
- 자격 증명을 프로비저닝하고 안전하게 저장하는 프로세스를 구현합니다(온프레미스 볼트에 저장).

장기 보안 인증이 필요한 시나리오에 대한 몇 가지 다른 옵션은 다음과 같습니다.

- 자체 토큰 벤딩 API를 구축합니다(Amazon API Gateway 사용).
- 장기 자격 증명을 사용해야 하는 상황이나 데이터베이스 로그인과 같이 AWS 액세스 키 이외의 자격 증명을 사용해야 하는 시나리오에서 AWS Secrets Manager와 같은 보안 암호 관리를 처리하도록 설계된 서비스를 사용할 수 있습니다. Secrets Manager는 암호화된 보안 암호의 관리, 교체 및 보안 저장을 간소화합니다. 많은 AWS 서비스가 Secrets Manager와의 직접 통합을 지원합니다.
- 멀티 클라우드 통합의 경우 소스 자격 증명 서비스 공급자(CSP) 자격 증명을 기반으로 ID 페더레이션을 사용할 수 있습니다(AWS STSAssumeRoleWithWebIdentity 참조).

장기 자격 증명 교체에 대한 자세한 내용은 rotating access keys를 참조하세요.

리소스

관련 모범 사례:

- SEC02-BP03 안전하게 보안 암호 저장 및 사용
- SEC02-BP04 중앙 집중식 ID 공급업체 사용

- SEC03-BP08 안전하게 조직과 리소스 공유

관련 문서:

- Temporary Security Credentials
- AWS 보안 인증
- IAM 보안 모범 사례
- IAM 역할
- IAM Identity Center
- Identity Providers and Federation
- Rotating Access Keys
- 보안 파트너 솔루션: 액세스 및 액세스 제어
- AWS 계정 루트 사용자
- Access AWS using a Google Cloud Platform native workload identity
- How to access AWS resources from Microsoft Entra ID tenants using AWS Security Token Service

관련 비디오:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP03 안전하게 보안 암호 저장 및 사용

워크로드에는 데이터베이스, 리소스 및 서드파티 서비스에 대한 자격 증명을 증명하는 자동화된 기능이 필요합니다. 이는 API 액세스 키, 암호, OAuth 토큰과 같은 보안 암호 액세스 자격 증명을 사용하여 수행됩니다. 특별 제작된 서비스를 사용하여 이러한 자격 증명을 저장, 관리 및 교체하면 해당 자격 증명이 손상될 가능성을 줄이는 데 도움이 됩니다.

원하는 성과: 다음 목표를 달성하는 애플리케이션 자격 증명을 안전하게 관리하기 위한 메커니즘을 구현합니다.

- 워크로드에 필요한 보안 암호를 식별합니다.
- 가능한 경우 장기 자격 증명을 단기 자격 증명으로 대체하여 필요한 장기 자격 증명의 수를 줄입니다.

- 나머지 장기 자격 증명의 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.
- 자격 증명이 의도치 않게 공개될 가능성을 줄입니다.

일반적인 안티 패턴:

- 자격 증명을 교체하지 않습니다.
- 소스 코드 또는 구성 파일에 장기 자격 증명을 저장합니다.
- 자격 증명을 저장 시 암호화되지 않은 상태로 저장합니다.

이 모범 사례 확립의 이점:

- 보안 암호는 저장 시 및 전송 중 암호화된 상태로 저장됩니다.
- 자격 증명에 대한 액세스 권한은 API를 통해 제한됩니다(자격 증명 자판기와 같음).
- 자격 증명에 대한 액세스 권한(읽기 및 쓰기 모두)은 감사되고 로깅됩니다.
- 우려 사항 분리: 자격 증명 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.
- 자격 증명에 대한 액세스 권한은 세분화된 방식으로 제어할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

과거에는 데이터베이스, 서드파티 API, 토큰 및 기타 보안 암호에 인증하는 데 사용되는 자격 증명이 소스 코드나 환경 파일에 포함되었을 수 있습니다. AWS는 이러한 자격 증명을 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 자격 증명은 저장, 관리 또는 처리할 필요가 없는 자격 증명입니다. 안전하게 제거할 수 있는 워크로드 기능에 더 이상 필요하지 않은 자격 증명이 있을 수 있습니다.

워크로드의 적절한 기능을 위해 여전히 필요한 자격 증명의 경우 장기 자격 증명을 임시 또는 단기 자격 증명으로 대체할 기회가 있을 수 있습니다. 예를 들어 AWS 비밀 액세스 키를 하드 코딩하는 대신 IAM 역할을 사용하여 해당 장기 자격 증명을 임시 자격 증명으로 대체하는 것이 좋습니다.

일부 장기 보안 암호는 제거하거나 대체하지 못할 수 있습니다. 이러한 보안 암호는 AWS Secrets Manager와 같은 서비스에 저장될 수 있습니다. 이를 통해 중앙에서 저장 및 관리되고 정기적으로 교체될 수 있습니다.

워크로드의 소스 코드 및 구성 파일을 감사하면 여러 유형의 자격 증명을 확인할 수 있습니다. 다음 표에는 일반적인 유형의 자격 증명을 처리하기 위한 전략이 요약되어 있습니다.

자격 증명 유형	설명	제안된 전략
IAM 액세스 키	워크로드 내에서 IAM 역할을 수임하는 데 사용되는 AWS IAM 액세스 및 비밀 키	교체: 대신 컴퓨팅 인스턴스에 할당된 IAM 역할을 사용합니다(예: Amazon EC2 또는 AWS Lambda). AWS 계정의 리소스에 대한 액세스가 필요한 서드 파티와의 상호 운용성을 위해 AWS 크로스 계정 액세스를 지원하는지 문의합니다. 모바일 앱의 경우 Amazon Cognito 자격 증명 풀(페더레이션 자격 증명)을 통한 임시 자격 증명 사용을 고려하세요. AWS 외부에서 실행되는 워크로드의 경우 IAM Roles Anywhere 또는 AWS Systems Manager 하이브리드 정품 인증을 고려하세요. 컨테이너는 Amazon ECS 작업 IAM 역할 또는 Amazon EKS node IAM role을 참조하세요.
SSH 키	수동으로 또는 자동화된 프로세스의 일부로 Linux EC2 인스턴스에 로그인하는 데 사용되는 Secure Shell 프라이빗 키	교체: AWS Systems Manager 또는 EC2 Instance Connect를 사용하여 IAM 역할을 통해 EC2 인스턴스에 대한 프로그

자격 증명 유형	설명	제안된 전략
		래밍 방식의 액세스 및 인적 액세스를 제공합니다.
애플리케이션 및 데이터베이스 자격 증명	암호 - 일반 텍스트 문자열	교체: AWS Secrets Manager에 자격 증명을 저장하고 가능하면 자동 교체를 설정합니다.
Amazon RDS 및 Aurora 관리자 데이터베이스 자격 증명	암호 - 일반 텍스트 문자열	교체: Amazon RDS와의 Secrets Manager 통합 또는 Amazon Aurora를 사용합니다. 또한 일부 RDS 데이터베이스 유형은 일부 사용 사례에서 암호 대신 IAM 역할을 사용할 수 있습니다(자세한 내용은 IAM 데이터베이스 인증 참조).
OAuth 토큰	보안 암호 토큰 - 일반 텍스트 문자열	교체: AWS Secrets Manager에 토큰을 저장하고 자동 교체를 구성합니다.
API 토큰 및 키	보안 암호 토큰 - 일반 텍스트 문자열	교체: AWS Secrets Manager에 저장하고 가능하면 자동 교체를 설정합니다.

일반적인 안티 패턴은 소스 코드, 구성 파일 또는 모바일 앱 내부에 IAM 액세스 키를 포함하는 것입니다. AWS 서비스와 통신하는 데 IAM 액세스 키가 필요한 경우 임시 (단기) 보안 자격 증명을 사용합니다. 이러한 단기 자격 증명은 EC2 인스턴스의 경우 IAM 역할, Lambda 함수의 경우 실행 역할, 모바일 사용자 액세스의 경우 Cognito IAM 역할, IoT 기기의 경우 IoT Core 정책을 통해 제공할 수 있습니다. 서드파티와 연결할 때 IAM 사용자를 구성하고 서드파티에 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 자격 증명은 물론, API 토큰, 암호 및 기타 자격 증명의 저장, 사용 및 교체를 안전하게 관리하기 위해 특별히 제작되었습니다.

AWS Secrets Manager는 민감한 자격 증명의 안전한 저장 및 처리를 보장하는 5가지 주요 기능, 즉 저장 시 암호화, 전송 중 암호화, 종합적 감사, 세분화된 액세스 제어, 확장 가능한 자격 증명 교체를 제공합니다. AWS 파트너의 기타 보안 암호 관리 서비스 또는 유사한 기능과 보증을 제공하는 현지 개발 솔루션도 허용됩니다.

보안 암호를 검색할 때 Secret Manager 클라이언트 측 캐싱 구성 요소를 사용하여 나중에 사용할 수 있도록 캐싱할 수 있습니다. 캐싱된 보안 암호를 검색하는 것이 Secret Manager에서 검색하는 것보다 빠릅니다. 또한 Secrets Manager API를 호출하는 데는 비용이 발생하므로 캐시를 사용하면 비용을 줄일 수 있습니다. 암호를 검색할 수 있는 모든 방법은 Get secrets을 참조하세요.

> **Note**
>
> 일부 언어에서는 클라이언트 측 캐싱을 위해 자체 메모리 내 암호화를 구현해야 할 수 있습니다.

구현 단계

1. Amazon CodeGuru와 같은 자동화된 도구를 사용하여 하드 코딩된 자격 증명이 포함된 코드 경로를 식별합니다.
   a. Amazon CodeGuru를 사용하여 코드 리포지토리를 스캔합니다. 검토가 완료되면 CodeGuru에서 유형=보안 암호를 필터링하여 문제가 있는 코드 줄을 찾습니다.
2. 제거하거나 대체할 수 있는 자격 증명을 식별합니다.
   a. 더 이상 필요하지 않은 자격 증명을 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 자격 증명이 필요한 경우 IAM Roles Anywhere 또는 AWS Systems Manager Hybrid Activations을 고려합니다.
3. 교체 전략을 사용해야 하는 서드파티의 장기 보안 암호의 경우 Secrets Manager를 코드에 통합하여 런타임 시 서드파티 보안 암호를 검색합니다.
   a. CodeGuru 콘솔은 검색된 자격 증명을 사용하여 Secrets Manager에서 보안 암호를 자동으로 생성할 수 있습니다.
   b. Secrets Manager의 보안 암호 검색을 애플리케이션 코드에 통합합니다.
      i. 서버리스 Lambda 함수는 언어에 구애받지 않는 Lambda 확장을 사용할 수 있습니다.
      ii. EC2 인스턴스 또는 컨테이너의 경우 AWS는 널리 사용되는 여러 프로그래밍 언어로 Secrets Manager에서 보안 암호를 검색하기 위한 클라이언트 측 코드 예제를 제공합니다.

4. 주기적으로 코드 베이스를 검토하고 다시 스캔하여 코드에 추가된 새 보안 암호가 없는지 확인합니다.
   a. 소스 코드 리포지토리에 새로운 보안 암호가 커밋되지 않도록 git-secrets와 같은 도구 사용을 고려하세요.
5. 예상치 못한 사용, 부적절한 보안 암호 액세스 또는 보안 암호 삭제 시도가 있는지 Secrets Manager 활동을 모니터링합니다.
6. 자격 증명에 대한 인적 노출을 줄입니다. 자격 증명을 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 자격 증명 사용
- SEC02-BP05 정기적으로 자격 증명 감사 및 교체

관련 문서:

- AWS Secrets Manager 시작하기
- Identity Providers and Federation
- Amazon CodeGuru Introduces Secrets Detector
- AWS Secrets Manager의 AWS Key Management Service 활용 방식
- Secret encryption and decryption in Secrets Manager
- Secrets Manager 블로그 항목
- Amazon RDS, AWS Secrets Manager와의 통합 발표

관련 비디오:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager

관련 워크숍:

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 중앙 집중식 ID 공급업체 사용

직원 ID(직원 및 계약업체)의 경우 중앙 위치에서 ID를 관리할 수 있는 ID 제공업체를 이용하세요. 이렇게 하면 단일 위치에서 액세스를 생성, 관리 및 취소하므로 여러 애플리케이션과 서비스에 대한 액세스를 더 쉽게 관리할 수 있습니다.

원하는 성과: 중앙 집중식 ID 제공업체를 통해 직원, 인증 정책(예: 다중 인증(MFA) 요구), 시스템 및 애플리케이션에 대한 권한 부여(예: 사용자의 그룹 구성원 자격 또는 특성에 따른 액세스 할당)를 중앙에서 관리할 수 있습니다. 직원은 중앙 ID 제공업체에 로그인하고 내부 및 외부 애플리케이션에 페더레이션(sSingle Sign On)하므로 사용자가 여러 자격 증명을 기억할 필요가 없습니다. ID 제공업체는 인사(HR) 시스템과 통합되므로 직원 변경 사항이 ID 제공업체와 자동으로 동기화됩니다. 예를 들어, 누군가가 조직을 떠나는 경우 페더레이션된 애플리케이션 및 시스템(AWS 포함)에 대한 액세스를 자동으로 취소할 수 있습니다. ID 제공업체에서 세부 감사 로깅을 활성화했으며 이러한 로그를 모니터링하여 비정상적인 사용자 동작이 있는지 확인하고 있습니다.

일반적인 안티 패턴:

- 페더레이션 및 Single Sign-On은 사용하지 않습니다. 직원이 여러 애플리케이션과 시스템에서 별도의 사용자 계정과 자격 증명을 생성합니다.
- ID 제공업체를 HR 시스템에 통합하는 등 직원의 ID 수명 주기를 자동화하지 않습니다. 사용자가 조직을 떠나거나 역할을 변경하면 수동 프로세스에 따라 여러 애플리케이션 및 시스템에서 기록을 삭제하거나 업데이트합니다.

이 모범 사례 확립의 이점: 중앙 집중식 ID 제공업체를 사용하면 직원 ID 및 정책을 한 곳에서 관리하고, 사용자와 그룹에 애플리케이션 액세스 권한을 할당하며, 사용자 로그인 활동을 모니터링할 수 있습니다. 인사(HR) 시스템과 통합하면 사용자가 역할을 변경하면 이러한 변경 내용이 ID 제공업체와 동기화되고 할당된 애플리케이션 및 권한이 자동으로 업데이트됩니다. 사용자가 조직을 떠나면 ID 제공업체에서 해당 ID가 자동으로 비활성화되어 페더레이션된 애플리케이션 및 시스템에 대한 액세스 권한이 취소됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS에 액세스하는 인력 사용자를 위한 지침 조직의 직원 및 계약직과 같은 인력 사용자는 직무를 수행하기 위해 AWS Management Console 또는 AWS Command Line Interface(AWS CLI)를 사용하여

AWS에 대한 액세스 권한이 필요할 수 있습니다. 중앙 집중식 ID 제공업체에서 두 가지 수준으로 AWS에 페더레이션하여 직원에게 AWS 액세스 권한을 부여할 수 있습니다. 즉, 각 AWS 계정에 대해 직접 페더레이션하거나 AWS 조직의 여러 계정에 페더레이션할 수 있습니다.

인력 사용자를 각 AWS 계정과 직접 페더레이션하려면 중앙 집중식 ID 제공업체를 사용하여 해당 계정에서 AWS Identity and Access Management에 페더레이션할 수 있습니다. IAM의 유연성을 통해 각 AWS 계정에 대해 별도의 SAML 2.0 또는 OIDC(Open ID Connect) ID 제공업체를 지원하고 액세스 제어를 위해 페더레이션 사용자 속성을 사용할 수 있습니다. 직원은 웹 브라우저를 사용하여 자격 증명(예: 암호 및 MFA 토큰 코드)을 제공하여 ID 제공업체에 로그인합니다. ID 제공업체가 브라우저에 SAML 어설션을 발행하면 이는 AWS Management Console 로그인 URL에 제출되고 사용자가 IAM 역할을 수임하여 AWS Management Console에 Single Sign-On으로 로그인할 수 있습니다. 또한 사용자는 ID 제공업체로부터 SAML 어설션을 사용해 IAM 역할을 수임하여 AWS STS로부터 AWS CLI 또는 AWS SDK에서 사용할 임시 AWS API 자격 증명을 확보할 수 있습니다.

직원을 AWS 조직의 여러 계정과 페더레이션하려면 AWS IAM Identity Center를 사용하여 AWS 계정 및 애플리케이션에 대한 직원의 액세스를 중앙에서 관리할 수 있습니다. 조직의 Identity Center를 활성화하고 자격 증명 소스를 구성합니다. IAM Identity Center는 사용자 및 그룹을 관리하는 데 사용할 수 있는 기본 자격 증명 소스 디렉터리를 제공합니다. 또는 SAML 2.0을 사용하여 외부 ID 제공업체에 연결하고 SCIM을 사용하여 사용자 및 그룹을 자동으로 프로비저닝하거나 AWS Directory Service를 사용하여 Microsoft AD 디렉터리에 연결함으로서 외부 자격 증명 소스를 선택할 수도 있습니다. 자격 증명 소스가 구성되면 권한 집합에 최소 권한 정책을 정의하여 AWS 계정 계정에 대한 사용자 및 그룹의 액세스 권한을 할당할 수 있습니다. 직원은 중앙 ID 제공업체를 통해 인증하여 AWS 액세스 포털에 로그인하고 할당된 클라우드 애플리케이션 및 AWS 계정에 Single Sign On으로 로그인할 수 있습니다. 사용자는 Identity Center를 통해 인증하고 AWS CLI 명령을 실행하기 위한 자격 증명을 얻기 위해 AWS CLI v2를 구성할 수 있습니다. 또한 Identity Center는 Amazon SageMaker AI Studio 및 AWS IoT Sitewise Monitor 포털과 같은 AWS 애플리케이션에 대한 Single Sign On을 허용합니다.

위의 지침을 따른 후에는 작업자가 AWS에서 워크로드를 관리할 때 정상적인 작업을 위해 더 이상 IAM 사용자와 그룹을 사용할 필요가 없습니다. 대신 사용자와 그룹은 AWS 외부에서 관리되며 사용자는 페더레이션형 ID로 AWS 리소스에 액세스할 수 있습니다. 페더레이션 ID는 중앙 ID 제공업체가 정의한 그룹을 사용합니다. AWS 계정에서 더 이상 필요하지 않은 IAM 그룹, IAM 사용자 그리고 장기 사용자 자격 증명(암호 및 액세스 키)을 식별하고 제거해야 합니다. IAM 자격 증명 보고서를 사용하여 사용하지 않은 자격 증명을 찾고, 해당 IAM 사용자를 삭제하며, IAM 그룹을 삭제할 수 있습니다. 조직에 서비스 제어 정책(SCP)를 적용하여 새로운 IAM 사용자 및 그룹 생성을 방지하고 페더레이션된 ID를 통해 AWS에 액세스하도록 할 수 있습니다.

> **Note**
>
> 사용자는 Automatic provisioning 설명서에 설명된 대로 SCIM 액세스 토큰의 교체를 처리할 책임이 있습니다. 또한 ID 페더레이션을 지원하는 인증서를 교체할 책임이 있습니다.

애플리케이션 사용자를 위한 지침 Amazon Cognito를 중앙 집중식 ID 제공업체로 사용하여 모바일 앱과 같은 애플리케이션 사용자의 자격 증명을 관리할 수 있습니다. Amazon Cognito는 웹 및 모바일 앱에 대한 인증, 권한 부여 및 사용자 관리를 지원합니다. Amazon Cognito는 수백만 명의 사용자로 확장 가능한 자격 증명 스토어를 제공하고, 소셜 및 엔터프라이즈 ID 페더레이션을 지원하며, 고급 보안 기능을 제공하여 사용자와 비즈니스를 보호할 수 있도록 지원합니다. 사용자 지정 웹 또는 모바일 애플리케이션을 Amazon Cognito에 통합하여 몇 분 만에 애플리케이션에 사용자 인증 및 액세스 제어를 추가할 수 있습니다. SAML 및 OIDC(Open ID Connect)와 같은 개방형 ID 표준을 기반으로 구축된 Amazon Cognito는 다양한 규정 준수 규정을 지원하고 프론트엔드 및 백엔드 개발 리소스와 통합됩니다.

구현 단계

직원이 AWS에 액세스하는 단계

- 다음 접근 방식 중 하나인 AWS 사용하여 직원을 중앙 집중식 ID 제공업체를 사용하도록 통합하세요.
  - IAM Identity Center를 사용하여 ID 제공업체와 페더레이션하여 AWS 조직 내 여러 AWS 계정에 대한 Single Sign On을 지원합니다.
  - IAM을 통해 ID 제공업체를 각 AWS 계정에 직접 연결하여 페더레이션된 세분화된 액세스를 지원합니다.
- 페더레이션 ID로 대체되는 IAM 사용자 및 그룹을 식별 및 제거합니다.

애플리케이션 사용자를 위한 단계

- Amazon Cognito를 애플리케이션에 대한 중앙 집중식 ID 제공업체로 사용합니다.
- OpenID Connect 및 OAuth를 사용하여 사용자 지정 애플리케이션을 Amazon Cognito에 통합합니다. 인증을 위해 Amazon Cognito와 같은 다양한 AWS 서비스와 통합할 수 있는 간단한 인터페이스를 제공하는 Amplify 라이브러리를 사용하여 사용자 지정 애플리케이션을 개발할 수 있습니다.

리소스

관련 모범 사례:

- SEC02-BP06 사용자 그룹 및 속성 사용
- SEC03-BP02 최소 권한 액세스 부여
- SEC03-BP06 수명 주기에 따라 액세스 관리

관련 문서:

- AWS의 ID 페더레이션
- IAM의 보안 모범 사례
- AWS Identity and Access Management Best practices
- Getting started with IAM Identity Center delegated administration
- How to use customer managed policies in IAM Identity Center for advanced use cases
- AWS CLI v2: IAM Identity Center credential provider

관련 비디오:

- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2018: Mastering Identity at Every Layer of the Cake

관련 예제:

- 워크숍: Using AWS IAM Identity Center to achieve strong identity management

관련 도구:

- AWS 보안 컴피턴시 파트너: 자격 증명 및 액세스 관리
- saml2aws

SEC02-BP05 정기적으로 자격 증명 감사 및 교체

자격 증명을 주기적으로 감사하고 교체하여 리소스에 액세스하는 데 자격 증명을 사용할 수 있는 기간을 제한합니다. 장기 자격 증명은 많은 위험을 초래하며 이러한 위험은 장기 자격 증명을 정기적으로 교체하여 줄일 수 있습니다.

원하는 성과: 자격 증명 교체를 구현하여 장기 자격 증명 사용과 관련된 위험을 줄입니다. 자격 증명 교체 정책 미준수를 정기적으로 감사하고 개선합니다.

일반적인 안티 패턴:

- 자격 증명 사용을 감사하지 않습니다.
- 장기 자격 증명을 불필요하게 사용합니다.
- 장기 자격 증명을 사용하고 정기적으로 교체하지 않습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

임시 자격 증명을 사용할 수 없으며 장기 자격 증명이 필요한 경우 자격 증명을 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 자격 증명을 사용하도록 규정해야 합니다. AWS Identity and Access Management(IAM) 사용자에서 중앙 집중식 ID로 전환하면서 자격 증명 보고서를 생성하여 사용자를 감사할 수 있습니다.

또한 ID 제공업체에서 MFA를 적용하고 모니터링하는 것이 좋습니다. 사용자가 MFA를 구성한 경우 AWS Config 규칙를 설정하거나 AWS Security Hub 보안 표준을 사용할 수 있습니다. 시스템 자격 증명에 대한 임시 자격 증명을 제공하려면 IAM Roles Anywhere를 사용하는 것이 좋습니다. IAM 역할 및 임시 자격 증명을 사용할 수 없는 상황에서는 빈번한 감사 및 교체 액세스 키가 필요합니다.

구현 단계

- 정기적으로 자격 증명 감사: ID 제공업체 및 IAM에 구성된 자격 증명을 감사하면 승인된 자격 증명만 워크로드에 액세스하도록 보장할 수 있습니다. 이러한 자격 증명에는 IAM 사용자, AWS IAM Identity Center 사용자, Active Directory 사용자 또는 다른 업스트림 ID 제공업체의 사용자가 포함될 수 있지만 이에 국한되지 않습니다. 예를 들어 퇴사하는 사람을 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. 자격 증명 보고서 및 AWS Identity and Access Management Access Analyzer를 사용하여 IAM 자격 증명 및 권한을 감사합니다. AWS 환경에서 직접 호출되는 특정 API 직접 호출에 대해 경보를 설정하도록 Amazon CloudWatch를 사용할 수 있습니다. Amazon GuardDuty는 예상치 못한 활동을 알릴 수도 있습니다. 이때 IAM 자격 증명에 대한 지나치게 허용적인 액세스 또는 의도하지 않은 액세스를 나타낼 수 있습니다.

- 정기적으로 자격 증명 교체: 임시 자격 증명을 사용할 수 없는 경우 장기 IAM 액세스 키를 정기적으로 교체합니다(최대 90일마다). 자신도 모르게 액세스 키가 의도치 않게 공개된 경우 자격 증명을 사용하여 리소스에 액세스할 수 있는 기간이 제한됩니다. IAM 사용자의 액세스 키 교체에 대한 자세한 내용은 액세스 키 교체를 참조하세요.
- IMA 권한 검토: AWS 계정의 보안을 개선하려면 모든 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 IAM 사용자를 시스템 ID로 교체: IAM Roles Anywhere를 사용하면 온프레미스 서버와 같이 기존에는 불가능했던 영역에서 역할을 사용할 수 있습니다. IAM Roles Anywhere는 신뢰할 수 있는 X.509 인증서를 사용하여 AWS에 인증하고 임시 자격 증명을 받습니다. IAM Roles Anywhere를 사용하면 장기 자격 증명이 온프레미스 환경에 더 이상 저장되지 않으므로 이러한 자격 증명을 교체할 필요가 없습니다. 만료가 가까워지면 X.509 인증서를 모니터링하고 교체해야 합니다.

리소스

관련 모범 사례:

- SEC02-BP02 임시 자격 증명 사용
- SEC02-BP03 안전하게 보안 암호 저장 및 사용

관련 문서:

- AWS Secrets Manager 시작하기
- IAM 모범 사례
- Identity Providers and Federation
- 보안 파트너 솔루션: 액세스 및 액세스 제어
- Temporary Security Credentials
- AWS 계정의 자격 증명 보고서 가져오기

관련 비디오:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP06 사용자 그룹 및 속성 사용

사용자 그룹 및 속성에 따라 권한을 정의하면 정책의 수와 복잡성을 줄여 최소 권한 원칙을 보다 간단하게 구현할 수 있습니다. 사용자 그룹을 사용하여 조직에서 수행하는 기능에 따라 한 곳에서 여러 사용자의 권한을 관리할 수 있습니다. 부서, 프로젝트 또는 위치와 같은 속성은 사용자가 유사한 기능을 수행하는 리소스의 다른 하위 세트에 대해 권한 범위 계층을 추가로 제공할 수 있습니다.

원하는 성과: 기능에 따른 권한 변경 사항을 해당 기능을 수행하는 모든 사용자에게 적용할 수 있습니다. 그룹 멤버십과 속성은 사용자 권한을 관리하므로, 개별 사용자 수준에서 권한을 관리할 필요가 줄어듭니다. ID 제공업체(idP)에서 정의한 그룹 및 속성은 AWS 환경에 자동으로 전파됩니다.

일반적인 안티 패턴:

- 개별 사용자의 권한을 관리하고 여러 사용자에 걸쳐 복제합니다.
- 지나치게 개괄적으로 그룹을 정의하여 너무 광범위한 권한을 부여합니다.
- 그룹을 너무 세밀하게 정의하여 멤버십에 대한 중복과 혼란을 야기합니다.
- 속성을 대신 사용할 수 있는 경우 리소스의 하위 세트에서 권한이 중복된 그룹을 사용합니다.
- AWS 환경에 통합되어 있는 표준화된 ID 제공업체를 통해 그룹, 속성 및 멤버십을 관리하지 않습니다.
- AWS IAM Identity Center 세션을 사용할 때 역할 체인 사용

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS 권한은 사용자, 그룹, 역할 또는 리소스와 같은 보안 주체와 관련된 정책이라는 문서에서 정의됩니다. 직무, 워크로드 및 SDLC 환경을 기반으로 권한 할당(그룹, 권한, 계정)을 구성하여 권한 관리를 확장할 수 있습니다. 직원의 경우 액세스 중인 리소스가 아닌 조직에서 사용자가 수행하는 기능을 기반으로 그룹을 정의할 수 있습니다. 예를 들어 WebAppDeveloper 그룹에는 개발 계정 내에서 Amazon CloudFront와 같은 서비스를 구성하기 위한 정책이 연결되어 있을 수 있습니다. `AutomationDeveloper` 그룹에는 `WebAppDeveloper` 그룹과 일부 중첩 권한이 있을 수 있습니다. 두 기능을 수행하는 사용자가 모두 `CloudFrontAccess` 그룹에 속하도록 하는 대신 이러한 일반적인 권한을 별도의 정책으로 캡처하여 두 그룹에 연결할 수 있습니다.

그룹 외에도 속성을 사용하여 액세스 범위를 확대할 수 있습니다. 예를 들어, `WebAppDeveloper` 그룹의 사용자에 대해 Project 속성을 지정하여 프로젝트 관련 리소스에 대한 액세스 범위를 설정할 수 있습니다. 이 기술을 사용하면 권한이 동일할 경우 상이한 프로젝트에서 작업하는 애플리케이션 개발자용 그룹을 서로 다르게 만들 필요가 없습니다. 권한 정책에서 속성을 참조하는 방법은 해당 속성이 페더레이션 프로토콜(예: SAML, OIDC 또는 SCIM)의 일부로 정의되었는지, 사용자 지정 SAML 어설션으로 정의되었는지, IAM Identity Center 내부에 설정되었는지에 관계없이 해당 소스를 기반으로 합니다.

구현 단계

1. 그룹 및 속성을 정의할 위치를 설정합니다.
   a. SEC02-BP04 중앙 집중식 ID 공급업체 사용의 지침에 따라 그룹 및 속성을 정의할 때 ID 제공업체 내부 또는 IAM Identity Center 내부에서 그룹 및 속성을 정의해야 하는지 아니면 특정 계정의 IAM 사용자 그룹을 사용해야 하는지 결정할 수 있습니다.
2. 그룹을 정의합니다.
   a. 기능 및 필요한 액세스 범위를 기준으로 그룹을 결정합니다. 계층 구조 또는 명명 규칙을 사용하여 그룹을 효과적으로 구성하는 것이 좋습니다.
   b. IAM Identity Center 내에 정의하는 경우 그룹을 만들고 권한 집합을 사용하여 원하는 수준의 액세스를 연결합니다.
   c. 외부 ID 제공업체 내에서 정의하는 경우 제공업체가 SCIM 프로토콜을 지원하는지 확인하고, IAM Identity Center 내에서 자동 프로비저닝을 활성화하는 방법을 고려하세요. 이 기능은 제공업체와 IAM Identity Center 간에 그룹 생성, 멤버십 및 삭제를 동기화합니다.
3. 속성을 정의합니다.
   a. 외부 ID 제공업체를 사용하는 경우 SCIM 및 SAML 2.0 프로토콜 모두 기본적으로 특정 속성을 제공합니다. `https://aws.amazon.com/SAML/Attributes/PrincipalTag` 속성 이름을 사용하는 SAML 어설션을 통해 추가 속성을 정의하고 전달할 수 있습니다. 사용자 지정 속성 정의 및 구성에 대한 지침은 ID 제공업체의 설명서를 참조하세요.
   b. IAM Identity Center 내부에서 역할을 정의하는 경우 속성 기반 액세스 제어(ABAC) 기능을 활성화하고 필요에 따라 속성을 정의합니다. 조직의 구조 또는 리소스 태그 지정 전략에 맞는 속성을 고려합니다.

IAM Identity Center를 통해 수임된 IAM 역할에서 IAM 역할 체인이 필요한 경우 `source-identity` 및 `principal-tags`와 같은 값은 전파되지 않습니다. 자세한 내용은 액세스 제어를 위한 속성 활성화 및 구성을 참조하세요.

1. 그룹 및 속성을 기반으로 권한 범위를 지정합니다.

a. 권한 정책에 보안 주체의 속성을 액세스 중인 리소스의 속성과 비교하는 조건을 포함하는 것을 고려해 보세요. 예를 들어 `PrincipalTag` 조건 키의 값이 같은 이름의 `ResourceTag` 키 값과 일치하는 경우에만 리소스에 대한 액세스를 허용하도록 조건을 정의할 수 있습니다.

b. ABAC 정책을 정의할 때는 ABAC 권한 부여 모범 사례 및 예시의 지침을 따르세요.

c. 조직의 요구가 변화함에 따라 그룹 및 속성 구조를 정기적으로 검토하고 업데이트하여 최적의 권한 관리를 보장합니다.

리소스

관련 모범 사례:

- SEC02-BP04 중앙 집중식 ID 공급업체 사용
- SEC03-BP02 최소 권한 액세스 부여
- COST02-BP04 그룹 및 역할 구현

관련 문서:

- IAM 모범 사례
- Manage Identities in IAM Identity Center
- What Is ABAC for AWS?
- ABAC In IAM Identity Center
- ABAC 정책 예시

관련 비디오:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC 3. 사람과 시스템에 대한 권한은 어떻게 관리하나요?

AWS 및 워크로드에 액세스해야 하는 인적 자격 증명 및 시스템 자격 증명에 대한 액세스를 제어하는 권한을 관리합니다. 권한을 통해 누가 어떤 조건에서 무엇에 액세스할 수 있는지를 제어할 수 있습니다. 구체적인 인적 및 머신 자격 증명에 권한을 설정하여 특정 리소스의 특정 서비스 작업에 대한 액세스 권한을 부여합니다. 또한 액세스 권한을 부여하려면 충족되어야 하는 조건을 지정할 수 있습니다.

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 서드파티와 리소스 공유

SEC03-BP01 액세스 요구 사항 정의

관리자, 최종 사용자 또는 기타 구성 요소별로 워크로드의 각 구성 요소 또는 리소스에 액세스해야 합니다. 각 구성 요소에 대한 액세스 권한 부여 대상을 명확하게 정의하고 적절한 ID 유형과 인증 및 권한 부여 방법을 선택합니다.

일반적인 안티 패턴:

- 애플리케이션에 보안 암호를 하드 코딩 또는 저장합니다.
- 각 사용자에 대한 사용자 지정 권한을 부여합니다.
- 장기 자격 증명을 사용합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

관리자, 최종 사용자 또는 기타 구성 요소별로 워크로드의 각 구성 요소 또는 리소스에 액세스해야 합니다. 각 구성 요소에 대한 액세스 권한 부여 대상을 명확하게 정의하고 적절한 ID 유형과 인증 및 권한 부여 방법을 선택합니다.

조직 내 AWS 계정에 대한 정기적인 액세스는 페더레이션 액세스 또는 중앙 집중식 ID 제공업체를 사용하여 제공되어야 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.

비인적 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 정책에서는 대부분의 일반적인 사용 사례를 다루는 미리 정의된 IAM 정책을 제공합니다.

AWS Secrets Manager 및 AWS Systems Manager Parameter Store와 같은 AWS 서비스를 사용하면 IAM 역할을 사용할 수 없는 경우 애플리케이션 또는 워크로드에서 보안 정보를 안전하게 분리할 수 있습니다. Secrets Manager에서 자격 증명의 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 IAM Roles Anywhere를 사용하여 AWS 외부에서 실행되는 워크로드에 대한 IAM의 임시 보안 자격 증명을 얻을 수 있습니다. 워크로드는 AWS 애플리케이션에서 AWS 리소스에 액세스하는 데 사용하는 것과 동일한 IAM 정책 및 IAM 역할을 사용할 수 있습니다.

가능한 경우, 장기적이고 정적인 자격 증명보다는 단기적이고 임시적인 자격 증명을 사용하는 것이 좋습니다. 사용자가 프로그래밍 방식 및 장기 자격 증명을 사용해야 하는 시나리오의 경우 액세스 키의 마지막 사용 정보를 사용하여 액세스 키를 교체하고 제거합니다.

사용자가 AWS Management Console 외부에서 AWS와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. 프로그래밍 방식으로 액세스를 부여하는 방법은 AWS에 액세스하는 사용자 유형에 따라 다릅니다.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 AWS IAM Identity Center을 사용하도록 AWS CLI 구성을 참조하세요. • AWS SDK, 도구,AWS API에 대해서는 AWS SDK 및

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		도구 참조 가이드에서 IAM Identity Center 인증을 참조하세요.
IAM	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	IAM 사용자 설명서의 AWS 리소스와 함께 임시 보안 인증 정보 사용에 나와 있는 지침을 따르세요.
IAM	(권장되지 않음) 장기 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 IAM 사용자 보안 인증 정보를 사용한 인증을 참조하세요. • AWS SDK와 도구에 대해서는 AWS SDK 및 도구 참조 가이드에서 장기 보안 인증 정보를 사용한 인증을 참조하세요. • AWS API에 대해서는 IAM 사용자 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.

리소스

관련 문서:

- ABAC(속성 기반 액세스 제어)
- AWS IAM Identity Center
- IAM Roles Anywhere

- AWS Managed policies for IAM Identity Center
- AWS IAM 정책 조건
- IAM 사용 사례
- 불필요한 자격 증명 제거
- 정책 작업
- How to control access to AWS resources based on AWS 계정, OU, or organization
- Identify, arrange, and manage secrets easily using enhanced search in AWS Secrets Manager

관련 비디오:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- Streamlining identity and access management for innovation

SEC03-BP02 최소 권한 액세스 부여

구체적인 조건에서 특정 리소스에 대해 일정한 작업을 수행하기 위해 사용자에게 필요한 액세스 권한만 부여합니다. 개별 사용자에 대한 권한을 정의하는 대신, 그룹 및 자격 증명 속성을 사용하여 대규모로 권한을 동적으로 설정합니다. 예를 들어 개발자 그룹이 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 성과: 사용자에게 자신의 특정 작업 기능에 필요한 최소 권한만 있습니다. 별도의 AWS 계정을 사용하여 개발자를 프로덕션 환경에서 격리합니다. 개발자가 특정 작업의 프로덕션 환경에 액세스해야 하는 경우 해당 작업을 수행하는 기간 동안에만 제한되고 제어된 액세스 권한이 부여됩니다. 프로덕션 액세스는 해당 개발자가 필요한 작업을 완료한 후 즉시 취소됩니다. 권한에 대한 정기적인 검토를 수행하고 사용자가 역할을 변경하거나 조직을 떠날 때와 같이 권한이 더 이상 필요하지 않은 경우 즉시 권한을 취소합니다. 관리자 권한을 신뢰할 수 있는 소규모 그룹으로 제한하여 위험 노출을 줄입니다. 머신 또는 시스템 계정에는 의도한 작업을 수행하는 데 필요한 최소 권한만 부여합니다.

일반적인 안티 패턴:

- 사용자에게 기본적으로 관리자 권한을 부여합니다.
- 일상 활동에 루트 사용자 계정을 사용합니다.
- 적절한 범위 조정 없이 지나치게 허용적인 정책을 생성합니다.
- 권한 검토는 자주 수행되지 않으므로 권한이 점차 커지는 상황이 발생합니다.

- 환경 격리 또는 권한 관리를 위해 속성 기반 액세스 제어에만 의존합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

최소 권한의 원칙은 자격 증명에서 특정 작업을 완수하는 데 필요한 최소한의 활동만 수행하도록 허용해야 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. 기본적으로 IAM 사용자와 역할에는 어떠한 권한도 없습니다. 루트 사용자는 기본적으로 전체 액세스 권한을 가지므로 엄격하게 제어 및 모니터링하고 루트 액세스 권한이 필요한 작업에만 사용해야 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 참이어야 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 CloudFormation과 같이 사용자를 대신하여 AWS 서비스에서 제출하는 요청을 제어할 수 있습니다. 다양한 정책 유형을 계층화하여 심층 방어를 설정하고 사용자의 권한 전반을 제한할 수 있습니다. 나아가 어떤 조건에서 어떤 권한을 허용할지도 제한할 수도 있습니다. 예를 들어 워크로드 팀이 구축하는 시스템에 대해 자체 IAM 정책을 만들도록 허용하되, 부여할 수 있는 최대 권한을 제한하기 위해 권한 경계를 적용할 경우에만 허용할 수 있습니다.

구현 단계

- 최소 권한 정책 구현: IAM 그룹 및 역할에 최소 권한이 적용된 액세스 정책을 할당하여 사용자별로 정의한 역할 또는 기능을 반영합니다.
- 별도의 AWS 계정을 통해 개발 및 프로덕션 환경 격리: 개발 및 프로덕션 환경에 별도의 AWS 계정을 사용하고 서비스 제어 정책, 리소스 정책 및 자격 증명 정책을 사용하여 이들 간의 액세스를 제어합니다.
- API 사용에 대한 기본 정책: AWS CloudTrail 로그를 검토하여 필요한 권한을 결정하는 한 가지 방법입니다. 이 검토를 사용해 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 마지막 액세스 정보를 추적할 수 있습니다.

- 작업 함수에 AWS 관리형 정책 사용 고려: 세분화된 권한 정책을 생성하기 시작할 때 결제, 데이터베이스 관리자 및 데이터 과학자와 같은 일반적인 작업 역할에 AWS 관리형 정책을 사용하는 것이 도움이 될 수 있습니다. 이러한 정책은 최소 권한 정책을 구현하는 방법을 결정하는 동시에 사용자의 액세스 범위를 좁히는 데 도움이 될 수 있습니다.
- 불필요한 권한 제거: 미사용 IAM 엔터티, 자격 증명 및 권한을 감지하고 제거하여 최소 권한 원칙을 달성합니다. IAM Access Analyzer를 사용하여 외부 및 미사용 액세스를 식별할 수 있으며, IAM Access Analyzer 정책 생성은 권한 정책을 미세 조정하는 데 도움이 될 수 있습니다.
- 프로덕션 환경에 대한 사용자 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 ID 기반 정책을 통해 IAM 엔터티에 부여할 수 있는 최대 권한을 설정하는 관리형 정책을 사용하는 기능입니다. 엔터티의 권한 경계는 자격 증명 기반 정책 및 관련 권한 경계 모두에서 허용되는 작업만 수행하도록 허용합니다.
- 속성 기반 액세스 제어 및 리소스 태그를 사용하여 액세스 구체화: 리소스 태그를 사용하는 속성 기반 액세스 제어(ABAC)를 사용하여 지원되는 경우 권한을 구체화할 수 있습니다. 보안 주체 태그를 리소스 태그와 비교하여 정의한 사용자 지정 차원에 따라 액세스를 구체화하는 ABAC 모델을 사용할 수 있습니다. 이 접근 방식은 조직의 권한 정책 수를 간소화하고 줄일 수 있습니다.
  - 보안 주체와 리소스를 모두 AWS 조직에서 소유한 경우에만 ABAC를 액세스 제어에 사용하는 것이 좋습니다. 외부 당사자는 자체 보안 주체 및 리소스에 대해 조직과 동일한 태그 이름 및 값을 사용할 수 있습니다. 외부 당사자 보안 주체 또는 리소스에 대한 액세스 권한을 부여하기 위해 이러한 이름-값 페어에만 의존하는 경우 의도하지 않은 권한을 제공할 수 있습니다.
- AWS Organizations에 서비스 제어 정책 사용: 서비스 제어 정책은 조직의 구성원 계정에 대해 사용 가능한 최대 권한을 중앙에서 제어합니다. 중요한 점은 서비스 제어 정책을 사용하여 구성원 계정의 루트 사용자 권한을 제한할 수 있다는 사실입니다. AWS Organizations를 보강하는 권장 관리 제어 기능을 제공하는 AWS Control Tower를 사용하는 것도 고려해 보세요. Control Tower 내에서 자체 제어 기능을 정의할 수도 있습니다.
- 조직을 위한 사용자 수명 주기 정책 수립: 사용자 수명 주기 정책은 사용자가 AWS에 온보딩하거나, 작업 역할 또는 범위가 변경되거나, 더 이상 AWS에 액세스할 필요가 없을 때 수행할 작업을 정의합니다. 사용자 수명 주기의 각 단계에서 권한 검토를 수행하여 권한이 적절하게 제한되는지 확인하고 권한이 커지는 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 사용하여 조직 내 사용자 책임에 따라 권한을 정의하고 분리할 수 있습니다. 개별 사용자나 역할에 권한을 직접 적용하는 대신 그룹을 사용합니다.

리소스

관련 문서:

- 최소 권한 적용
- Permissions boundaries for IAM entities
- Techniques for writing least privilege IAM policies
- IAM Access Analyzer makes it easier to implement least privilege permissions by generating IAM policies based on access activity
- Delegate permission management to developers by using IAM permissions boundaries
- Refining Permissions using last accessed information
- IAM policy types and when to use them
- IAM 정책 시뮬레이터로 IAM 정책 테스트
- Guardrails in AWS Control Tower
- Zero Trust architectures: An AWS perspective
- How to implement the principle of least privilege with CloudFormation StackSets
- ABAC(속성 기반 액세스 제어)
- Reducing policy scope by viewing user activity
- View role access
- Use Tagging to Organize Your Environment and Drive Accountability
- AWS Tagging Strategies
- AWS 리소스에 태그 지정

관련 비디오:

- Next-generation permissions management

- Zero Trust: An AWS perspective

SEC03-BP03 긴급 액세스 프로세스 설정

중앙 집중식 ID 제공업체에 문제가 발생할 경우 예상치 못한 상황에서 워크로드에 긴급 액세스할 수 있는 프로세스를 만드세요.

긴급 상황을 초래할 수 있는 다양한 장애 모드에 대한 프로세스를 설계해야 합니다. 예를 들어, 일반적인 상황에서는 직원이 중앙 집중식 ID 제공업체를 사용하여 클라우드로 페더레이션함으로써(SEC02-BP04) 워크로드를 관리합니다. 그러나 중앙 집중식 ID 제공업체에 장애가 발생하거나 클라우드에서의 페더레이션 구성이 수정되면 직원이 클라우드로 페더레이션하지 못할 수 있습니다. 긴급 액세스 프로세스를 통해 권한 있는 관리자는 대체 수단(예: 대체 형태의 페더레이션 또는 직접 사용자 액세스)을 통해 클라우드 리소스에 액세스하여 페더레이션 구성 또는 워크로드 관련 문제를 해결할 수 있습니다. 긴급 액세스 프로세스는 일반 페더레이션 메커니즘이 복원될 때까지 사용됩니다.

원하는 성과:

- 긴급 상황으로 간주되는 장애 모드를 정의하고 문서화했습니다. 일반적인 상황과 사용자가 워크로드를 관리하기 위해 사용하는 시스템을 고려하세요. 이러한 각 종속성이 어떻게 실패하여 긴급 상황을 초래할 수 있는지 생각해 보세요. 장애 가능성을 최소화하기 위해 장애 모드를 식별하고 보다 탄력적인 시스템을 설계하는 데 유용한 신뢰성 원칙의 질문 및 모범 사례를 찾아볼 수 있습니다.
- 장애를 긴급 상황으로 확인하기 위해 따라야 하는 단계를 문서화했습니다. 예를 들어 ID 관리자에게 기본 및 대기 ID 제공업체의 상태를 확인하고 둘 다 사용할 수 없는 경우 ID 제공업체 장애에 대한 긴급 이벤트를 선언하도록 요청할 수 있습니다.
- 각 유형의 긴급 또는 장애 모드에 맞는 긴급 액세스 프로세스를 정의했습니다. 구체적으로 설명하면 모든 유형의 긴급 상황에서 일반 프로세스를 과도하게 사용하려는 사용자의 유혹을 줄일 수 있습니다. 긴급 액세스 프로세스는 각 프로세스를 사용해야 하는 상황과 반대로 프로세스를 사용하지 않아야 하는 상황을 설명하고 적용될 수 있는 대체 프로세스를 가리킵니다.
- 프로세스는 빠르고 효율적으로 따를 수 있는 상세한 지침과 플레이북과 함께 잘 문서화되어 있습니다. 긴급 상황은 사용자에게 스트레스를 주는 시간이 될 수 있고 사용자들이 극심한 시간 압박을 받을 수 있다는 점을 기억하세요. 따라서 프로세스를 최대한 단순하게 설계하세요.

일반적인 안티 패턴:

- 긴급 액세스 절차가 제대로 문서화되고 테스트되지 않습니다. 사용자는 긴급 상황에 대비하지 않고 긴급 상황 발생 시 즉흥적인 프로세스를 따릅니다.

- 긴급 액세스 프로세스는 일반 액세스 메커니즘과 동일한 시스템(예: 중앙 집중식 ID 제공업체)을 기반으로 합니다. 즉, 이러한 시스템에 장애가 발생하면 정상 액세스 메커니즘과 긴급 액세스 메커니즘에 모두 영향을 미치고 장애 복구 능력이 저하될 수 있습니다.
- 긴급 액세스 프로세스는 비응급 상황에서 사용됩니다. 예를 들어, 사용자는 파이프라인을 통해 변경 사항을 제출하는 것보다 직접 변경하는 것이 더 쉽기 때문에 긴급 액세스 프로세스를 자주 오용합니다.
- 긴급 액세스 프로세스는 프로세스를 감사하기에 충분한 로그를 생성하지 않거나 프로세스의 오용 가능성에 대해 경고할 수 있도록 로그를 모니터링하지 않습니다.

이 모범 사례 확립의 이점:

- 잘 문서화되고 테스트를 거친 긴급 액세스 프로세스를 갖추면 사용자가 긴급 상황에 대응하고 해결하는 데 걸리는 시간을 줄일 수 있습니다. 이를 통해 가동 중지 시간이 줄어들고 고객에게 제공하는 서비스의 가용성이 향상될 수 있습니다.
- 각 긴급 액세스 요청을 추적하고, 프로세스를 긴급 상황이 아닌 이벤트에 악용하려는 무단 시도를 감지하고 경고를 보낼 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

이 섹션에서는 모든 장애 모드에 적용되는 공통 지침부터 시작하여 장애 모드 유형에 따른 구체적인 지침에 이어 AWS에 배포된 워크로드와 관련된 여러 장애 모드에 대한 긴급 액세스 프로세스를 만드는 방법에 대한 지침을 제공합니다.

모든 장애 모드에 대한 공통 지침

장애 모드에 대한 긴급 액세스 프로세스를 설계할 때는 다음 사항을 고려하세요.

- 프로세스의 사전 조건과 전제 조건, 즉 프로세스를 사용해야 하는 시기와 사용하지 말아야 하는 경우를 문서화하세요. 장애 모드를 자세히 설명하고 다른 관련 시스템의 상태와 같은 가정을 문서화하는 데 도움이 됩니다. 예를 들어 장애 모드 2의 프로세스에서는 ID 제공업체를 사용할 수 있지만 설정된 AWS 구성이 수정되었거나 만료된 것으로 가정합니다.
- 긴급 액세스 프로세스에 필요한 리소스를 미리 생성합니다(SEC10-BP05). 예를 들어, 모든 워크로드 계정에서 IAM 사용자 및 역할과 크로스 계정 IAM 역할을 사용하여 긴급 액세스 AWS 계정을 미리 생성합니다. 이를 통해 긴급 상황 발생 시 이러한 리소스가 준비되어 있고 사용할 수 있는지 확인할 수 있습니다. 리소스를 미리 생성하면 긴급 상황에서 사용할 수 없는 AWS 컨트롤 플레인 API(AWS 리

소스 생성 및 수정에 사용됨)에 대한 종속성이 없습니다. 또한 IAM 리소스를 미리 생성하면 최종 일관성으로 인한 잠재적 지연을 고려할 필요가 없습니다.

- 사고 관리 계획의 일부로 긴급 액세스 프로세스를 포함하세요(SEC10-BP02). 긴급 상황이 어떻게 추적되고 동료 팀, 경영진 그리고 해당하는 경우 외부 고객 및 비즈니스 파트너와 같은 조직 내 다른 사람에게 전달되는지 문서화하세요.
- 기존 서비스 요청 워크플로 시스템(있는 경우)에서 긴급 액세스 요청 프로세스를 정의하세요. 일반적으로 이러한 워크플로우 시스템에서는 접수 양식을 만들어 요청에 대한 정보를 수집하고, 워크플로의 각 단계를 통해 요청을 추적하며, 자동 승인 단계와 수동 승인 단계를 모두 추가할 수 있습니다. 각 요청을 인시던트 관리 시스템에서 추적되는 해당 긴급 이벤트와 연관시키세요. 긴급 액세스를 위한 통일된 시스템을 구축하면 단일 시스템에서 이러한 요청을 추적하고, 사용 추세를 분석하며, 프로세스를 개선할 수 있습니다.
- 긴급 액세스 프로세스는 승인된 사용자만 시작할 수 있고 필요에 따라 사용자의 동료 또는 경영진의 승인이 필요한지 확인하세요. 승인 절차는 업무 시간 내외에서 모두 효과적으로 운영되어야 합니다. 1차 승인자를 사용할 수 없고 승인될 때까지 관리망에 에스컬레이션되는 경우 승인 요청을 2차 승인자가 허용할 수 있는 방법을 정의합니다.
- 긴급 액세스 프로세스 및 메커니즘에 대한 강력한 로깅, 모니터링 및 알림 메커니즘을 구현합니다. 모든 긴급 액세스 시도 성공 및 실패에 대한 자세한 감사 로그를 생성합니다. 인시던트 관리 시스템의 진행 중인 긴급 이벤트와 활동의 상관관계를 파악하고, 작업이 예상 기간을 벗어나거나 정상 운영 중에 긴급 액세스 계정이 사용되는 경우 알림을 시작합니다. 긴급 절차는 백도어로 간주될 수 있으므로 긴급 상황에만 긴급 액세스 계정에 액세스해야 합니다. 보안 정보 및 이벤트 관리(SIEM) 도구 또는 AWS Security Hub와 통합하여 긴급 액세스 기간 동안 모든 활동을 보고하고 감사할 수 있습니다. 정상 작업으로 돌아가면 긴급 액세스 자격 증명을 자동으로 교체하고 관련 팀에 알립니다.
- 긴급 액세스 프로세스를 정기적으로 테스트하여 단계가 명확한지 확인하고 올바른 액세스 수준을 빠르고 효율적으로 부여하세요. 긴급 액세스 프로세스는 인시던트 대응 시뮬레이션(SEC10-BP07) 및 재해 복구 테스트(REL13-BP03)의 일부로 테스트해야 합니다.

장애 모드 1: AWS로 페더레이션에 사용된 ID 제공업체를 사용할 수 없는 경우

SEC02-BP04 중앙 집중식 ID 공급업체 사용에서 설명한 대로, 중앙 집중식 ID 제공업체를 통해 직원을 페데레이션하여 AWS 계정에 액세스 권한을 부여하는 것이 좋습니다. IAM Identity Center를 사용하여 AWS 조직 내 여러 AWS 계정으로 페더레이션하거나 IAM을 사용하여 개별 AWS 계정에 페더레이션할 수 있습니다. 두 경우 모두, 직원이 Single Sign On을 위해 AWS 로그인 엔드포인트로 리디렉션되기 전에 중앙 집중식 ID 제공업체를 통해 인증합니다.

드문 경우지만 중앙 집중식 ID 제공업체를 사용할 수 없는 경우 직원은 AWS 계정에 페더레이션하거나 워크로드를 관리할 수 없습니다. 이 긴급 상황에서 중앙 집중식 ID 제공업체가 다시 온라인 상태가 될

때까지 기다릴 수 없는 중요한 작업을 수행할 수 있도록 소수의 관리자가 AWS 계정에 액세스할 수 있는 긴급 액세스 프로세스를 제공할 수 있습니다. 예를 들어 ID 제공업체를 4시간 동안 사용할 수 없는 경우, 예상치 못한 고객 트래픽 급증에 대처하려면 Production 계정의 Amazon EC2 Auto Scaling 그룹 상한선을 수정해야 합니다. 긴급 관리자는 긴급 액세스 프로세스에 따라 특정 프로덕션 AWS 계정에 대한 액세스 권한을 얻고 필요한 사항을 변경해야 합니다.

긴급 액세스 프로세스는 긴급 액세스 AWS 계정에만 사용되고 긴급 액세스 프로세스를 지원하는 AWS 리소스(예: IAM 역할 및 IAM 사용자)가 있는 미리 생성된 긴급 액세스를 기반으로 합니다. 정상적으로 운영되는 동안에는 아무도 긴급 액세스 계정에 접속해서는 안 되며 이 계정의 오용을 모니터링하고 경고해야 합니다(자세한 내용은 이전 공통 지침 섹션 참조).

긴급 액세스 계정에는 긴급 액세스가 필요한 AWS 계정에서 크로스 계정 역할을 수임할 수 있는 권한이 있는 긴급 액세스 IAM 역할이 있습니다. 이러한 IAM 역할은 긴급 계정의 IAM 역할을 신뢰하는 신뢰 정책으로 미리 생성되고 구성됩니다.

긴급 액세스 프로세스는 다음 방법 중 하나를 사용할 수 있습니다.

- 긴급 액세스 계정에서 관련 강력한 암호 및 MFA 토큰을 사용하여 긴급 관리자용 IAM 사용자 세트를 미리 생성할 수 있습니다. 이러한 IAM 사용자는 IAM 역할을 수임할 권한을 보유하고 있습니다. 이를 통해 이후 긴급 액세스가 필요한 AWS 계정에 크로스 계정 액세스를 허용합니다. 가능한 한 적은 수의 사용자를 생성하고 각 사용자를 한 명의 긴급 관리자에게 할당하는 것이 좋습니다. 긴급 상황 발생 시 긴급 관리자 사용자는 암호와 MFA 토큰 코드를 사용하여 긴급 액세스 계정에 로그인하고, 긴급 계정에서 긴급 액세스 IAM 역할로 전환하며, 마지막으로 워크로드 계정의 긴급 액세스 IAM 역할로 전환하여 긴급 변경 작업을 수행합니다. 이 접근 방식의 장점은 각 IAM 사용자가 한 명의 긴급 관리자로 할당되며 CloudTrail 이벤트를 검토하여 어떤 사용자가 로그인했는지 알 수 있다는 점입니다. 단점은 연결된 장기 암호와 MFA 토큰을 사용하여 여러 IAM 사용자를 유지 관리해야 한다는 점입니다.
- 긴급 액세스 AWS 계정 루트 사용자를 사용하여 긴급 액세스 계정에 로그인하고 긴급 액세스를 위한 IAM 역할을 수임하며 워크로드 계정에서 크로스 계정 역할을 수임할 수 있습니다. 루트 사용자에게는 강력한 암호와 여러 MFA 토큰을 설정하는 것이 좋습니다. 또한 강력한 인증 및 권한 부여를 시행하는 안전한 엔터프라이즈 자격 증명 볼트에 암호와 MFA 토큰을 저장하는 것이 좋습니다. 암호 및 MFA 토큰 재설정 요소를 보호해야 합니다. 계정의 이메일 주소를 클라우드 보안 관리자가 모니터링하는 이메일 배포 목록으로 설정하고 계정의 전화번호를 보안 관리자가 모니터링하는 공유 전화번호로 설정합니다. 이 접근 방식의 장점은 한 세트의 루트 사용자 자격 증명을 관리할 수 있다는 것입니다. 단점은 공유 사용자이기 때문에 여러 관리자가 루트 사용자로 로그인할 수 있다는 것입니다. 엔터프라이즈 볼트 로그 이벤트를 감사하여 루트 사용자 암호를 체크아웃한 관리자를 식별해야 합니다.

장애 모드 2: AWS의 ID 제공업체 구성이 수정되었거나 만료된 경우

인력 사용자가 AWS 계정에 페더레이션할 수 있도록 하려면 외부 ID 제공업체를 사용하여 IAM Identity Center를 구성하거나 IAM 자격 증명 공급자를 생성할 수 있습니다(SEC02-BP04). 일반적으로 ID 제공업체가 제공한 SAML 메타데이터 XML 문서를 가져와서 이를 구성합니다. 메타데이터 XML 문서에는 ID 제공업체가 SAML 어설션에 서명하는 데 사용하는 개인 키에 해당하는 X.509 인증서가 포함되어 있습니다.

AWS 측의 이러한 구성은 관리자가 실수로 수정하거나 삭제할 수 있습니다. 또 다른 시나리오에서는 AWS로 가져온 X.509 인증서가 만료되고 새 인증서가 포함된 새 메타데이터 XML을 AWS에 아직 가져오지 않은 경우가 있습니다. 두 시나리오 모두 인력 사용자에 대한 AWS 페더레이션을 중단하여 긴급 상황을 초래할 수 있습니다.

이러한 긴급 상황의 경우 ID 관리자에게 페더레이션 문제를 해결할 수 있는 AWS 액세스 권한을 제공할 수 있습니다. 예를 들어, ID 관리자는 긴급 액세스 프로세스를 사용하여 AWS 계정에 긴급 액세스로 로그인하고, Identity Center 관리자 계정의 역할로 전환하며, 페더레이션을 다시 활성화하기 위해 ID 제공업체로부터 최신 SAML 메타데이터 XML 문서를 가져와서 외부 ID 제공업체 구성을 업데이트합니다. 페더레이션이 수정되면 인력 사용자는 계속해서 일반 운영 프로세스를 사용하여 워크로드 계정에 페더레이션합니다.

이전 장애 모드 1에 설명된 접근 방식에 따라 긴급 액세스 프로세스를 만들 수 있습니다. ID 관리자에게 Identity Center 관리자 계정에만 액세스하고 해당 계정의 Identity Center에서 작업을 수행할 수 있는 최소 권한 권한을 부여할 수 있습니다.

장애 모드 3: Identity Center 중단

IAM Identity Center의 예상치 못한 상황이나 AWS 리전 중단이 발생할 경우 AWS Management Console에 임시 액세스를 제공하는 데 사용할 수 있는 구성을 설정하는 것이 좋습니다.

긴급 액세스 프로세스는 ID 제공업체로부터 긴급 계정의 IAM으로 직접 페더레이션을 사용합니다. 프로세스 및 설계 고려 사항에 대한 자세한 내용은 Set up emergency access to the AWS Management Console을 참조하세요.

구현 단계

모든 장애 모드에 대한 공통 단계

- 긴급 액세스 프로세스 AWS 계정 전용 프로세스를 생성합니다. 계정에 필요한 IAM 리소스(예: IAM 역할 또는 IAM 사용자) 및 선택적으로 IAM ID 제공업체를 미리 생성합니다. 또한 긴급 액세스 계정의 해당 IAM 역할과의 신뢰 관계를 사용하여 워크로드 AWS 계정에서 크로스 계정 IAM 역할을 미리

생성합니다. AWS Organizations에서 AWS CloudFormation StackSets를 사용하여 조직의 구성원 계정에서 이러한 리소스를 만들 수 있습니다.

- AWS Organizations 서비스 제어 정책(SCP)을 생성하여 구성원 AWS 계정에서 크로스 계정 IAM 역할의 삭제 및 수정을 거부합니다.
- 긴급 액세스 AWS 계정에 대해 CloudTrail을 활성화하고 로그 컬렉션 AWS 계정에서 중앙 S3 버킷으로 트레일 이벤트를 전송합니다. AWS Control Tower를 사용하여 AWS 다중 계정 환경을 설정하고 관리하는 경우 AWS Control Tower를 사용하여 생성하거나 AWS Control Tower에 등록한 모든 계정에서는 기본적으로 CloudTrail이 활성화되고 전용 로그 아카이브 AWS 계정의 S3 버킷으로 전송됩니다.
- 긴급 IAM 역할별로 콘솔 로그인 및 API 활동과 일치하는 EventBridge 규칙을 생성하여 긴급 액세스 계정의 활동을 모니터링합니다. 인시던트 관리 시스템에서 추적되는 진행 중인 긴급 이벤트 외부에서 활동이 발생하는 경우 보안 운영 센터에 알림을 보냅니다.

장애 모드 1: AWS로 페더레이션에 사용된 ID 제공업체를 사용할 수 없는 경우 및 장애 모드 2: AWS의 ID 제공업체 구성이 수정되었거나 만료된 경우의 추가 단계

- 긴급 액세스를 위해 선택한 메커니즘에 따라 리소스를 미리 생성하세요.
  - IAM 사용자 사용: 강력한 암호 및 관련 MFA 디바이스를 사용하여 IAM 사용자를 미리 생성하세요.
  - 긴급 계정 루트 사용자 사용: 강력한 암호로 루트 사용자를 구성하고 엔터프라이즈 자격 증명 저장소에 암호를 저장합니다. 여러 물리적 MFA 디바이스를 루트 사용자와 연결하고 긴급 관리자 팀원이 빠르게 액세스할 수 있는 위치에 디바이스를 저장합니다.

장애 모드 3: Identity Center 중단의 추가 단계

- AWS Management Console에 대한 긴급 액세스 설정에서 설명한 대로, 긴급 액세스 AWS 계정에서는 ID 제공업체로부터 직접 SAML 페더레이션을 활성화하도록 IAM ID 제공업체를 생성합니다.
- IdP에 구성원 없이 긴급 운영 그룹을 만드세요.
- 긴급 액세스 계정에서 긴급 운영 그룹에 해당하는 IAM 역할을 생성합니다.

리소스

관련 Well-Architected 모범 사례:

- SEC02-BP04 중앙 집중식 ID 공급업체 사용

- SEC03-BP02 최소 권한 액세스 부여
- SEC10-BP02 인시던트 관리 계획 개발
- SEC10-BP07 게임 데이 진행

관련 문서:

- Set up emergency access to the AWS Management Console
- SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기
- Break glass access

관련 비디오:

- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

관련 예제:

- AWS Break Glass Role
- AWS customer playbook framework
- AWS incident response playbook samples

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 ID와 권한을 지속적으로 모니터링하고 제거합니다.

원하는 성과: 권한 정책은 최소 권한 원칙을 준수해야 합니다. 직무와 역할이 더 잘 정의됨에 따라 권한 정책을 검토하여 불필요한 권한을 제거해야 합니다. 이 접근 방식은 자격 증명이 의도치 않게 노출되거나 권한 부여 없이 액세스되는 경우 영향 범위를 줄입니다.

일반적인 안티 패턴:

- 사용자에게 기본적으로 관리자 권한을 부여합니다.
- 전체 관리자 권한은 아니지만 과도하게 허용적인 정책을 생성합니다.

- 더 이상 필요하지 않은 권한 정책을 유지합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 자격 증명을 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. ABAC(속성 기반 액세스 제어)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 Analyzer를 사용하면 Amazon Simple Storage Service(S3) 버킷 또는 IAM 역할과 같은 조직 및 계정 내 리소스 중 외부 엔터티와 공유되는 리소스를 식별할 수 있습니다.
- IAM Access Analyzer 정책 생성 사용: IAM Access Analyzer 정책 생성을 통해 IAM 사용자 또는 역할의 액세스 활동에 따라 세분화된 권한 정책을 생성할 수 있습니다.
- 프로덕션 전 하위 환경의 권한 테스트: 먼저 덜 중요한 샌드박스 및 개발 환경을 사용하여 IAM Access Analyzer를 사용하여 다양한 작업 기능에 필요한 권한을 테스트하는 것으로 시작합니다. 그런 다음 프로덕션에 적용하기 전에 테스트, 품질 보증 및 스테이징 환경 전반에서 이러한 권한을 점진적으로 더 엄격하게 하고 검증합니다. 서비스 제어 정책(SCP)이 부여된 최대 권한을 제한하여 가드레일을 적용하므로 하위 환경은 처음에 더 완화된 권한을 가질 수 있습니다.
- IAM 사용자 및 역할에 대해 허용되는 기간 및 사용 정책 결정: 마지막으로 액세스한 타임스탬프를 사용하여 사용되지 않는 사용자 및 역할을 식별하고 제거합니다. 서비스 및 작업의 마지막 액세스 정

보를 검토하여 특정 사용자 및 역할에 대한 권한을 식별하고 범위를 지정합니다. 예를 들어 마지막 액세스 정보를 사용하면 애플리케이션 역할에 필요한 특정 Amazon S3 작업을 식별하여 그러한 작업으로만 역할의 액세스 권한을 제한할 수 있습니다. 마지막 액세스 정보 기능은 AWS Management Console에서 제공되며, 프로그래밍 방식으로 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 방법을 고려하세요.

리소스

관련 문서:

- Grant least privilege
- 불필요한 자격 증명 제거
- What is AWS CloudTrail?
- 정책 작업
- DynamoDB의 로깅 및 모니터링
- Amazon S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 사용
- AWS 계정의 자격 증명 보고서 가져오기

관련 비디오:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

SEC03-BP05 조직에 대한 권한 가드레일 정의

권한 가드레일을 사용하여 보안 주체에 부여할 수 있는 사용 가능한 권한의 범위를 줄이세요. 권한 정책 평가 체인에는 권한 부여 결정을 내릴 때 보안 주체의 유효 권한을 결정하기 위한 가드레일이 포함됩니다. 계층 기반 접근 방식을 사용하여 가드레일을 정의할 수 있습니다. 가드레일 중 일부는 조직 전체에 광범위하게 적용하고 일부는 임시 액세스 세션에 세부적으로 적용하세요.

원하는 성과: 별도의 AWS 계정을 사용하여 환경을 명확하게 격리할 수 있습니다. 서비스 제어 정책(SCP)은 조직 전체의 권한 가드레일을 정의하는 데 사용됩니다. 더 포괄적인 가드레일은 조직 루트에 가장 근접한 계층 수준에서 설정되고, 더 엄격한 가드레일은 개별 계정 수준에 더 가깝게 설정됩니다.

지원되는 경우 리소스 정책은 보안 주체가 리소스에 액세스하기 위해 충족해야 하는 조건을 정의합니다. 또한, 리소스 정책은 적절한 경우 허용 가능한 작업 집합의 범위를 세분화합니다. 권한 경계는 워크로드 권한을 관리하는 보안 주체에 부여되어 권한 관리를 개별 워크로드 소유자에게 위임합니다.

일반적인 안티 패턴:

- AWS Organization 내에서 구성원 AWS 계정을 생성하지만, 루트 자격 증명에 이용할 수 있는 사용 및 권한을 제한하기 위해 SCP를 사용하지 않습니다.
- 최소 권한을 기준으로 권한을 할당하지만, 부여할 수 있는 최대 권한 집합에는 가드레일을 설정하지 않습니다.
- AWS IAM의 암묵적 거부 기반에 의존하여 권한을 제한하고, 정책이 원치 않는 명시적 허용 권한을 부여하지 않을 것이라고 신뢰합니다.
- 동일한 AWS 계정에서 여러 워크로드 환경을 실행한 후 VPC, 태그 또는 리소스 정책 등의 메커니즘에 의존하여 권한 경계를 적용합니다.

이 모범 사례 확립의 이점: 권한 가드레일은 권한 정책이 허용을 시도하더라도 원하지 않는 권한을 부여할 수 없다는 확신을 심어줍니다. 이를 통해 고려해야 할 권한의 최대 범위를 줄여 권한 정의 및 관리를 단순화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

계층 기반 접근 방식을 사용하여 조직의 권한 가드레일을 정의하는 것이 좋습니다. 이 접근 방식은 추가 계층이 적용될 때 가능한 최대 권한 집합을 체계적으로 줄입니다. 이렇게 하면 최소 권한 원칙에 따라 액세스 권한을 부여하여 잘못된 정책 구성으로 인해 의도하지 않은 액세스가 발생할 위험을 줄일 수 있습니다.

권한 가드레일을 구축하는 첫 단계는 워크로드와 환경을 별도의 AWS 계정으로 분리하는 것입니다. 한 계정의 보안 주체는 명시적인 권한 없이는 다른 계정의 리소스에 액세스할 수 없습니다. 이는 두 계정이 동일한 AWS 조직 또는 동일한 조직 단위(OU)에 속하더라도 마찬가지입니다. OU를 사용하여 관리하려는 계정을 단일 단위로 그룹화할 수 있습니다.

다음 단계는 조직 구성원 계정 내에서 보안 주체에 부여할 수 있는 최대 권한 집합을 줄이는 것입니다. 이를 위해 OU 또는 계정에 적용 가능한 서비스 제어 정책(SCP)을 사용할 수 있습니다. SCP는 특정

AWS 리전에 대한 액세스 제한과 같은 일반적인 액세스 제어를 시행하여 리소스 삭제를 방지하거나 잠재적으로 위험한 서비스 작업을 비활성화하도록 할 수 있습니다. 조직의 루트에 적용하는 SCP는 구성원 계정에만 영향을 주고 관리 계정에는 영향을 주지 않습니다. SCP는 조직 내 보안 주체만 관리합니다. SCP는 리소스에 액세스하는 조직 외부의 보안 주체를 관리하지 않습니다.

AWS Control Tower를 사용하는 경우 제어 및 랜딩 존을 권한 가드레일 및 다중 계정 환경의 기반으로 활용할 수 있습니다. 랜딩 존은 다양한 워크로드 및 애플리케이션에 대한 별도의 계정을 갖춘 사전 구성된 안전한 기준 환경을 제공합니다. 가드레일은 서비스 제어 정책(SCP), AWS Config 규칙 및 기타 구성의 조합을 통해 보안, 운영 및 규정 준수에 대한 필수 제어를 적용합니다. 그러나 사용자 지정 조직 SCP와 함께 Control Tower 가드레일 및 랜딩 존을 사용하는 경우 충돌을 방지하고 적절한 거버넌스를 보장하기 위해 AWS 설명서에 설명된 모범 사례를 따르는 것이 중요합니다. Control Tower 환경 내에서 SCP, 계정 및 조직 단위(OU)를 관리하는 방법에 대한 자세한 권장 사항은 의 AWS Control Tower guidance for AWS Organizations 섹션을 참조하세요.

이러한 지침을 준수하면 Control Tower의 가드레일, 랜딩 존 및 사용자 지정 SCP를 효과적으로 활용하는 동시에 잠재적 충돌을 완화하고 다중 계정 AWS 환경에 대한 적절한 거버넌스 및 제어를 보장할 수 있습니다.

다음 단계는 IAM 리소스 정책을 사용하여 보안 주체 대행이 충족해야 하는 모든 조건과 함께 관리하는 리소스에서 수행할 수 있는 작업 범위를 지정하는 것입니다. 이는 보안 주체가 조직의 일원인 경우 (PrincipalOrgId 조건 키 사용) 모든 작업을 허용하는 것만큼 광범위할 수도 있고, 특정 IAM 역할의 특정 작업만 허용하는 것처럼 세밀할 수도 있습니다. IAM 역할 신뢰 정책의 조건을 사용하여 비슷한 접근 방식을 취할 수 있습니다. 리소스 또는 역할 신뢰 정책에서 해당 리소스나 정책이 관리하는 역할이나 리소스와 동일한 계정의 보안 주체를 명시적으로 지정하는 경우, 해당 보안 주체에는 같은 권한을 부여하는 연결된 IAM 정책이 필요하지 않습니다. 보안 주체가 리소스와 다른 계정에 있는 경우 보안 주체에 해당 권한을 부여하는 연결된 IAM 정책이 필요합니다.

워크로드 팀은 워크로드에 필요한 권한을 관리하고자 하는 경우가 많습니다. 이를 위해서는 새 IAM 역할 및 권한 정책을 만들어야 할 수도 있습니다. 팀이 IAM 권한 경계에서 부여할 수 있는 최대 권한 범위를 캡처하고, 이 문서를 팀이 IAM 역할 및 권한을 관리하는 데 사용할 수 있는 IAM 역할에 연결할 수 있습니다. 이러한 접근 방식을 통해 IAM 관리 액세스로 인한 위험을 완화하면서 작업을 완료하는 유연성을 제공할 수 있습니다.

보다 세분화된 단계는 권한 있는 액세스 관리(PAM) 및 임시 승격 액세스 관리(TEAM) 기술을 구현하는 것입니다. 보안 주체가 권한 있는 작업을 수행하기 전에 다중 인증을 수행하도록 요구하는 것을 PAM의 예로 들 수 있습니다. 자세한 내용은 Configuring MFA-protected API access를 참조하세요. TEAM에는 보안 주체에 상위 액세스 권한을 부여할 수 있는 승인 및 기간을 관리하는 솔루션이 필요합니다. 한 가지 방법은 액세스 권한이 승격된 IAM 역할에 대한 역할 신뢰 정책에 보안 주체를 임시로 추가하

는 것입니다. 또 다른 방법은 정상적인 운영 상태에서 세션 정책을 사용하여 IAM 역할이 보안 주체에 부여한 권한의 범위를 좁힌 다음 승인된 기간에 이 제한을 일시적으로 해제하는 것입니다. AWS 및 일부 파트너가 검증한 솔루션에 대해 자세히 알아보려면 Temporary elevated access를 참조하세요.

구현 단계

1. 워크로드와 환경을 별도의 AWS 계정으로 분리합니다.
2. SCP를 사용하여 조직 구성원 계정 내에서 보안 주체에 부여할 수 있는 최대 권한 집합을 줄이세요.
   a. SCP를 정의하여 조직의 멤버 계정 내 보안 주체에게 부여할 수 있는 최대 권한 세트를 줄일 때 허용 목록 또는 거부 목록 접근 방식 중에서 선택할 수 있습니다. 허용 목록 전략은 허용되는 액세스를 명시적으로 지정하고 다른 모든 액세스를 암시적으로 차단합니다. 거부 목록 전략은 허용되지 않는 액세스를 명시적으로 지정하고 기본적으로 다른 모든 액세스를 허용합니다. 두 전략 모두 장단점이 있으며 적절한 선택은 조직의 특정 요구 사항과 위험 모델에 따라 달라집니다. 자세한 내용은 Strategy for using SCPs를 참조하세요.
   b. 또한 Service control policy examples도 검토하여 SCP를 효과적으로 구성하는 방법을 이해하세요.
3. IAM 리소스 정책을 사용하여 리소스에 허용된 작업의 범위를 좁히고 조건을 지정할 수 있습니다. IAM 역할 신뢰 정책의 조건을 사용하여 역할 수임에 대한 제한을 만드세요.
4. IAM 역할에 IAM 권한 경계를 할당하면 워크로드 팀이 자체 워크로드 IAM 역할 및 권한을 관리하는 데 사용할 수 있습니다.
5. 필요에 따라 PAM 및 TEAM 솔루션을 평가하세요.

리소스

관련 문서:

- Data perimeters on AWS
- Establish permissions guardrails using data perimeters
- 정책 평가 로직

관련 예제:

- Service control policy examples

관련 도구:

- AWS Solution: Temporary Elevated Access Management
- Validated security partner solutions for TEAM

SEC03-BP06 수명 주기에 따라 액세스 관리

조직 내 전체 수명 주기 동안 보안 주체(사용자, 역할, 그룹)에게 부여된 권한을 모니터링하고 조정합니다. 사용자의 역할이 변경됨에 따라 그룹 멤버십을 조정하고, 사용자가 조직을 떠나면 액세스 권한을 제거하세요.

원하는 성과: 조직 내 보안 주체의 수명 주기 전반에 걸쳐 권한을 모니터링하고 조정하여 불필요한 권한이 부여될 위험을 줄입니다. 사용자를 생성할 때 적절한 액세스 권한을 부여합니다. 사용자의 책임이 변경되면 액세스 권한을 수정하고, 사용자가 더 이상 활동하지 않거나 조직을 떠난 경우 액세스 권한을 제거합니다. 사용자, 역할 및 그룹에 대한 변경 사항을 중앙에서 관리합니다. 자동화를 사용하여 변경 사항을 AWS 환경에 전파합니다.

일반적인 안티 패턴:

- 처음에 필요한 것 이상으로 과도하거나 광범위한 액세스 권한을 자격 증명에 미리 부여합니다.
- 시간이 지남에 따라 자격 증명의 역할과 책임이 변경되어도 액세스 권한을 검토하고 조정하지 않습니다.
- 활동하지 않거나 퇴사한 직원 자격 증명의 활성 액세스 권한을 그대로 둡니다. 이로 인해 무단 액세스의 위험이 증가합니다.
- 자격 증명의 수명 주기를 관리하는 데 자동화를 활용하지 않습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

ID(예: 사용자, 역할, 그룹)의 수명 주기 전반에 걸쳐 부여한 액세스 권한을 신중하게 관리하고 조정하세요. 이 수명 주기에는 초기 온보딩 단계, 역할 및 책임의 지속적인 변경, 최종 오프보딩 또는 해고가 포함됩니다. 수명 주기 단계에 따라 액세스를 사전 예방적으로 관리하여 적절한 액세스 수준을 유지하세요. 최소 권한 원칙을 준수하여 과도하거나 불필요한 액세스 권한으로 인해 발생하는 위험을 줄입니다.

AWS 계정 내에서 직접 IAM 사용자의 수명 주기를 관리하거나 직원 ID 제공업체에서 AWS IAM Identity Center로의 페더레이션을 통해 관리할 수 있습니다. IAM 사용자의 경우 AWS 계정 내에서 사용자 및 관련 권한을 생성, 수정 및 삭제할 수 있습니다. 페더레이션 사용자의 경우 IAM Identity Center

를 사용하여 System for Cross-domain Identity Management(SCIM) 프로토콜을 통해 조직 ID 제공업체의 사용자 및 그룹 정보를 동기화하여 수명 주기를 관리할 수 있습니다.

SCIM은 다양한 시스템에서 사용자 ID를 자동 프로비저닝하고 프로비저닝 해제하기 위한 개방형 표준 프로토콜입니다. SCIM을 통해 ID 제공업체와 IAM Identity Center를 통합하여 사용자 및 그룹 정보를 자동으로 동기화함으로써 조직의 권한 있는 ID 소스의 변경 사항에 따라 액세스 권한이 부여, 수정 또는 취소되는지 확인할 수 있습니다.

조직 내에서 직원의 역할과 책임이 변경되면 그에 따라 액세스 권한을 조정하세요. IAM Identity Center의 권한 집합을 사용하여 다양한 직무 역할 또는 책임을 정의하고 적절한 IAM 정책 및 권한에 연결할 수 있습니다. 직원의 역할이 변경되면 새로운 담당 업무를 반영하도록 지정된 권한 집합을 업데이트할 수 있습니다. 최소 권한 원칙을 준수하면서 필요한 액세스 권한이 부여되었는지 확인하세요.

구현 단계

1. 초기 액세스 권한 부여, 정기 검토, 오프보딩 절차를 비롯한 액세스 관리 수명 주기 프로세스를 정의하고 문서화합니다.
2. IAM 역할, 그룹 및 권한 경계를 구현하여 액세스 권한을 집합적으로 관리하고 최대 허용 액세스 수준을 적용합니다.
3. IAM Identity Center를 사용하여 사용자 및 그룹 정보의 권한 있는 소스로 페더레이션 ID 제공업체(예: Microsoft Active Directory, Okta, Ping Identity)와 통합합니다.
4. SCIM 프로토콜을 사용하여 ID 제공업체의 사용자 및 그룹 정보를 IAM Identity Center의 자격 증명 스토어로 동기화합니다.
5. 조직 내의 다양한 직무 역할 또는 책임을 나타내는 권한 집합을 IAM Identity Center에서 생성합니다. 각 권한 집합에 적합한 IAM 정책 및 권한을 정의합니다.
6. 정기적인 액세스 검토, 신속한 액세스 취소, 액세스 관리 수명 주기 프로세스의 지속적인 개선을 구현합니다.
7. 직원에게 액세스 관리 모범 사례를 주제로 교육을 제공하고 인식을 제고합니다.

리소스

관련 모범 사례:

- SEC02-BP04 중앙 집중식 ID 공급업체 사용

관련 문서:

- Manage your identity source
- Manage identities in IAM Identity Center
- AWS Identity and Access Management Access Analyzer 사용
- IAM Access Analyzer policy generation

관련 비디오:

- AWS re:Inforce 2023 - Manage temporary elevated access with AWS IAM Identity Center
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2022 - Harness power of IAM policies & rein in permissions w/Access Analyzer

SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석

퍼블릭 및 크로스 계정 액세스를 강조하는 결과를 지속적으로 모니터링합니다. 이 유형의 액세스가 필요한 리소스만 허용하도록 퍼블릭 액세스 및 크로스 계정 액세스를 줄입니다.

원하는 성과: 어떤 AWS 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 구성원이 아닌 경우 개별 계정과 리소스를 공유할 수 있습니다. 리소스 기반 정책(예: Amazon Simple Storage Service(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 부여되는지 확인합니다. 공개적으로 사용 가능해야 하는 모든 리소스를 승인하는 프로세스를 정의합니다.

AWS Identity and Access Management Access Analyzer에서는 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 있습

니다. 모든 계정에 대한 가시성을 확보할 수 있도록 AWS Organizations에서 IAM Access Analyzer를 구성하는 방법을 고려하세요. 또한 IAM Access Analyzer를 사용하면 리소스 권한을 배포하기 전에 조사 결과를 미리 볼 수 있습니다. 따라서 정책 변경 사항이 리소스에 대해 의도한 퍼블릭 및 크로스 계정 액세스만 부여하는지 확인할 수 있습니다. 다중 계정 액세스를 설계할 때는 신뢰 정책을 사용하여 역할을 수임할 수 있는 사례를 제어할 수 있습니다. 예를 들어 `PrincipalOrgId` 조건 키를 사용하여 AWS Organizations 외부에서 역할을 수임하려는 시도를 거부할 수 있습니다.

AWS Config에서 잘못 구성된 리소스를 보고하고 AWS Config 정책 검사를 통해 퍼블릭 액세스가 구성된 리소스를 감지할 수 있습니다. AWS Control Tower 및 AWS Security Hub와 같은 서비스는 AWS Organizations에서 탐지 제어 및 가드레일을 배포하기만 하면 공개적으로 노출된 리소스를 파악 및 개선할 수 있습니다. 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 사용 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 규칙 규칙를 배포하여 퍼블릭 액세스가 구성된 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 수정을 사용하여 Amazon S3 버킷의 퍼블릭 액세스 구성 변경에 대응: Amazon S3 버킷에 대한 퍼블릭 액세스 차단 설정을 자동으로 활성화할 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 알림 설정을 구현해야 합니다. 또한 AWS Organizations를 사용하는 경우 Amazon S3 퍼블릭 액세스 정책의 변경을 방지하는 서비스 제어 정책을 생성할 수 있습니다. AWS Trusted Advisor는 열기 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 검사에서는 명시적인 버킷 권한뿐 아니라 버킷 권한을 재정의할 수 있는 관련 버킷 정책도 확인합니다. 또한 AWS Config를 사용하여 퍼블릭 액세스에 대해 Amazon S3 버킷을 모니터링할 수 있습니다. 자세한 내용은 How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access를 참조하세요.

Amazon S3 버킷에 대한 액세스 제어를 검토할 때 버킷 내에 저장된 데이터의 특성을 고려하는 것이 중요합니다. Amazon Macie는 개인 식별 정보(PII), 보호 대상 건강 정보(PHI), 프라이빗 키 또는 AWS

액세스 키 등의 자격 증명과 같은 민감한 데이터를 검색하고 보호하는 데 도움이 되도록 설계된 서비스입니다.

리소스

관련 문서:

- AWS Identity and Access Management Access Analyzer 사용
- AWS Control Tower controls library
- AWS Foundational Security Best Practices standard
- AWS Config 관리형 규칙
- AWS Trusted Advisor check reference
- Monitoring AWS Trusted Advisor check results with Amazon EventBridge
- Managing AWS Config Rules Across All Accounts in Your Organization
- AWS Config 및 AWS Organizations
- AMI를 Amazon EC2에서 공개적으로 사용할 수 있도록 설정

관련 비디오:

- Best Practices for securing your multi-account environment
- Dive Deep into IAM Access Analyzer

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 분리하는 구성이 있을 수 있습니다. 그러나 분리 구성을 적용한다고 해서 안전하게 공유하지 못하는 것은 아닙니다. 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 성과: 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. 외부 공유 리소스 분석에 대한 권장 가이드는 SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석을 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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, 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 기타 공통 리소스(예: VPC 서브넷 및 Transit Gateway Attachment, AWS Network Firewall 또는 Amazon SageMaker AI 파이프라인)를 공유할 수 있습니다.

조직 내에서만 리소스를 공유하도록 계정을 제한하려면 서비스 제어 정책(SCP)을 사용하여 외부 보안 주체에 대한 액세스를 방지합니다. 리소스를 공유할 때는 의도하지 않은 액세스를 방지하도록 자격 증명 기반 제어와 네트워크 제어를 결합하여 조직의 데이터 경계를 생성합니다. 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 또는 AWS 서비스 보안 주체에 적용되지 않는다는 점에 유의해야 합니다.

Amazon S3를 사용할 때 Amazon S3 버킷에 대한 ACL을 끄고 IAM 정책을 사용하여 액세스 제어를 정의합니다. Amazon CloudFront에서 Amazon S3 오리진에 대한 액세스 권한을 제한하려면 오리진 액세

스 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 서드파티 액세스 권한을 부여할 수 있습니다. 외부에서 리소스를 공유하기 위한 권한 관리에 대한 권장 가이드는 권한 관리를 참조하세요.

구현 단계

1. AWS Organizations 사용: AWS Organizations은 사용자가 생성하고 중앙에서 관리하는 단일 조직으로 여러 AWS 계정을 통합하기 위해 사용할 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 조직과 통합된 AWS 서비스의 다중 계정 관리를 사용할 수 있습니다.
2. AWS Organizations을 AWS 서비스와 통합: AWS 서비스가 조직의 구성원 계정 내에서 사용자를 대신하여 작업을 수행하는 경우 AWS Organizations은 각 구성원 계정에서 해당 서비스에 대해 IAM 서비스 연결 역할(SLR)을 생성합니다. AWS Management Console, AWS API 또는 AWS CLI를 사용하여 신뢰할 수 있는 액세스를 관리해야 합니다. 신뢰할 수 있는 액세스 활성화에 대한 권장 가이드는 Using AWS Organizations with other AWS services 및 AWS services that you can use with Organizations를 참조하세요.
3. 데이터 경계 설정: 데이터 경계는 신뢰와 소유권의 명확한 경계를 제공합니다. AWS에서는 일반적으로 AWS 리소스에 액세스하는 온프레미스 네트워크 또는 시스템과 함께 AWS Organizations에서 관리하는 AWS 조직으로 표시됩니다. 데이터 경계의 목표는 자격 증명을 신뢰할 수 있고 리소스를 신뢰할 수 있으며 네트워크가 예상되는 경우 액세스가 허용되는지 확인하는 것입니다. 그러나 데이터 경계를 설정하는 것이 모든 상황에 적합한 접근 방식은 아닙니다. 특정 보안 위험 모델 및 요구 사항에 따라 Building a Perimeter on AWS 백서에 설명된 제어 목표를 평가하고 채택합니다. 고유한 위험 태세를 신중하게 고려하고 보안 요구 사항에 맞는 경계 제어를 구현해야 합니다.
4. AWS 서비스에서 리소스 공유 사용 및 적절히 제한: Amazon Machine Image(AMI) 및 AWS Resource Access Manager(AWS RAM)와 같이 많은 AWS 서비스에서는 다른 계정과 리소스를 공유하거나 다른 계정에서 리소스를 대상으로 지정할 수 있습니다. AMI를 공유하기 위해 신뢰할 수 있는 계정을 지정하도록 `ModifyImageAttribute` API를 제한합니다. 신뢰할 수 없는 자격 증명의 액세스를 방지하도록 AWS RAM을 사용할 때 `ram:RequestedAllowsExternalPrincipals` 조건을 지정하여 사용자 조직으로만 공유를 제한합니다. 권장 가이드 및 고려 사항은 Resource sharing and external targets를 참조하세요.
5. AWS RAM을 사용하여 한 계정 내에서 또는 다른 AWS 계정과 안전하게 공유: AWS RAM은 사용자가 생성한 리소스를 사용자 계정 및 다른 AWS 계정의 역할 및 사용자와 안전하게 공유하는 데 도움이 됩니다. 다중 계정 환경에서 AWS RAM을 사용하면 리소스를 한 번 생성하여 다른 계정과 공유할

수 있습니다. 이 접근 방식은 Amazon CloudWatch 및 AWS CloudTrail과의 통합을 통해 일관성, 가시성 및 감사 가능성을 제공하는 동시에 운영 오버헤드를 줄이는 데 도움이 됩니다. 이러한 이점은 크로스 계정 액세스를 사용할 경우 얻을 수 없습니다.

이전에 리소스 기반 정책을 사용하여 공유한 리소스가 있는 경우 `PromoteResourceShareCreatedFromPolicy` API 또는 이에 상응하는 기능을 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 서드파티와 리소스 공유
- SEC05-BP01 네트워크 계층 생성

관련 문서:

- 버킷 소유자가 자신의 소유가 아닌 객체에 크로스 계정 권한 부여
- How to use Trust Policies with IAM
- Building Data Perimeter on AWS
- AWS 리소스에 대한 서드파티 액세스 권한을 부여할 때 외부 ID를 사용하는 방법
- AWS services you can use with AWS Organizations
- Establishing a data perimeter on AWS: Allow only trusted identities to access company data

관련 비디오:

- Granular Access with AWS Resource Access Manager
- Securing your data perimeter with VPC endpoints
- Establishing a data perimeter on AWS

관련 도구:

- Data Perimeter Policy Examples

SEC03-BP09 안전하게 서드파티와 리소스 공유

클라우드 환경의 보안은 조직에 국한되지 않습니다. 조직은 서드파티를 이용하여 데이터의 일부를 관리할 수 있습니다. 서드파티 관리형 시스템에 대한 권한 관리는 임시 자격 증명으로 최소 권한 원칙을 사용하여 적시 액세스 방식을 따라야 합니다. 서드파티와 긴밀히 협력하면 영향 범위와 의도치 않은 액세스의 위험을 함께 줄일 수 있습니다.

원하는 성과: 액세스 키 및 보안 키와 같은 장기 AWS Identity and Access Management(IAM) 자격 증명은 오용될 경우 보안 위험이 있으므로 사용하지 않습니다. 대신, IAM 역할과 임시 자격 증명을 사용하여 보안 태세를 개선하고 장기 자격 증명 관리의 운영 오버헤드를 최소화합니다. 서드파티 액세스 권한을 부여할 때 IAM 신뢰 정책의 외부 ID로 Universally Unique Identifier(UUID)를 사용하고 최소 권한 액세스를 보장하기 위해 IAM 정책을 제어하에 역할에 연결해 둡니다. 외부에서 공유되는 리소스에 대한 권장 가이드는 SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석을 참조하세요.

일반적인 안티 패턴:

- 조건 없이 기본 IAM 신뢰 정책을 사용합니다.
- 장기 IAM 자격 증명 및 액세스 키를 사용합니다.
- 외부 ID를 재사용합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

AWS Organizations 외부에서 리소스 공유를 허용하거나 서드파티에 계정에 대한 액세스 권한을 부여할 수 있습니다. 예를 들어, 서드파티가 계정 내 리소스에 액세스해야 하는 모니터링 솔루션을 제공할 수 있습니다. 이 경우, 서드파티에만 필요한 권한이 포함된 IAM 크로스 계정 역할을 생성합니다. 또한 외부 ID 조건을 사용하여 신뢰 정책을 정의합니다. 외부 ID를 사용하는 경우 사용자 또는 서드파티는 각 고객, 서드파티 또는 테넌시용으로 고유한 ID를 생성할 수 있습니다. 고유한 ID는 생성된 후에는 사용자 외에는 누구도 제어해서는 안 됩니다. 서드파티는 안전하고 감사 가능하며 재현 가능한 방식으로 외부 ID를 고객과 연결하는 프로세스를 구현해야 합니다.

또한 IAM Roles Anywhere를 사용하여 AWS API를 사용하는 AWS 외부 애플리케이션에 대한 IAM 역할을 관리할 수 있습니다.

서드파티가 더 이상 환경에 액세스할 필요가 없으면 역할을 제거합니다. 장기 자격 증명을 서드파티에 제공하지 않아야 합니다. 다른 AWS 계정과 워크로드를 공유할 수 있도록 허용하는 AWS Well-

Architected Tool 및 소유한 AWS 리소스를 다른 계정과 안전하게 공유할 수 있도록 지원하는 AWS Resource Access Manager 등 공유를 지원하는 다른 AWS 서비스를 지속적으로 파악하세요.

구현 단계

1. 크로스 계정 역할을 사용하여 외부 계정에 대한 액세스 권한을 제공합니다. 크로스 계정 역할을 사용하면 고객을 지원하기 위해 서드파티 및 외부 계정에서 저장하는 민감한 정보의 양을 줄일 수 있습니다. 교차 계정 역할을 사용하면 AWS 파트너 또는 조직의 다른 계정과 같은 서드파티에 계정의 AWS 리소스에 대한 액세스 권한을 안전하게 부여하는 동시에 해당 액세스를 관리 및 감사하는 기능을 유지할 수 있습니다. 서드파티는 하이브리드 인프라에서 서비스를 제공하거나 오프사이트 위치로 데이터를 가져올 수 있습니다. IAM Roles Anywhere를 사용하면 서드파티 워크로드에서 AWS 워크로드와 안전하게 상호 작용하고 추가적으로 장기 자격 증명의 필요성을 줄일 수 있습니다.

   외부 계정 액세스를 제공하기 위해 장기 자격 증명 또는 사용자와 연결된 액세스 키를 사용해서는 안 됩니다. 대신 크로스 계정 역할을 사용하여 크로스 계정 액세스를 제공합니다.

2. 실사를 수행하고 서드파티 SaaS 공급자에 대한 보안 액세스를 보장합니다. 서드파티 SaaS 공급자와 리소스를 공유할 때 철저한 실사를 수행하여 서드파티가 AWS 리소스에 액세스할 수 있는 안전하고 책임 있는 접근 방식을 갖추도록 하세요. 서드파티의 공동 책임 모델을 평가하여 서드파티가 제공하는 보안 조치와 나의 책임에 해당하는 조치를 파악합니다. SaaS 공급자가 외부 ID 및 최소 권한 액세스 원칙 사용을 포함하여 리소스에 액세스하기 위한 안전하고 감사 가능한 프로세스를 갖추고 있는지 확인합니다. 외부 ID를 사용하면 혼동된 대리자 문제를 해결하는 데 도움이 됩니다.

   보안 제어를 구현하여 서드파티 SaaS 공급자에 액세스 권한을 부여할 때 보안 액세스 및 최소 권한 원칙을 준수하도록 합니다. 여기에는 외부 ID, Universally Unique Identifiers(UUID) 및 엄격하게 필요한 것으로만 액세스를 제한하는 IAM 신뢰 정책의 사용이 포함될 수 있습니다. SaaS 공급자와 긴밀히 협력하여 안전한 액세스 메커니즘을 설정하고 AWS 리소스에 대한 SaaS 공급자의 액세스를 정기적으로 검토하며 감사를 수행하여 보안 요구 사항을 준수하는지 확인합니다.

3. 고객이 제공한 장기 자격 증명을 사용 중단합니다. 장기 자격 증명 사용을 중단하고 크로스 계정 역할 또는 IAM Roles Anywhere를 사용합니다. 장기 자격 증명을 사용해야 하는 경우 역할 기반 액세스로의 마이그레이션 계획을 수립합니다. 키 관리에 대한 자세한 내용은 Identity management를 참조하세요. 또한 AWS 계정 팀 및 서드파티와 협력하여 위험 완화 런북을 수립합니다. 보안 인시던트의 잠재적 영향에 대한 대응 및 완화에 대한 권장 가이드는 Incident response를 참조하세요.

4. 설정에 권장 가이드가 있는지 또는 자동화되어 있는지 확인합니다. 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 서드파티가 외부 ID를 생성할 수 있습니다. ID 생성 담당자를 결정하는 프로세스를 정의합니다. 외부 ID를 생성하는 엔터티에 관계없이 서드파티는 고객 간에 고유성과 형식을 일관되게 적용합니다.

사용자 계정의 크로스 계정 액세스를 위해 생성된 정책은 최소 권한 원칙을 따라야 합니다. 서드파티는 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 교차 계정 역할을 생성하는 방법에 대한 자세한 내용은 Cross-Account Roles을 참조하세요.

서드파티는 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 방식의 일환으로 드리프트 감지를 사용하여 변경 사항을 모니터링해야 합니다.

5. 변경 사항을 고려합니다. 계정 구조, 서드파티에 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 서드파티 액세스를 취소할 준비가 되어 있어야 합니다. 또한 수행하는 데 걸리는 시간, 관련된 담당자, 비용, 다른 리소스에 미치는 영향을 포함하여 취소 작업에 미치는 영향을 측정합니다.

   탐지 방법에 대한 권장 가이드는 탐지 모범 사례를 참조하세요.

리소스

관련 모범 사례:

- SEC02-BP02 임시 자격 증명 사용
- SEC03-BP05 조직에 대한 권한 가드레일 정의
- SEC03-BP06 수명 주기에 따라 액세스 관리
- SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석
- SEC04 Detection

관련 문서:

- 버킷 소유자가 자신의 소유가 아닌 객체에 크로스 계정 권한 부여
- How to use trust policies with IAM roles
- Delegate access across AWS 계정 using IAM roles
- IAM을 사용하여 다른 AWS 계정의 리소스에 액세스하려면 어떻게 해야 합니까?
- IAM의 보안 모범 사례
- Cross-account policy evaluation logic
- How to use an external ID when granting access to your AWS resources to a third party
- Collecting Information from AWS CloudFormation Resources Created in External Accounts with Custom Resources
- Securely Using External ID for Accessing AWS Accounts Owned by Others
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere

관련 비디오:

- How do I allow users or roles in a separate AWS 계정 access to my AWS 계정?
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions

관련 예제:

- Amazon DynamoDB에 대한 크로스 계정 액세스 구성
- AWS STS Network Query Tool

탐지

질문

- SEC 4. 보안 이벤트는 어떻게 감지하고 조사하나요?

SEC 4. 보안 이벤트는 어떻게 감지하고 조사하나요?

로그와 지표에서 이벤트를 캡처하고 분석하여 가시성을 확보합니다. 보안 이벤트 및 잠재적 위협에 대해 조치를 취해 워크로드를 보호할 수 있습니다.

모범 사례

- SEC04-BP01 서비스 및 애플리케이션 로깅 구성
- SEC04-BP02 표준화된 위치에서 로그, 조사 결과 및 지표 캡처
- SEC04-BP03 보안 알림 보강 및 상관관계 지정
- SEC04-BP04 규정 미준수 리소스 관련 문제 해결 시작

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 성과: 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 생성에도 로그가 필요합니다. 쿼리 및 검색 메커니즘을 선택, 활성화, 저장 및 설정하고 경보를 설정하는 것이 중요합니다.

구현 단계

- 로그 소스를 선택하고 사용합니다. 보안 조사에 앞서 관련 로그를 캡처하여 AWS 계정의 활동을 소급하여 재구성해야 합니다. 워크로드와 관련된 로그 소스를 선택합니다.

  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 직접 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 Resolver 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 Resolver 쿼리 로그는 AWS에서 보안 조사를 지원하는 기본 로깅 소스입니다. 또한 Amazon Security Lake를 사용하여 쿼리에 사용할 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 서드파티 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 않는 로그를 생성할 수 있습니다. 로깅 및 모니터링 옵션의 전체 목록은 AWS Security Incident Response Guide의 Appendix A: Cloud capability definitions – Logging and Events를 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(S3) 및 Amazon CloudWatch입니다. 보존 기간이 긴 경우 비용 효율성과 유연한 수명 주기 기능을 위해 Amazon S3를 사용하는 것이 좋습니다. 기본 로깅 옵션이 Amazon CloudWatch Logs인 경우 액세스 빈도가 낮은 로그를 Amazon S3에 아카이브하는 방법을 고려해야 합니다.

- 로그 스토리지 선택: 로그 스토리지의 선택은 일반적으로 사용하는 쿼리 도구, 보존 기능, 친숙도 및 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 로그 인사이트를 통해 내구성이 뛰어난 스토리지와 기본 제공 쿼리 기능을 제공합니다.
- 적절한 로그 보존 식별: 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 사용: 조직의 각 AWS 서비스 또는 애플리케이션에 대해 특정 로깅 구성 지침을 찾습니다.
  - AWS CloudTrail 트레일 구성
  - VPC 흐름 로그 구성
  - Amazon GuardDuty 조사 결과 내보내기 구성
  - AWS Config 레코딩 구성
  - AWS WAF 웹 ACL 트래픽 구성
  - AWS Network Firewall네트워크 트래픽 로그 구성
  - Elastic Load Balancing 액세스 로그 구성
  - Amazon Route 53 Resolver 쿼리 로그 구성
  - Amazon RDS 로그 구성
  - Amazon EKS 컨트롤 플레인 로그 구성
  - Amazon EC2 인스턴스 및 온프레미스 서버에 대해 Amazon CloudWatch 에이전트 구성
- 로그에 대한 쿼리 메커니즘 선택 및 구현: 로그 쿼리의 경우 CloudWatch 로그 그룹에 저장된 데이터에는 CloudWatch 로그 인사이트를 사용할 수 있고 Amazon S3에 저장된 데이터에는 Amazon Athena 및 Amazon OpenSearch Service를 사용할 수 있습니다. 보안 정보 및 이벤트 관리(SIEM) 서비스와 같은 서드파티 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 선

택합니다. 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 서드파티 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 Config의 경우 AWS 리소스 구성을 모니터링 및 기록하며, 원하는 구성을 기준으로 자동으로 평가하고 수정할 수 있습니다.
  - Amazon GuardDuty는 악성 활동 및 무단 행위를 지속적으로 모니터링하여 AWS 계정 계정 및 워크로드를 보호하는 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 Hub에서는 여러 AWS 서비스 및 서드파티 제품(선택 사항)의 보안 알림 또는 탐지 결과를 집계하고 정리하며 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 탐지를 구축하는 방법에 대한 자세한 내용은 AWS Security Incident Response Guide의 Detection을 참조하세요.

리소스

관련 모범 사례:

- SEC04-BP02 표준화된 위치에서 로그, 조사 결과 및 지표 캡처
- SEC07-BP04 확장 가능한 데이터 수명 주기 관리 정의
- SEC10-BP06 도구 사전 배포

관련 문서:

- AWS Security Incident Response Guide

- Amazon Security Lake 시작하기
- Getting started: Amazon CloudWatch Logs

관련 비디오:

- AWS re:Invent 2022 - Introducing Amazon Security Lake

관련 예제:

- Assisted Log Enabler for AWS
- AWS Security Hub Findings Historical Export

SEC04-BP02 표준화된 위치에서 로그, 조사 결과 및 지표 캡처

보안 팀은 로그와 조사 결과를 바탕으로 무단 활동이나 의도하지 않은 변경을 나타낼 수 있는 이벤트를 분석합니다. 이 분석을 간소화하려면 표준화된 위치에서 보안 로그와 조사 결과를 캡처하세요. 이를 통해 관심 데이터 포인트를 상관관계 분석에 사용할 수 있고 도구 통합을 간소화할 수 있습니다.

원하는 성과: 로그 데이터, 조사 결과 및 지표를 수집, 분석 및 시각화하는 표준화된 접근 방식이 있습니다. 보안 팀은 서로 다른 시스템 전반의 보안 데이터를 효율적으로 분석 및 시각화하고 상관관계를 파악하여 잠재적 보안 이벤트를 발견하고 이상 징후를 식별할 수 있습니다. 보안 정보 및 이벤트 관리 (SIEM) 시스템 또는 기타 메커니즘이 통합되어 보안 이벤트의 시기적절한 대응, 추적, 에스컬레이션을 위해 로그 데이터를 쿼리하고 분석합니다.

일반적인 안티 패턴:

- 팀이 조직의 로깅 전략과 일치하지 않는 로깅 및 지표 수집을 독립적으로 소유하고 관리합니다.
- 팀에 수집된 데이터의 가시성과 변경을 제한할 수 있는 적절한 액세스 제어 기능이 없습니다.
- 팀이 데이터 분류 정책의 일부로 보안 로그, 조사 결과 및 지표를 관리하지 않습니다.
- 팀이 데이터 수집을 구성할 때 데이터 주권 및 현지화 요구 사항을 무시합니다.

이 모범 사례 확립의 이점: 로그 데이터 및 이벤트를 수집하고 쿼리하는 표준화된 로깅 솔루션은 포함된 정보에서 도출되는 인사이트를 개선합니다. 수집된 로그 데이터의 자동화된 수명 주기를 구성하면 로그 스토리지로 인해 발생하는 비용을 줄일 수 있습니다. 팀에서 필요로 하는 데이터의 민감도와 액세스 패턴에 따라 수집된 로그 정보에 대한 세분화된 액세스 제어 기능을 구축할 수 있습니다. 도구를 통합하여 데이터의 상관관계를 파악하고, 데이터를 시각화하고, 데이터에서 인사이트를 도출할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

조직 내 AWS 사용량이 증가하면 분산된 워크로드와 환경의 수가 늘어납니다. 이러한 각 워크로드 및 환경은 내부에 활동 데이터를 생성하므로, 보안 운영 측면에서 데이터를 로컬로 캡처하고 저장하기란 어렵습니다. 보안 팀은 보안 정보 및 이벤트 관리(SIEM) 시스템과 같은 도구를 사용하여 분산된 소스에서 데이터를 수집하고 상관관계 파악, 분석 및 대응 워크플로를 거칩니다. 이를 위해서는 다양한 데이터 소스에 액세스하기 위한 복잡한 권한 집합을 관리해야 하고 추출, 전환, 적재(ETL) 프로세스를 운영하는 데 추가 오버헤드가 듭니다.

이러한 문제를 해결하려면 Organizing Your AWS Environment Using Multiple Accounts에서 설명한 대로 보안 로그 데이터의 모든 관련 소스를 로그 아카이브 계정으로 집계하는 방법을 고려하세요. 여기에는 AWS CloudTrail, AWS WAF, Elastic Load Balancing, Amazon Route 53과 같이 AWS 서비스에서 생성하는 로그 및 워크로드의 모든 보안 관련 데이터가 포함됩니다. 적절한 크로스 계정 권한이 있는 별도의 AWS 계정에서 표준화된 위치의 데이터를 캡처하면 몇 가지 이점이 있습니다. 이러한 방식은 손상된 워크로드 및 환경 내에서 로그 변조를 방지하는 데 도움이 되며, 추가 도구를 위한 단일 통합 지점을 제공하고, 데이터 보존 및 수명 주기 구성을 위한 보다 간소화된 모델을 지원합니다. 데이터 주권, 규정 준수 범위 및 기타 규정의 영향을 평가하여 여러 보안 데이터 스토리지와 보존 기간이 필요한지 결정합니다.

로그와 조사 결과를 쉽게 캡처하고 표준화하려면 로그 아카이브 계정에서 Amazon Security Lake를 평가합니다. CloudTrail, Route 53, Amazon EKS, VPC 흐름 로그와 같은 일반적인 소스에서 자동으로 데이터를 수집하도록 Security Lake를 구성할 수 있습니다. 또한 Amazon GuardDuty 및 Amazon Inspector와 같은 기타 AWS 서비스에서 조사 결과와 로그 데이터를 연관시킬 수 있도록 Security Lake의 데이터 소스로 AWS Security Hub를 구성할 수 있습니다. 서드파티 데이터 소스 통합을 사용하거나 사용자 지정 데이터 소스를 구성할 수도 있습니다. 모든 통합은 데이터를 OSCF(Open Cybersecurity Schema Framework) 형식으로 표준화하고 Amazon S3 버킷에 Parquet 파일로 저장되므로, ETL 처리가 필요하지 않습니다.

보안 데이터를 표준화된 위치에 저장하면 고급 분석 기능이 제공됩니다. AWS는 AWS 환경에서 작동하는 보안 분석 도구를 로그 아카이브 계정과 별개인 보안 도구 계정에 배포할 것을 권장합니다. 이 접근 방식을 사용하면 로그와 로그 관리 프로세스에 액세스하는 도구와는 별개로 로그 및 로그 관리 프로세스의 무결성과 가용성을 보호하기 위한 제어 기능을 심층적으로 구현할 수 있습니다. Amazon Athena와 같은 서비스를 사용하여 여러 데이터 소스를 상관시키는 온디맨드 쿼리를 실행하는 방법을 고려하세요. QuickSight와 같은 시각화 도구를 통합할 수도 있습니다. AI 기반 솔루션은 점점 더 많이 사용되고 있으며, 조사 결과를 사람이 읽을 수 있는 요약 정보와 자연어 상호 작용으로 변환하는 등의 기능을 수행할 수 있습니다. 쿼리를 위한 표준화된 데이터 스토리지 위치를 사용하면 이러한 솔루션을 보다 쉽게 통합할 수 있는 경우가 많습니다.

구현 단계

1. 로그 아카이브 및 보안 도구 계정 생성
   a. AWS Organizations를 사용하여 보안 조직 단위 아래에 로그 아카이브 및 보안 도구 계정을 생성합니다. AWS Control Tower를 사용하여 조직을 관리하는 경우 로그 아카이브 계정 및 보안 도구 계정이 자동으로 생성됩니다. 필요에 따라 이러한 계정에 액세스하고 관리하기 위한 역할 및 권한을 구성합니다.
2. 표준화된 보안 데이터 위치 구성
   a. 표준화된 보안 데이터 위치를 만들기 위한 전략을 결정합니다. 일반적인 데이터 레이크 아키텍처 접근 방식, 서드파티 데이터 제품 또는 Amazon Security Lake와 같은 옵션을 통해 이를 달성할 수 있습니다. AWS는 적극적으로 사용하지 않을 때도 계정에 대해 옵트인한 AWS 리전에서 보안 데이터를 캡처할 것을 권장합니다.
3. 표준화된 위치에 데이터 소스 게시 구성
   a. 보안 데이터의 소스를 식별하고 표준화된 위치에 게시하도록 구성합니다. ETL 프로세스를 개발해야 하는 경우와 달리 원하는 형식으로 데이터를 자동으로 내보내는 옵션을 평가합니다. Amazon Security Lake를 사용하면 지원되는 AWS 소스와 통합된 서드파티 시스템에서 데이터를 수집할 수 있습니다.
4. 표준화된 위치에 액세스할 수 있는 도구 구성
   a. 표준화된 위치에 필요한 액세스 권한을 갖도록 Amazon Athena, QuickSight 또는 서드파티 솔루션과 같은 도구를 구성합니다. 해당하는 경우 로그 아카이브 계정에 대한 크로스 계정 읽기 권한이 있는 보안 도구 계정에서 작동하도록 관련 도구를 구성합니다. Amazon Security Lake에서 구독자를 생성하여 이러한 도구에 데이터 액세스 권한을 제공합니다.

리소스

관련 모범 사례:

- SEC01-BP01 계정을 사용하여 워크로드 분리
- SEC07-BP04 확장 가능한 데이터 수명 주기 관리 정의
- SEC08-BP04 액세스 제어 적용
- OPS08-BP02 워크로드 로그 분석

관련 문서:

- AWS Whitepapers: Organizing Your AWS Environment Using Multiple Accounts

- AWS Prescriptive Guidance: AWS Security Reference Architecture (AWS SRA)
- AWS Prescriptive Guidance: Logging and monitoring guide for application owners

관련 예제:

- Aggregating, searching, and visualizing log data from distributed sources with Amazon Athena and QuickSight
- How to visualize Amazon Security Lake findings with QuickSight
- Generate AI powered insights for Amazon Security Lake using Amazon SageMaker AI Studio and Amazon Bedrock
- Identify cybersecurity anomalies in your Amazon Security Lake data using Amazon SageMaker AI
- Ingest, transform, and deliver events published by Amazon Security Lake to Amazon OpenSearch Service
- CloudTrail Lake에서 자연어 쿼리 생성을 통한 AWS CloudTrail 로그 분석 간소화

관련 도구:

- Amazon Security Lake
- Amazon Security Lake Partner 통합
- Open Cybersecurity Schema Framework (OCSF)
- Amazon Athena
- QuickSight
- Amazon Bedrock

SEC04-BP03 보안 알림 보강 및 상관관계 지정

예상치 못한 활동은 여러 소스에서 다양한 보안 알림을 생성할 수 있으므로, 전체 컨텍스트를 이해하려면 추가 상관관계 분석 및 보강이 필요합니다. 자동화된 상관관계 분석을 구현하고 보안 알림을 보강하면 인시던트를 보다 정확하게 식별하고 대응할 수 있습니다.

원하는 성과: 활동이 워크로드 및 환경 내에서 다양한 알림을 생성하면 자동화된 메커니즘이 데이터의 상관관계를 파악하고 해당 데이터를 추가 정보로 보강합니다 이러한 사전 처리를 통해 이벤트를 보다 자세히 파악할 수 있으므로, 조사관이 이벤트의 심각성과 정식 대응이 필요한 인시던트인지를 판단하는 데 도움이 됩니다. 이 프로세스를 통해 모니터링 및 조사 팀의 업무가 줄어듭니다.

일반적인 안티 패턴:

- 업무 분담 요구 사항에서 달리 규정하지 않는 한, 여러 그룹의 사람들이 서로 다른 시스템에서 생성된 결과 및 알림을 조사합니다.
- 모든 보안 조사 결과 및 알림 데이터를 표준 위치에 퍼널링하지만, 조사관이 수동으로 상관관계 분석 및 보강 작업을 수행해야 합니다.
- 조사 결과를 보고하고 중요도를 설정하는 데 위협 탐지 시스템의 인텔리전스에만 의존합니다.

이 모범 사례 확립의 이점: 경보 보강 및 자동화된 상관관계 분석은 조사자의 전반적인 인지 부담과 데이터 준비 수작업을 으로 줄이는 데 도움이 됩니다. 이렇게 하면 이벤트가 인시던트인지를 판단하고 정식 대응을 시작하는 데 걸리는 시간을 줄일 수 있습니다. 또한, 추가 맥락 정보를 통해 이벤트의 실제 심각도를 정확하게 평가할 수 있습니다. 이벤트의 심각도는 특정 알림에서 제안하는 것보다 높거나 낮을 수 있기 때문입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

보안 알림은 다음을 포함하여 AWS 내부의 다양한 소스에서 비롯될 수 있습니다.

- Amazon GuardDuty, AWS Security Hub, Amazon Macie, Amazon Inspector, AWS Config, AWS Identity and Access Management Access Analyzer, Network Access Analyzer와 같은 서비스
- Amazon OpenSearch Service의 보안 분석과 같은 AWS 서비스, 인프라 및 애플리케이션 로그의 자동화된 분석에 기반하여 알림을 생성합니다.
- Amazon CloudWatch, Amazon EventBridge 또는 AWS Budgets와 같은 소스에서 청구 활동의 변경에 대응할 때 경보를 생성합니다.
- AWS Partner Network의 보안 파트너 솔루션 및 위협 인텔리전스 피드와 같은 서드파티 소스
- AWS Trust & Safety 또는 기타 소스(예: 고객 또는 내부 직원)를 통해 문의합니다.

누가(보안 주체 또는 자격 증명) 무엇(영향을 받는 리소스)에 대해 어떤 일(취해진 조치)을 수행하고 있는지에 대한 정보를 포함하는 것이 알림의 가장 기본적인 형식입니다. 각 소스에 대해 상관관계 분석을 수행하기 위한 토대로 이러한 ID, 작업, 리소스에 대한 식별자 간의 매핑을 생성할 수 있는 방법이 있는지 확인하세요. 이는 알림 소스를 보안 정보 및 이벤트 관리(SIEM) 도구와 통합하여 자동화된 상관관계 분석을 수행하거나, 자체 데이터 파이프라인 및 처리 과정을 구축하거나, 이 둘을 조합한 형태를 취할 수 있습니다.

사용자를 대신하여 상관관계 분석을 수행할 수 있는 서비스의 예로는 Amazon Detective가 있습니다. Detective는 다양한 AWS 및 서드파티 소스의 알림을 지속적으로 수집하고 여러 형태의 인텔리전스를 통해 관계를 시각적 그래프로 구성하여 조사를 지원합니다.

알림의 초기 중요도는 우선순위를 정하는 데 도움이 되지만, 알림이 발생한 맥락에 따라 실제 중요도가 결정됩니다. 예를 들어, Amazon GuardDuty는 워크로드 내 Amazon EC2 인스턴스가 예상치 못한 도메인 이름을 쿼리하고 있다고 알릴 수 있습니다. GuardDuty는 자체적으로 이 경고에 낮은 중요도를 할당할 수 있습니다. 그러나 알림이 발생한 당시 다른 활동과의 상관관계를 자동으로 분석하면 수백 개의 EC2 인스턴스가 동일한 ID로 배포되어 전체 운영 비용이 증가할 수 있습니다. 이 이벤트에서 이 상관관계가 있는 이벤트 맥락은 새 보안 알림을 게시하고 중요도는 높음으로 조정될 수 있으며, 이 경우 추가 조치를 신속하게 처리할 수 있습니다.

구현 단계

1. 보안 알림 정보의 소스를 식별합니다. 이러한 시스템의 알림이 ID, 작업 및 리소스를 어떻게 나타내는지 이해하여 상관관계 분석이 가능한 부분을 결정합니다.
2. 다양한 소스에서 알림을 캡처하기 위한 메커니즘을 설정합니다. 이를 위해 Security Hub, EventBridge, CloudWatch와 같은 서비스를 고려하세요.
3. 데이터 상관관계 분석과 보강을 위한 소스를 식별합니다. 예시 소스에는 AWS CloudTrail, VPC 흐름 로그, Route 53 Resolver 로그, 인프라 및 애플리케이션 로그가 포함됩니다. 이러한 로그의 일부 또는 전부는 Amazon Security Lake와의 단일 통합을 통해 사용될 수 있습니다.
4. 알림을 데이터 상관관계 분석 및 보강 소스와 통합하여 보다 상세한 보안 이벤트 맥락을 생성하고 중요도를 설정합니다.
   a. Amazon Detective, SIEM 도구 또는 기타 서드파티 솔루션은 특정 수준의 수집, 상관관계 분석, 보강을 자동으로 수행할 수 있습니다.
   b. AWS 서비스를 사용하여 직접 구축할 수도 있습니다. 예를 들어, AWS Lambda 함수를 간접 호출하여 AWS CloudTrail 또는 Amazon Security Lake에 대해 Amazon Athena 쿼리를 실행하고 결과를 EventBridge에 게시할 수 있습니다.

리소스

관련 모범 사례:

- SEC10-BP03 포렌식 역량 확보
- OPS08-BP04 실행 가능한 알림 생성
- REL06-BP03 알림 전송(실시간 처리 및 경보)

관련 문서:

- AWS Security Incident Response Guide

관련 예제:

- 계정 메타데이터로 AWS Security Hub 조사 결과를 보강하는 방법

관련 도구:

- Amazon Detective
- Amazon EventBridge
- AWS Lambda
- Amazon Athena

SEC04-BP04 규정 미준수 리소스 관련 문제 해결 시작

탐지 제어를 통해 구성 요구 사항을 준수하지 않는 리소스에 대해 알림을 보낼 수 있습니다. 프로그래밍 방식으로 정의된 수정을 수동 또는 자동으로 시작하여 이러한 리소스를 수정하고 잠재적 영향을 최소화할 수 있습니다. 수정을 프로그래밍 방식으로 정의하면 신속하고 일관된 조치를 취할 수 있습니다.

자동화는 보안 운영을 개선할 수 있지만, 자동화를 신중하게 구현하고 관리해야 합니다. 적절한 감독 및 제어 메커니즘을 마련하여 자동 대응이 효과적이고 정확하며 조직의 정책과 위험을 바라보는 관점에 부합하는지 확인하세요.

원하는 성과: 리소스가 규정을 준수하지 않는 것으로 탐지될 때 이를 수정하기 위한 단계와 함께 리소스 구성 표준을 정의합니다. 가능하면 수동으로 또는 자동화를 통해 시작할 수 있도록 프로그래밍 방식으로 수정을 정의했습니다. 규정 미준수 리소스를 식별하고 보안 담당자가 모니터링하는 중앙 집중식 도구에 알림을 게시할 수 있는 탐지 시스템이 마련되어 있습니다. 이러한 도구는 수동 또는 자동으로 프로그래밍 방식의 수정 실행을 지원합니다. 자동 수정에 사용을 관리하기 위한 적절한 감독 및 제어 메커니즘이 마련되어 있습니다.

일반적인 안티 패턴:

- 자동화를 구현했지만, 수정 조치를 철저하게 테스트하고 검증하지 못했습니다. 이로 인해 정상적인 비즈니스 운영이 중단되거나 시스템이 불안정해지는 등 의도하지 않은 결과가 발생할 수 있습니다.
- 자동화를 통해 대응 시간과 절차를 개선할 수 있지만, 필요한 경우 사람이 개입하고 판단할 수 있는 적절한 모니터링 기능과 메커니즘이 없습니다.

- 보다 광범위한 인시던트 대응 및 복구 프로그램의 일부로서 수정이 아닌 일반적인 수정에만 의존합니다.

이 모범 사례 확립의 이점: 자동 수정은 수동 프로세스를 사용할 때보다 잘못된 구성에 더 빠르게 대응할 수 있으므로, 비즈니스에 미칠 수 있는 영향을 최소화하고 의도하지 않은 사용에 투입된 잠재적인 기간을 줄일 수 있습니다. 수정을 프로그래밍 방식으로 정의하면 일관되게 적용되어 인적 오류 위험이 줄어듭니다. 자동화는 또한 많은 양의 알림을 동시에 처리할 수 있는데, 이는 대규모로 운영되는 환경에서 특히 중요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

SEC01-BP03 제어 목표 파악 및 검증에서 설명한 대로, AWS Config 및 AWS Security Hub와 같은 서비스를 사용하면 요구 사항을 준수하는지 계정의 리소스 구성을 모니터링할 수 있습니다. 규정을 준수하지 않는 리소스가 감지되면 AWS Security Hub와 같은 서비스가 알림을 적절하게 라우팅하고 문제를 해결하는 데 도움이 될 수 있습니다. 이러한 솔루션은 보안 조사관이 문제를 모니터링하고 시정 조치를 취할 수 있는 중앙 장소를 제공합니다.

일부 리소스가 규정을 준수하지 않는 고유한 문제가 발생하여 수정하려면 사람의 판단이 필요한 경우도 있지만, 프로그래밍 방식으로 정의할 수 있는 표준 대응이 효과가 있는 상황도 있습니다. 예를 들어, 잘못 구성된 VPC 보안 그룹에 대한 표준 대응은 허용되지 않는 규칙을 제거하고 소유자에게 알리는 것일 수 있습니다. 응답은 AWS Lambda 함수, AWS Systems Manager Automation 문서에서 정의하거나 원하는 다른 코드 환경을 통해 정의할 수 있습니다. 수정 조치에 필요한 최소한의 권한으로 IAM 역할을 사용하여 환경이 AWS에 인증할 수 있는지 확인하세요.

원하는 수정 조치를 정의한 후에는 이를 시작하는 데 원하는 방법을 결정할 수 있습니다. AWS Config에서는 자동으로 수정을 시작할 수 있습니다. Security Hub를 사용하는 경우 사용자 지정 작업을 통해 이 작업을 수행할 수 있습니다. 그러면 조사 결과 정보를 Amazon EventBridge에 게시합니다. 그러면 EventBridge 규칙에 따라 수정이 시작될 수 있습니다. Security Hub에서 해결 방법을 자동 또는 수동으로 실행하도록 구성할 수 있습니다.

프로그래밍 방식의 수정을 위해서는 수행된 조치와 결과에 대해 포괄적인 로그와 감사를 수행하는 것이 좋습니다. 이러한 로그를 검토 및 분석하여 자동화된 프로세스의 효과를 평가하고 개선 영역을 식별합니다. Amazon CloudWatch Logs에서 로그를 캡처하고 Security Hub에서 조사 결과 노트로 결과를 캡처합니다.

시작점으로 Automated Security Response on AWS를 고려하세요. 여기에는 일반적인 보안 구성 오류를 해결하기 위한 수정 방법이 미리 구축되어 있습니다.

구현 단계

1. 알림을 분석하고 우선순위를 지정합니다.
   a. 다양한 AWS 서비스의 보안 알림을 Security Hub에 통합하여 중앙 집중식 가시성, 우선순위 지정 및 문제 해결을 제공합니다.
2. 수정 방안을 개발합니다.
   a. Systems Manager 및 AWS Lambda 등의 서비스를 사용하여 프로그래밍 방식의 수정을 실행할 수 있습니다.
3. 수정 시작 방법을 구성합니다.
   a. Systems Manager를 사용하여 조사 결과를 EventBridge에 게시할 사용자 지정 작업을 정의합니다. 이러한 작업이 수동 또는 자동으로 시작되도록 구성합니다.
   b. 또한 필요한 경우 Amazon Simple Notification Service(SNS)를 사용하여 관련 이해관계자(예: 보안 팀 또는 인시던트 대응 팀)를 대상으로 수동 개입 또는 에스컬레이션에 대한 알림 및 경보를 보낼 수도 있습니다.
4. 수정 로그를 검토 및 분석하여 효과와 개선 사항을 확인합니다.
   a. CloudWatch Logs로 로그 출력을 전송합니다. Security Hub에서 조사 결과를 결과로 캡처합니다.

리소스

관련 모범 사례:

- SEC06-BP03 수동 관리 및 대화형 액세스 감소

관련 문서:

- AWS Security Incident Response Guide - Detection

관련 예제:

- Automated Security Response on AWS
- Monitor EC2 instance key pairs using AWS Config
- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Automatically remediate unencrypted Amazon RDS DB instances and clusters

관련 도구:

- AWS Systems Manager Automation
- Automated Security Response on AWS

인프라 보호

Questions

- SEC 5. 네트워크 리소스는 어떻게 보호하나요?
- SEC 6. 컴퓨팅 리소스는 어떻게 보호하나요?

SEC 5. 네트워크 리소스는 어떻게 보호하나요?

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 네트워크 계층 내 트래픽 흐름 제어
- SEC05-BP03 검사 기반 보호 구현
- SEC05-BP04 네트워크 보호 자동화

SEC05-BP01 네트워크 계층 생성

데이터 민감도 및 액세스 요구 사항에 따라 워크로드 구성 요소의 논리적 그룹을 기반으로 네트워크 토폴로지를 여러 계층으로 세분화합니다. 인터넷에서 인바운드 액세스를 필요로 하는 구성 요소(예: 퍼블릭 웹 엔드포인트)와 내부 액세스만 필요한 구성 요소(예: 데이터베이스)를 구분합니다.

원하는 성과: 네트워크 계층은 워크로드의 자격 증명 인증 및 권한 부여 전략을 보완하는 보안에 대한 통합 심층 방어 접근 방식의 일부입니다. 데이터 민감도 및 액세스 요구 사항에 따라 적절한 트래픽 흐름 및 제어 메커니즘과 함께 계층이 배치됩니다.

일반적인 안티 패턴:

- 단일 VPC 또는 서브넷에 모든 리소스를 생성합니다.
- 데이터 민감도 요구 사항, 구성 요소 동작 또는 기능을 고려하지 않고 네트워크 계층을 구성합니다.
- 모든 네트워크 계층 고려 사항의 기본값으로 VPC와 서브넷을 사용하며, AWS 관리형 서비스가 토폴로지에 미치는 영향을 고려하지 않습니다.

이 모범 사례 확립의 이점: 네트워크 계층 구축은 네트워크를 통한 불필요한 경로, 특히 중요한 시스템 및 데이터로 이어지는 불필요한 경로를 제한하는 첫 번째 단계입니다. 이를 통해 승인되지 않은 행위자가 네트워크에 액세스할 권한을 얻어 내부의 추가 리소스를 탐색하기가 더 어려워집니다. 개별 네트워크 계층은 침입 탐지 또는 맬웨어 방지와 같은 검사 시스템의 분석 범위를 효과적으로 줄입니다. 이렇게 하면 오탐과 불필요한 처리 오버헤드가 생길 가능성이 낮아집니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드 아키텍처를 설계할 때는 보통 구성 요소를 책임에 따라 여러 계층으로 분리합니다. 예를 들어, 웹 애플리케이션에는 프레젠테이션 계층, 애플리케이션 계층, 데이터 계층이 있을 수 있습니다. 네트워크 토폴로지를 설계할 때도 비슷한 접근 방식을 사용할 수 있습니다. 기본 네트워크 제어는 워크로드의 데이터 액세스 요구 사항을 적용하는 데 도움이 될 수 있습니다. 예를 들어, 3계층 웹 애플리케이션 아키텍처에서는 정적 프레젠테이션 계층 파일을 Amazon S3에 저장하고 Amazon CloudFront 등의 콘텐츠 전송 네트워크(CDN)에서 제공할 수 있습니다. 애플리케이션 계층에는 프라이빗 서브넷에 배포된 백엔드 서비스를 통해 Application Load Balancer(ALB)가 Amazon VPC 퍼블릭 서브넷(DMZ와 유사함)에서 지원하는 퍼블릭 엔드포인트가 있을 수 있습니다. 데이터베이스, 공유 파일 시스템과 같은 리소스를 호스팅하는 데이터 계층은 애플리케이션 계층의 리소스와는 다른 프라이빗 서브넷에 있을 수 있습니다. 각 계층 경계(CDN, 퍼블릭 서브넷, 프라이빗 서브넷)에서 승인된 트래픽만 해당 경계를 통과하도록 허용하는 제어 기능을 배포할 수 있습니다.

워크로드 구성 요소의 기능적 목적을 기반으로 네트워크 계층을 모델링하는 것과 마찬가지로, 처리 중인 데이터의 민감도도 고려하세요. 웹 애플리케이션 예제를 사용하면 모든 워크로드 서비스가 애플리케이션 계층 내에 있을 수 있지만, 서비스마다 민감도 수준이 다른 데이터를 처리할 수 있습니다. 이 경우 데이터 민감도 수준별로 여러 프라이빗 서브넷, 동일한 AWS 계정의 다른 VPC 또는 다른 AWS 계정의 다른 VPC를 사용하여 애플리케이션 계층을 나누는 것이 제어 요구 사항에 따라 적합할 수 있습니다.

네트워크 계층에 대해 추가로 고려해야 할 사항은 워크로드 구성 요소의 동작 일관성입니다. 계속해서 예를 들자면, 애플리케이션 계층에는 최종 사용자의 입력을 받아들이는 서비스 또는 다른 서비스에 대한 입력보다 본질적으로 더 위험한 외부 시스템 통합이 있을 수 있습니다. 파일 업로드, 실행할 코드 스크립트, 이메일 스캔 등을 그 예로 들 수 있습니다. 이러한 서비스를 자체 네트워크 계층에 배치하면 주위에 더 강력한 격리 경계가 형성되고, 검사 시스템에서 이러한 서비스의 고유한 동작으로 인해 오탐 알림이 발생하는 것을 방지할 수 있습니다.

설계의 일부로 AWS 관리형 서비스 사용이 네트워크 토폴로지에 어떤 영향을 미치는지 고려해 보세요. Amazon VPC Lattice와 같은 서비스를 통해 네트워크 계층 전반에서 워크로드 구성 요소의 상호 운용성을 더 쉽게 지원하는 방법을 알아보세요. AWS Lambda를 사용할 때는 특별한 이유가 없다면 VPC

서브넷에 배포합니다. VPC 엔드포인트와 AWS PrivateLink가 인터넷 게이트웨이에 대한 액세스를 제한하는 보안 정책을 간편하게 준수할 수 있는 위치를 확인합니다.

구현 단계

1. 워크로드 아키텍처를 검토합니다. 제공하는 기능, 처리 중인 데이터의 민감도, 동작을 기반으로 구성 요소와 서비스를 논리적으로 그룹화하세요.
2. 인터넷 요청에 응답하는 구성 요소의 경우 로드 밸런서 또는 기타 프록시를 사용하여 퍼블릭 엔드포인트를 제공하는 것을 고려해 보세요. 퍼블릭 엔드포인트를 호스팅하기 위해 CloudFront, Amazon API Gateway, Elastic Load Balancing, AWS Amplify와 같은 관리형 서비스를 사용하여 변화하는 보안 제어를 살펴보세요.
3. Amazon EC2 인스턴스, AWS Fargate 컨테이너 또는 Lambda 함수와 같은 컴퓨팅 환경에서 실행되는 구성 요소의 경우 첫 번째 단계부터 그룹을 기반으로 이러한 구성 요소를 프라이빗 서브넷에 배포합니다.
4. Amazon DynamoDB, Amazon Kinesis 또는 Amazon SQS와 같은 완전관리형 AWS 서비스의 경우 프라이빗 IP 주소를 통한 액세스의 기본값으로 VPC 엔드포인트를 사용하는 방법을 고려하세요.

리소스

관련 모범 사례:

- REL02 네트워크 토폴로지 계획
- PERF04-BP01 네트워킹이 성능에 미치는 영향 파악

관련 비디오:

- AWS re:Invent 2023 - AWS networking foundations

관련 예제:

- VPC 예시
- AWS Fargate, AWS PrivateLink 및 Network Load Balancer를 사용하여 Amazon ECS에서 컨테이너 애플리케이션에 비공개로 액세스
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront

SEC05-BP02 네트워크 계층 내 트래픽 흐름 제어

네트워크 계층 내에서 추가 세분화를 사용하여 각 워크로드에 필요한 흐름으로만 트래픽을 제한할 수 있습니다. 먼저, 워크로드와 사용자 환경에 대한 인터넷 또는 기타 외부 시스템 간의 트래픽(남북 트래픽)을 제어하는 데 중점을 둡니다. 그런 다음 서로 다른 구성 요소와 시스템 간의 흐름(동서 트래픽)을 살펴보세요.

원하는 성과: 워크로드 구성 요소가 서로 통신하고 해당 클라이언트 및 해당 클라이언트가 의존하는 다른 서비스와 통신하는 데 필요한 네트워크 흐름만 허용합니다. 설계에서 프라이빗 송수신과 비교한 퍼블릭 송수신, 데이터 분류, 지역 규정, 프로토콜 요구 사항 등의 고려 사항을 고려합니다. 가능한 경우 최소 권한 원칙 설계의 일환으로 네트워크 피어링보다 지점 간 흐름을 선호합니다.

일반적인 안티 패턴:

- 네트워크 보안에 대한 경계 기반 접근 방식을 취하고 네트워크 계층 경계에서의 트래픽 흐름만 제어합니다.
- 네트워크 계층 내의 모든 트래픽이 인증되고 승인되었다고 가정합니다.
- 수신 트래픽과 송신 트래픽 중 하나에 제어 기능을 적용하지만, 둘 다에 적용하지는 않습니다.
- 트래픽을 인증하고 승인하는 데 워크로드 구성 요소 및 네트워크 제어에만 의존합니다.

이 모범 사례 확립의 이점: 이 방법은 네트워크 내 무단 이동의 위험을 줄이고 워크로드에 추가 인증 계층을 더하는 데 도움이 됩니다. 트래픽 흐름 제어를 수행하면 보안 인시던트의 영향 범위를 제한하고 탐지 및 대응 속도를 높일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

네트워크 계층은 유사한 기능, 데이터 민감도 수준 및 동작을 제공하는 워크로드 구성 요소 주변의 경계를 설정하는 데 도움이 됩니다. 다만 최소 권한 원칙을 따르는 이러한 계층 내 구성 요소를 추가로 분할하는 기법을 사용하여 훨씬 더 세분화된 수준의 트래픽 제어를 구성할 수 있습니다. AWS에서 네트워크 계층은 주로 Amazon VPC 내 IP 주소 범위에 따라 서브넷을 사용하여 정의됩니다. 다양한 VPC를 사용하여 계층을 정의할 수도 있습니다(예: 비즈니스 도메인별로 마이크로서비스 환경을 그룹화하는 경우). 여러 VPC를 사용하는 경우 AWS Transit Gateway를 통해 라우팅을 조정합니다. 이 경우 보안 그룹 및 라우팅 테이블을 사용하여 계층 4 수준(IP 주소 및 포트 범위)의 트래픽 제어를 제공하지만, AWS PrivateLink, Amazon Route 53 Resolver DNS Firewall, AWS Network Firewall, AWS WAF와 같은 추가 서비스를 사용하여 또 다른 제어 기능을 확보할 수 있습니다.

연결 개시 당사자, 포트, 프로토콜 및 네트워크 계층 측면에서 워크로드의 데이터 흐름 및 커뮤니케이션 요구 사항을 이해하고 인벤토리를 작성합니다. 연결 설정 및 데이터 전송에 사용할 수 있는 프로토콜을 평가하여 보호 요구 사항(예: HTTP가 아닌 HTTPS)을 충족하는 프로토콜을 선택합니다. 네트워크 경계와 각 계층 내 모두에서 이러한 요구 사항을 파악합니다. 요구 사항이 확인되면 각 연결 지점에서 필요한 트래픽만 흐르도록 허용하는 옵션을 살펴보세요. 탄력적 네트워크 인터페이스(ENI)를 사용하는 리소스(예: Amazon EC2 인스턴스, Amazon ECS 작업, Amazon EKS 포드 또는 Amazon RDS 데이터베이스)에 연결할 수 있으므로, 처음에는 VPC 내에서 보안 그룹을 사용해서 시작하는 것이 좋습니다. 계층 4 방화벽과 달리 보안 그룹에는 식별자별로 다른 보안 그룹의 트래픽을 허용하는 규칙이 있어 시간이 지남에 따라 그룹 내 리소스가 변경될 때 업데이트를 최소화할 수 있습니다. 또한, 보안 그룹을 통해 인바운드 규칙과 아웃바운드 규칙을 모두 사용하여 트래픽을 필터링할 수 있습니다.

트래픽이 VPC 간에 이동할 때 단순 라우팅에 VPC 피어링을 사용하거나 복잡한 라우팅에 AWS Transit Gateway를 사용하는 것이 일반적입니다. 이러한 접근 방식을 사용하면 소스 네트워크와 대상 네트워크의 IP 주소 범위 간 트래픽이 원활하게 흐르도록 할 수 있습니다. 하지만 워크로드에 서로 다른 VPC의 특정 구성 요소 간 트래픽 흐름만 필요한 경우에는 AWS PrivateLink를 통해 지점 간 연결을 사용하는 것이 좋습니다. 이를 위해서는 생산자 역할을 해야 하는 서비스와 소비자 역할을 해야 하는 서비스를 식별해야 합니다. 생산자에 대해 호환 가능한 로드 밸런서를 배포하고, 그에 따라 PrivateLink를 설정한 다음, 소비자의 연결 요청을 수락합니다. 그러면 생산자 서비스에 소비자 VPC의 프라이빗 IP 주소가 할당되며, 소비자는 이를 사용하여 후속 요청을 할 수 있습니다. 이 접근 방식을 사용하면 네트워크를 피어링할 필요가 줄어듭니다. PrivateLink 평가의 일부로 데이터 처리 및 로드 밸런싱에 드는 비용을 포함합니다.

보안 그룹과 PrivateLink는 워크로드의 구성 요소 간 흐름을 제어하는 데 도움이 되지만, 리소스에 액세스할 수 있는 DNS 도메인(있는 경우)을 제어하는 방법도 또 다른 주요 고려 사항입니다. VPC의 DHCP 구성에 따라 이를 위해 두 가지 다른 AWS 서비스를 고려할 수 있습니다. 대부분의 고객은 CIDR 범위의 +2 주소에 있는 VPC에서 사용할 수 있는 기본 Route 53 Resolver DNS 서비스(Amazon DNS 서버 또는 AmazonProvidedDNS라고도 함)를 사용합니다. 이 접근 방식을 사용하면 DNS 방화벽 규칙을 생성하고, 이를 VPC에 연결하여 제공한 도메인 목록에 대해 수행해야 할 작업을 결정할 수 있습니다.

Route 53 Resolver를 사용하지 않거나 도메인 필터링 외에도 심층 검사 및 흐름 제어 기능으로 Resolver를 보완하려면 AWS Network Firewall 배포를 고려해 보세요. 이 서비스는 상태 비저장 또는 상태 저장 규칙을 통해 개별 패킷을 검사하여 트래픽을 거부할지, 아니면 허용할지 결정합니다. AWS WAF를 사용하여 퍼블릭 엔드포인트에 대한 인바운드 웹 트래픽을 필터링할 때도 비슷한 접근 방식을 취할 수 있습니다. 이러한 서비스에 대한 추가 지침은 SEC05-BP03 검사 기반 보호 구현을 참조하세요.

구현 단계

1. 워크로드 구성 요소 간에 필요한 데이터 흐름을 식별합니다.

2. 보안 그룹과 라우팅 테이블 사용을 포함하여 인바운드 및 아웃바운드 트래픽 모두에 대해 심층 방어 접근 방식으로 다중 제어 기능을 적용합니다.
3. 방화벽을 사용하여 Route 53 Resolver DNS 방화벽, AWS Network Firewall, AWS WAF와 같은 VPC 내부, 외부 및 전체의 네트워크 트래픽에 대한 세분화된 제어를 정의합니다. AWS Firewall Manager를 사용하여 조직 전체의 방화벽 규칙을 중앙에서 구성하고 관리하는 방법을 고려하세요.

리소스

관련 모범 사례:

- REL03-BP01 워크로드를 세그먼트화하는 방법 선택
- SEC09-BP02 전송 중 암호화 적용

관련 문서:

- VPC에 대한 보안 모범 사례
- AWS Network Optimization Tips
- Guidance for Network Security on AWS
- Secure your VPC's outbound network traffic in the AWS 클라우드

관련 도구:

- AWS Firewall Manager

관련 비디오:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- AWS re:Inforce 2023: Firewalls and where to put them

SEC05-BP03 검사 기반 보호 구현

네트워크 계층 간에 트래픽 검사 지점을 설정하여 전송 중인 데이터가 예상 범주 및 패턴과 일치하는지 확인하세요. 트래픽 흐름, 메타데이터, 패턴을 분석하여 이벤트를 보다 효과적으로 식별, 탐지 및 대응할 수 있습니다.

원하는 성과: 네트워크 계층 사이를 이동하는 트래픽을 검사하고 승인합니다. 허용 및 거부 결정은 명시적 규칙, 위협 인텔리전스 및 기준 행동과의 편차를 기반으로 합니다. 트래픽이 민감한 데이터에 가까워질수록 보호가 더욱 엄격해집니다.

일반적인 안티 패턴:

- 포트 및 프로토콜에 기반한 방화벽 규칙에만 의존합니다. 지능형 시스템을 활용하지 않습니다.
- 변경될 수 있는 특정 최신 위협 패턴을 기반으로 방화벽 규칙을 작성합니다.
- 프라이빗 서브넷에서 퍼블릭 서브넷으로 이동하는 트래픽 또는 퍼블릭 서브넷에서 인터넷으로 전송되는 트래픽만 검사합니다.
- 네트워크 트래픽의 기본 뷰를 통해 이상 동작을 비교할 수 없습니다.

이 모범 사례 확립의 이점: 검사 시스템을 사용하면 트래픽 데이터에 특정 조건이 있는 경우에만 트래픽을 허용하거나 거부하는 등의 지능형 규칙을 작성할 수 있습니다. 시간이 흘러 위협 환경이 변화함에 따라 최신 위협 인텔리전스를 기반으로 AWS 및 파트너의 관리형 규칙 집합을 활용할 수 있습니다. 이를 통해 규칙을 유지 관리하고 보안 침해 지표를 조사하는 데 드는 오버헤드가 줄어 오탐이 발생할 가능성이 낮아집니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS Network Firewall 또는 Gateway Load Balancer(GWLB) 이면에 배포할 수 있는 기타 방화벽 및 침입 방지 시스템(IPS)을 AWS Marketplace에서 사용하여 상태 저장 및 상태 비저장 네트워크 트래픽을 세밀하게 제어할 수 있습니다. AWS Network Firewall은 워크로드를 보호하는 데 도움이 되는 Suricata 호환 오픈 소스 IPS 사양을 지원합니다.

GWLB를 사용하는 AWS Network Firewall 및 공급업체 솔루션 모두 서로 다른 인라인 검사 배포 모델을 지원합니다. 예를 들어, VPC별로 검사를 수행하거나, 검사 VPC를 중앙 집중화하거나, 검사 VPC를 통해 동서 트래픽이 흐르고 VPC별로 인터넷 수신이 검사되는 하이브리드 모델에 배포할 수 있습니다. 또 다른 고려 사항은 솔루션이 전송 계층 보안(TLS) 언래핑을 지원하여 어느 방향에서든 시작된 트래픽 흐름에 대한 심층 패킷 검사를 지원하는지 여부입니다. 이러한 구성에 대한 심층적인 세부 정보와 자세한 내용은 AWS Network Firewall Best Practice guide를 참조하세요.

무차별 모드로 작동하는 네트워크 인터페이스의 패킷 데이터에 대한 pcap 분석과 같이 대역 외 검사를 수행하는 솔루션을 사용하는 경우 VPC 트래픽 모니터링을 구성할 수 있습니다. 미러링된 트래픽은 인터페이스의 가용 대역폭에 포함되며 미러링되지 않은 트래픽과 동일한 데이터 전송 요금이 부과됩니다. 이러한 어플라이언스의 가상 버전을 AWS Marketplace에서 사용할 수 있는지 확인할 수 있으며, 이를 통해 GWLB 이면에서 인라인 배포를 지원할 수 있습니다.

HTTP 기반 프로토콜을 통해 트랜잭션하는 구성 요소의 경우 웹 애플리케이션 방화벽(WAF)을 통해 일반적인 위협으로부터 애플리케이션을 보호합니다. AWS WAF는 Amazon API Gateway, Amazon CloudFront, AWS AppSync 또는 Application Load Balancer로 전송하기 전에 구성 가능한 규칙과 일치하는 HTTP(S) 요청을 모니터링하고 차단하는 웹 애플리케이션 방화벽입니다. 일부 방화벽에서는 트래픽 검사 전에 TLS를 종료해야 하므로, 웹 애플리케이션 방화벽의 배포를 평가할 때는 심층 패킷 검사를 고려해 보세요. AWS WAF를 시작하려면 AWS Managed Rules를 자체 규칙과 함께 사용하거나 기존 파트너 통합을 사용할 수 있습니다.

AWS Firewall Manager를 사용하여 AWS 조직 전반에 걸쳐 AWS WAF, AWS Shield Advanced, AWS Network Firewall, Amazon VPC 보안 그룹을 중앙에서 관리할 수 있습니다.

구현 단계

1. 검사 VPC를 통해서처럼 검사 규칙의 범위를 광범위하게 지정할 수 있는지 또는 VPC별로 좀 더 세분화된 접근 방식이 필요한지 결정합니다.
2. 인라인 검사 솔루션의 경우:
   a. AWS Network Firewall을 사용하는 경우 규칙, 방화벽 정책 및 방화벽 자체를 생성합니다. 구성이 완료되면 트래픽을 방화벽 엔드포인트로 라우팅하여 검사를 활성화할 수 있습니다.
   b. Gateway Load Balancer(GWLB)와 함께 서드파티 어플라이언스를 사용하는 경우 하나 이상의 가용 영역에 어플라이언스를 배포하고 구성합니다. 그런 다음 GWLB, 엔드포인트 서비스, 엔드포인트를 생성하고 트래픽에 대한 라우팅을 구성합니다.
3. 대역 외 검사 솔루션의 경우:
   1. 인바운드 및 아웃바운드 트래픽을 미러링해야 하는 인터페이스에서 VPC Traffic Mirroring을 활성화합니다. Amazon EventBridge 규칙을 사용하여 새 리소스가 생성될 때 인터페이스에서 트래픽 모니터링을 활성화하는 AWS Lambda 함수를 간접 호출할 수 있습니다. Traffic Mirroring 세션이 트래픽을 처리하는 어플라이언스 앞에 있는 Network Load Balancer를 가리키도록 합니다.
4. 인바운드 웹 트래픽 솔루션의 경우:
   a. AWS WAF를 구성하려면 먼저 웹 액세스 제어 목록(웹 ACL)을 구성합니다. 웹 ACL은 순차적으로 처리된 기본 작업(ALLOW 또는 DENY)이 포함된 규칙 모음으로, WAF가 트래픽을 처리하는 방식을 정의합니다. 자체 규칙 및 그룹을 만들거나 웹 ACL에서 AWS 관리형 규칙 그룹을 사용할 수 있습니다.
   b. 웹 ACL이 구성되면 웹 ACL을 AWS 리소스(예: Application Load Balancer, API Gateway REST API 또는 CloudFront 배포)와 연결하여 웹 트래픽 보호를 시작합니다.

리소스

관련 문서:

- What is Traffic Mirroring?
- Implementing inline traffic inspection using third-party security appliances
- AWS Network Firewall example architectures with routing
- Centralized inspection architecture with AWS Gateway Load Balancer and AWS Transit Gateway

관련 예제:

- Gateway Load Balancer 배포 모범 사례
- TLS inspection configuration for encrypted egress traffic and AWS Network Firewall

관련 도구:

- AWS Marketplace IDS/IPS

SEC05-BP04 네트워크 보호 자동화

코드형 인프라(IaC) 및 CI/CD 파이프라인과 같은 DevOps 사례를 사용하여 네트워크 보호 배포를 자동화합니다. 이러한 관행은 버전 제어 시스템을 통해 네트워크 보호의 변경 사항을 추적하고, 변경 사항을 배포하는 데 걸리는 시간을 줄이며, 네트워크 보호가 원하는 구성과 다른지 탐지하는 데 도움이 될 수 있습니다.

원하는 성과: 템플릿을 사용하여 네트워크 보호를 정의하고 이를 버전 제어 시스템에 커밋합니다. 테스트 및 배포를 오케스트레이션하는 새로운 변경이 발생하면 자동화된 파이프라인이 시작됩니다. 배포 전에 변경 사항을 검증하기 위한 정책 검사 및 기타 정적 테스트가 마련되어 있습니다. 스테이징 환경에 변경 사항을 배포하여 제어 기능이 예상대로 작동하는지 확인합니다. 제어가 승인되면 프로덕션 환경으로의 배포도 자동으로 수행됩니다.

일반적인 안티 패턴:

- 개별 워크로드 팀에 의존하여 각자 전체 네트워크 스택, 보호 및 자동화를 정의합니다. 워크로드 팀이 사용할 수 있도록 네트워크 스택 및 보호의 표준 측면을 중앙 집중식으로 게시하지 않습니다.
- 중앙 네트워크 팀에 의존하여 네트워크, 보호 및 자동화의 모든 측면을 정의합니다. 네트워크 스택 및 보호의 워크로드별 측면을 해당 워크로드 팀에 맡기지 않습니다.

- 네트워크 팀과 워크로드 팀 사이에 중앙 집중화와 위임 간의 적절한 균형을 유지하고 있지만, IaC 템플릿 및 CI/CD 파이프라인 전체에서 일관된 테스트 및 배포 표준을 적용하지 않습니다. 템플릿의 준수 여부를 검사하는 도구에서 필수 구성을 캡처하지 않습니다.

이 모범 사례 확립의 이점: 템플릿을 사용하여 네트워크 보호를 정의하면 버전 제어 시스템을 통해 시간 경과에 따른 변경 사항을 추적하고 비교할 수 있습니다. 자동화를 사용하여 변경 사항을 테스트하고 배포하면 표준화와 예측 가능성을 높여 배포가 성공할 확률이 커지고 반복적인 수동 구성 작업을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

SEC05-BP02 네트워크 계층 내 트래픽 흐름 제어 및 SEC05-BP03 검사 기반 보호 구현에 설명된 여러 네트워크 보호 제어 기능에는 최신 위협 인텔리전스를 기반으로 자동 업데이트할 수 있는 관리형 규칙 시스템이 함께 제공됩니다. 웹 엔드포인트 보호의 예로는 AWS WAF 관리형 규칙 및 AWS Shield Advanced Shield Advanced 자동 애플리케이션 계층 DDoS 완화가 있습니다. AWS Network Firewall 관리형 규칙 그룹을 사용하여 평판이 낮은 도메인 목록과 위협 서명도 최신 상태로 유지합니다.

관리형 규칙 외에도 DevOps 관행을 적용하여 네트워크 리소스, 보호 및 지정한 규칙의 배포를 자동화하는 것이 좋습니다. 이러한 정의를 AWS CloudFormation 또는 다른 원하는 코드형 인프라(IaC) 도구로 캡처하고 버전 제어 시스템에 커밋하며 CI/CD 파이프라인을 사용하여 배포할 수 있습니다. 이 접근 방식을 사용하면 네트워크 제어 관리 시 DevOps의 전통적인 이점을 누릴 수 있습니다. 예를 들어 릴리스 예측 가능성을 높이고, AWS CloudFormation Guard와 같은 도구를 사용하여 테스트를 자동화하며, 배포된 환경과 원하는 구성 간의 차이를 탐지할 수 있습니다.

SEC05-BP01 네트워크 계층 생성의 일부로 내린 결정에 따라 중앙 관리 접근 방식을 통해 수신, 송신 및 검사 흐름 전용 VPC를 생성할 수 있습니다. AWS Security Reference Architecture(AWS SRA)에서 설명한 대로, 전용 네트워크 인프라 계정에서 이러한 VPC를 정의할 수 있습니다. 유사한 기술을 사용하여 다른 계정의 워크로드, 보안 그룹, AWS Network Firewall 배포, Route 53 Resolver 규칙, DNS 방화벽 구성, 기타 네트워크 리소스에서 사용하는 VPC를 중앙에서 정의할 수 있습니다. AWS Resource Access Manager을 통해 다른 계정과 이러한 리소스를 공유할 수 있습니다. 이 접근 방식을 사용하면 관리할 대상이 하나뿐이므로, 네트워크 제어 기능은 네트워크 계정에 자동으로 테스트하고 배포하는 작업을 간소화할 수 있습니다. 하이브리드 모델에서 이 작업을 수행할 수 있습니다. 하이브리드 모델에서는 특정 제어 기능을 중앙에서 배포 및 공유하고 다른 제어 기능은 개별 워크로드 팀과 해당 계정에 위임할 수 있습니다.

구현 단계

1. 네트워크와 보호의 어떤 부분을 중앙에서 정의하고 워크로드 팀은 어떤 부분을 유지 관리할지에 대한 소유권을 설정합니다.
2. 네트워크 및 보호에 대한 변경 사항을 테스트하고 배포할 수 있는 환경을 만듭니다. 예를 들어, 네트워크 테스트 계정과 네트워크 프로덕션 계정을 사용하세요.
3. 버전 관리 시스템에서 템플릿을 저장하고 유지 관리하는 방법을 결정합니다. 중앙 템플릿은 워크로드 리포지토리와는 다른 리포지토리에 저장하고, 워크로드 템플릿은 해당 워크로드와 관련된 리포지토리에 저장할 수 있습니다.
4. CI/CD 파이프라인을 생성하여 템플릿을 테스트하고 배포합니다. 잘못된 구성이 있는지 점검하고 템플릿이 기업 표준을 준수하는지 확인하는 테스트를 정의합니다.

리소스

관련 모범 사례:

- SEC01-BP06 표준 보안 제어의 배포 자동화

관련 문서:

- AWS Security Reference Architecture - Network account

관련 예제:

- AWS Deployment Pipeline Reference Architecture
- NetDevSecOps to modernize AWS networking deployments
- Integrating AWS CloudFormation security tests with AWS Security Hub and AWS CodeBuild reports

관련 도구:

- AWS CloudFormation
- AWS CloudFormation Guard
- cfn_nag

SEC 6. 컴퓨팅 리소스는 어떻게 보호하나요?

워크로드의 컴퓨팅 리소스를 외부 및 내부 위협으로부터 보호할 수 있는 다중 방어 계층이 필요합니다. 컴퓨팅 리소스에는 EC2 인스턴스, 컨테이너, AWS Lambda 함수, 데이터베이스 서비스, IoT 디바이스 등이 포함됩니다.

모범 사례

- SEC06-BP01 취약성 관리 수행
- SEC06-BP02 강화된 이미지로부터 컴퓨팅 프로비저닝
- SEC06-BP03 수동 관리 및 대화형 액세스 감소
- SEC06-BP04 소프트웨어 무결성 검증
- SEC06-BP05 컴퓨팅 보호 자동화

SEC06-BP01 취약성 관리 수행

코드, 종속성 및 인프라에 취약성이 있는지 자주 스캔하고 패치를 적용하여 새로운 위협으로부터 보호합니다.

원하는 성과: 워크로드에서 소프트웨어 취약성, 잠재적 결함 및 의도하지 않은 네트워크 노출을 지속적으로 검사하는 솔루션이 있습니다. 위험 평가 기준에 따라 이러한 취약성을 식별하고, 우선순위를 지정하고, 해결하는 프로세스와 절차를 수립했습니다. 또한 컴퓨팅 인스턴스에 자동 패치 관리를 구현했습니다. 취약성 관리 프로그램은 CI/CD 파이프라인 중에 소스 코드를 스캔하는 솔루션과 함께 소프트웨어 개발 수명 주기에 통합됩니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

취약성 관리는 안전하고 견고한 클라우드 환경을 유지하는 데 있어 중요한 요소입니다. 여기에는 보안 스캔, 문제의 식별 및 우선순위 지정, 식별된 취약성을 해결하기 위한 패치 작업을 포함하는 포괄적인

프로세스가 포함됩니다. 자동화는 잠재적인 문제 및 의도하지 않은 네트워크 노출과 문제 해결 노력이 있는지 워크로드를 지속적으로 스캔할 수 있도록 하기 때문에 이 프로세스에서 중추적 역할을 합니다.

AWS 공동 책임 모델은 취약성 관리를 뒷받침하는 기본 개념입니다. 이 모델에 따르면 AWS는 AWS 서비스를 실행하는 하드웨어, 소프트웨어, 네트워킹 및 시설을 포함한 기본 인프라를 보호할 책임이 있습니다. 반대로 사용자는 서비스와 관련된 Amazon EC2 인스턴스 및 Amazon S3 객체 등 사용자의 데이터, 보안 구성 및 관리 작업을 보호할 책임이 있습니다.

AWS는 취약성 관리 프로그램을 지원하는 다양한 서비스를 제공합니다. Amazon Inspector는 지속적으로 AWS 워크로드에 소프트웨어 취약성과 의도하지 않은 네트워크 액세스가 있는지 스캔하는 한편, AWS Systems Manager Patch Manager는 Amazon EC2 인스턴스 전반의 패치 관리를 지원합니다. 이러한 서비스는 AWS 보안 검사를 자동화하고, 보안 알림을 중앙 집중화하고, 조직의 보안 태세에 대한 포괄적인 보기를 제공하는 클라우드 보안 태세 관리 서비스인 AWS Security Hub와 통합할 수 있습니다. 또한 Amazon CodeGuru Security는 정적 코드 분석을 사용하여 개발 단계에서 Java 및 Python 애플리케이션의 잠재적 문제를 식별합니다.

취약성 관리 사례를 소프트웨어 개발 수명 주기에 통합하면 취약성이 프로덕션 환경에 도입되기 전에 사전 예방적으로 해결할 수 있으므로 보안 이벤트의 위험을 줄이고 취약성의 잠재적 영향을 최소화할 수 있습니다.

구현 단계

1. 공동 책임 모델 이해: AWS 공동 책임 모델을 검토하여 클라우드에서 워크로드와 데이터를 보호하는 책임을 이해합니다. AWS는 기본 클라우드 인프라를 보호하는 역할을 담당하고, 사용자는 애플리케이션, 데이터 및 사용하는 서비스를 보호할 책임이 있습니다.
2. 취약성 스캔 구현: Amazon Inspector와 같은 취약성 스캔 서비스를 구성하여 컴퓨팅 인스턴스(예: 가상 머신, 컨테이너 또는 서버리스 함수)에 소프트웨어 취약성, 잠재적 결함 및 의도하지 않은 네트워크 노출이 있는지 자동으로 검사합니다.
3. 취약성 관리 프로세스 수립: 취약성을 식별하고, 우선순위를 지정하고, 해결하기 위한 프로세스 및 절차를 정의합니다. 여기에는 정기적인 취약성 검사 일정 설정, 위험 평가 기준 설정, 취약성 심각도에 따른 개선 일정 정의가 포함될 수 있습니다.
4. 패치 관리 설정: 패치 관리 서비스를 사용하여 운영 체제 및 애플리케이션에 대한 컴퓨팅 인스턴스 패치 프로세스를 자동화합니다. 누락된 패치가 있는지 인스턴스를 스캔하고 일정에 따라 자동으로 설치하도록 서비스를 구성할 수 있습니다. 이 기능을 제공하려면 AWS Systems Manager Patch Manager를 고려해 보세요.
5. 맬웨어 방지 구성: 환경에서 악성 소프트웨어를 감지하는 메커니즘을 구현합니다. 예를 들어 Amazon GuardDuty와 같은 도구를 사용하여 EC2 및 EBS 볼륨에서 맬웨어를 분석 및 탐지하고 맬

웨어에 관해 알림을 보낼 수 있습니다. 또한 GuardDuty는 Amazon S3에 새로 업로드된 객체를 스캔하여 잠재적 맬웨어 또는 바이러스가 있는지 확인하고 다운스트림 프로세스에 수집하기 전에 격리 조치를 취할 수 있습니다.

6. CI/CD 파이프라인에 취약성 스캔 통합: 애플리케이션 배포에 CI/CD 파이프라인을 사용하는 경우 취약성 스캔 도구를 파이프라인에 통합합니다. Amazon CodeGuru Security 및 오픈 소스 옵션과 같은 도구는 소스 코드, 종속성 및 아티팩트에서 잠재적 보안 문제를 스캔할 수 있습니다.
7. 보안 모니터링 서비스 구성: AWS Security Hub와 같은 보안 모니터링 서비스를 설정하여 여러 클라우드 서비스에서 보안 태세를 포괄적으로 확인할 수 있습니다. 이 서비스는 다양한 소스에서 보안 조사 결과를 수집하여 표준화된 형식으로 제시함으로써 우선순위 지정 및 수정을 용이하게 해야 합니다.
8. 웹 애플리케이션 침투 테스트 구현: 애플리케이션이 웹 애플리케이션이고 조직이 필요한 스킬을 갖추고 있거나 외부 지원할 사람을 고용할 수 있는 경우, 애플리케이션의 잠재적 취약성을 식별하기 위해 웹 애플리케이션 침투 테스트를 구현하는 것이 좋습니다.
9. 코드형 인프라로 자동화: AWS CloudFormation과 같은 코드형 인프라(IaC) 도구를 사용하여 앞서 언급한 보안 서비스를 포함하여 리소스의 배포 및 구성을 자동화합니다. 이 방법을 사용하면 여러 계정 및 환경에서 보다 일관되고 표준화된 리소스 아키텍처를 생성할 수 있습니다.
10. 모니터링 및 지속적인 개선: 취약성 관리 프로그램의 효과를 지속적으로 모니터링하고 필요에 따라 개선합니다. 보안 조사 결과를 검토하고, 개선 노력의 효과를 평가하고, 그에 따라 프로세스와 도구를 조정합니다.

리소스

관련 문서:

- AWS Systems Manager
- Security Overview of AWS Lambda
- Amazon CodeGuru
- Improved, Automated Vulnerability Management for Cloud Workloads with a New Amazon Inspector
- Automate vulnerability management and remediation in AWS using Amazon Inspector and AWS Systems Manager – Part 1

관련 비디오:

- Securing Serverless and Container Services

- Security best practices for the Amazon EC2 instance metadata service

SEC06-BP02 강화된 이미지로부터 컴퓨팅 프로비저닝

강화된 이미지에서 배포하여 런타임 환경에 의도치 않게 액세스하는 상황을 줄이세요. 신뢰할 수 있는 레지스트리에서 컨테이너 이미지 및 애플리케이션 라이브러리와 같은 런타임 종속성만 획득하고 해당 서명을 확인합니다. 자체 프라이빗 레지스트리를 생성하여 빌드 및 배포 프로세스에 사용할 신뢰할 수 있는 이미지와 라이브러리를 저장하세요.

원하는 성과: 컴퓨팅 리소스가 강화된 기준 이미지에서 프로비저닝됩니다. 신뢰할 수 있는 레지스트리에서만 컨테이너 이미지 및 애플리케이션 라이브러리와 같은 외부 종속성을 검색하고 해당 서명을 확인합니다. 이러한 정보는 빌드 및 배포 프로세스에서 참조할 수 있도록 프라이빗 레지스트리에 저장됩니다. 이미지와 종속성을 정기적으로 스캔하고 업데이트하여 새로 발견된 취약성으로부터 보호합니다.

일반적인 안티 패턴:

- 신뢰할 수 있는 레지스트리에서 이미지와 라이브러리를 가져오지만, 사용하기 전에 서명을 확인하거나 취약성 스캔을 수행하지는 않습니다.
- 이미지를 강화하지만, 정기적으로 새로운 취약성을 테스트하거나 최신 버전으로 업데이트하지는 않습니다.
- 이미지의 예상 수명 주기 동안 필요하지 않은 소프트웨어 패키지를 설치하거나 제거하지 않습니다.
- 프로덕션 컴퓨팅 리소스를 최신 상태로 유지하는 데 패치 작업에만 의존합니다. 패치만 적용하면 시간이 지나면서 컴퓨팅 리소스가 강화된 표준에서 벗어날 수 있습니다. 또한, 패치를 적용해도 보안 이벤트 중에 위협 행위자가 설치했을 수 있는 맬웨어를 제거하지 못할 수 있습니다.

이 모범 사례 확립의 이점: 이미지를 강화하면 런타임 환경에서 승인되지 않은 사용자나 서비스에 의도하지 않은 액세스를 허용할 수 있는 경로의 수를 줄일 수 있습니다. 또한, 의도하지 않은 액세스가 발생할 경우 영향을 받는 범위를 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

시스템을 강화하려면 최신 버전의 운영 체제, 컨테이너 이미지, 애플리케이션 라이브러리부터 시작하세요. 알려진 문제에 패치를 적용합니다. 불필요한 애플리케이션, 서비스, 디바이스 드라이버, 기본 사용자 및 기타 자격 증명을 제거하여 시스템을 최소화합니다. 워크로드에 필요한 리소스와 기능만 갖춘

환경을 만들기 위해 포트를 비활성화하는 등 필요한 기타 조치를 수행합니다. 이 기준에서 워크로드 모니터링 또는 취약성 관리와 같은 목적에 필요한 소프트웨어, 에이전트 또는 기타 프로세스를 설치할 수 있습니다.

CIS(Center for Internet Security) 및 DISA(Defense Information Systems Agency) STIG(Security Technical Implementation Guide) 등 신뢰할 수 있는 소스에서 제공하는 지침을 활용하여 시스템 강화에 대한 부담을 덜 수 있습니다. 먼저 AWS 또는 APN 파트너가 게시한 Amazon Machine Image(AMI)로 시작하여 CIS 및 STIG 제어의 적절한 조합에 따라 AWS EC2 Image Builder를 사용하여 구성을 자동화하는 것이 좋습니다.

CIS 또는 DISA STIG 권장 사항을 적용하는 강화된 이미지와 EC2 Image Builder 레시피가 제공되지만, 이러한 구성으로 인해 소프트웨어가 제대로 실행되지 않을 수 있습니다. 이 경우 강화되지 않은 기본 이미지에서 시작하여 소프트웨어를 설치한 다음, CIS 제어 기능을 점진적으로 적용하여 영향을 테스트하면 됩니다. 소프트웨어 실행을 방해하는 CIS 제어 기능의 경우, 대신 DISA에서 세분화된 강화 권장 사항을 구현할 수 있는지 테스트하세요. 성공적으로 적용할 수 있는 다양한 CIS 제어 기능 및 DISA STIG 구성을 추적합니다. 이를 사용하여 EC2 Image Builder의 이미지 강화 레시피를 적절하게 정의합니다.

컨테이너식 워크로드의 경우 Docker의 강화된 이미지는 Amazon Elastic Container Registry(ECR) 퍼블릭 리포지토리에서 사용할 수 있습니다. EC2 Image Builder를 사용하여 AMI와 함께 컨테이너 이미지를 강화할 수 있습니다.

운영 체제 및 컨테이너 이미지와 마찬가지로 pip, npm, Maven 및 NuGet과 같은 도구를 통해 퍼블릭 리포지토리에서 코드 패키지(또는 라이브러리)를 가져올 수 있습니다. AWS CodeArtifact 내부와 같은 프라이빗 리포지토리를 신뢰할 수 있는 퍼블릭 리포지토리와 통합하여 코드 패키지를 관리하는 것이 좋습니다. 이 통합을 통해 패키지를 검색 및 저장하고 최신 상태로 유지할 수 있습니다. 그러면 애플리케이션 빌드 프로세스에서 소프트웨어 구성 분석(SCA), 정적 애플리케이션 보안 테스트(SAST), 동적 애플리케이션 보안 테스트(DAST)와 같은 기술을 사용하여 애플리케이션과 함께 이러한 패키지의 최신 버전을 구해 테스트할 수 있습니다.

AWS Lambda를 사용하는 서버리스 워크로드의 경우 Lambda 계층을 사용하여 패키지 종속성 관리를 간소화합니다. Lambda 계층을 사용하여 여러 기능에서 공유되는 표준 종속성 집합을 독립형 아카이브로 구성합니다. 자체 빌드 프로세스를 통해 함수를 중앙에서 최신 상태로 유지하는 방법을 제공하며 계층을 생성하고 유지 관리할 수 있습니다.

구현 단계

- 운영 체제를 강화합니다. 신뢰할 수 있는 소스의 기본 이미지를 기반으로 강화된 AMI를 구축합니다. EC2 Image Builder를 사용하면 이미지에 설치된 소프트웨어를 사용자 지정하는 데 도움이 됩니다.

- 컨테이너식 리소스를 강화합니다. 보안 모범 사례에 맞춰 컨테이너식 리소스를 구성합니다. 컨테이너를 사용할 때는 빌드 파이프라인에서 이미지 리포지토리에 대해 정기적으로 ECR Image Scanning을 구현하여 컨테이너에서 CVE를 찾습니다.
- AWS Lambda를 통해 서버리스 구현을 사용하는 경우 Lambda 계층을 사용하여 애플리케이션 함수 코드와 공유 종속 라이브러리를 분리합니다. 신뢰할 수 있는 코드만 Lambda 함수에서 실행되도록 Lambda에 대한 코드 서명을 구성합니다.

리소스

관련 모범 사례:

- OPS05-BP05 패치 관리 수행

관련 비디오:

- Deep dive into AWS Lambda security

관련 예제:

- Quickly build STIG-compliant AMI using EC2 Image Builder
- Building better container images
- Using Lambda layers to simplify your development process
- Develop & Deploy AWS Lambda Layers using Serverless Framework
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools

SEC06-BP03 수동 관리 및 대화형 액세스 감소

자동화를 사용하여 가능한 모든 곳에서 배포, 구성, 유지 관리 및 조사 작업을 수행합니다. 긴급 절차나 안전한(샌드박스) 환경에서 자동화를 사용할 수 없는 경우에는 컴퓨팅 리소스에 수동으로 액세스하는 것이 좋습니다.

원하는 성과: 프로그래밍 스크립트와 자동화 문서(런북)가 컴퓨팅 리소스에서 승인된 작업을 캡처합니다. 이러한 런북은 변경 탐지 시스템을 통해 자동으로 시작될 수도, 사람의 판단이 필요할 때는 수동으로 시작될 수도 있습니다. 컴퓨팅 리소스에 대한 직접 액세스는 자동화를 사용할 수 없는 긴급 상황에서만 지원됩니다. 모든 수동 활동이 로깅되고 검토 프로세스에 통합되어 자동화 기능을 지속적으로 개선합니다.

일반적인 안티 패턴:

- SSH 또는 RDP와 같은 프로토콜을 사용하여 Amazon EC2 인스턴스에 대화식으로 액세스합니다.
- `/etc/passwd` 또는 Windows 로컬 사용자와 같은 개별 사용자 로그인을 유지 관리합니다.
- 여러 사용자 간에 인스턴스에 액세스하기 위한 암호 또는 프라이빗 키를 공유합니다.
- 수동으로 소프트웨어를 설치하고 구성 파일을 만들거나 업데이트합니다.
- 수동으로 소프트웨어를 업데이트하거나 패치를 적용합니다.
- 문제 해결을 위해 인스턴스에 로그인합니다.

이 모범 사례 확립의 이점: 자동화를 통해 작업을 수행하면 의도하지 않은 변경 및 잘못된 구성으로 인한 운영 위험을 줄일 수 있습니다. 대화형 액세스에 Secure Shell(SSH) 및 Remote Desktop Protocol(RDP) 사용을 제거하면 컴퓨팅 리소스에 대한 액세스 범위가 줄어듭니다. 이렇게 하면 무단 행위가 발생하는 일반적인 경로가 사라집니다. 자동화 문서 및 프로그래밍 스크립트에 컴퓨팅 리소스 관리 작업을 캡처하면 승인된 활동의 전체 범위를 세밀하게 정의하고 감사할 수 있는 메커니즘이 제공됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

인스턴스에 로그인하는 것은 전형적인 시스템 관리 접근 방식입니다. 사용자는 일반적으로 서버 운영 체제를 설치한 후 수동으로 로그인하여 시스템을 구성하고 원하는 소프트웨어를 설치합니다. 서버 수명 기간에 사용자는 로그인하여 소프트웨어 업데이트를 수행하고, 패치를 적용하며, 구성을 변경하고, 문제를 해결할 수 있습니다.

그러나 수동으로 액세스하면 여러 위험이 따릅니다. 무단 액세스에 대한 잠재적 경로를 제공할 수 있는 SSH 또는 RDP 서비스와 같은 요청을 수신하는 서버를 필요로 합니다. 또한, 수동 단계 수행과 관련되어 인적 오류가 발생할 위험도 증가합니다. 이로 인해 워크로드 인시던트, 데이터 손상 또는 폐기를 비롯하여 기타 보안 문제가 발생할 수 있습니다. 나아가 사람이 액세스하려면 자격 증명 공유에 대한 보호가 필요하므로, 관리 오버헤드가 가중됩니다.

이러한 위험을 완화하기 위해 AWS Systems Manager와 같은 에이전트 기반 원격 액세스 솔루션을 구현할 수 있습니다. AWS Systems Manager 에이전트(SSM 에이전트)는 암호화된 채널을 시작하므로, 외부에서 시작된 요청을 수신 대기하는 데 의존하지 않습니다. VPC 엔드포인트를 통해 이 채널을 설정하도록 SSM 에이전트를 구성하는 방법을 고려하세요.

Systems Manager를 사용하면 관리형 인스턴스와 상호 작용하는 방법을 세밀하게 제어할 수 있습니다. 실행할 자동화, 실행할 수 있는 사용자, 실행할 수 있는 시기를 정의합니다. Systems Manager는 인

스턴스에 대한 대화형 액세스 없이 패치를 적용하고, 소프트웨어를 설치하며, 구성을 변경할 수 있습니다. 또한 Systems Manager는 원격 쉘에 액세스하여 세션 중에 간접 호출된 모든 명령과 해당 출력을 로그 및 Amazon S3에 로깅할 수 있습니다. AWS CloudTrail은 검사 목적으로 Systems Manager API 간접 호출을 기록합니다.

구현 단계

1. Amazon EC2 인스턴스에 AWS Systems Manager 에이전트(SSM 에이전트)를 설치합니다. SSM Agent가 기본 AMI 구성의 일부로 포함되고 자동으로 시작되는지 확인합니다.
2. EC2 인스턴스 프로파일과 연결된 IAM 역할에 `AmazonSSMManagedInstanceCore` 관리형 IAM 정책이 포함되어 있는지 확인합니다.
3. 인스턴스에서 실행 중인 SSH, RDP 및 기타 원격 액세스 서비스를 비활성화합니다. 시작 템플릿의 사용자 데이터 섹션에 구성된 스크립트를 실행하거나 EC2 Image Builder와 같은 도구를 사용하여 사용자 지정 AMI를 구축하면 됩니다.
4. EC2 인스턴스에 적용되는 보안 그룹 수신 규칙이 포트 22/tcp(SSH) 또는 포트 3389/tcp(RDP)에서의 액세스를 허용하지 않는지 확인합니다. AWS Config 등의 서비스를 사용하여 잘못 구성된 보안 그룹에 대한 탐지 및 알림을 구현합니다.
5. Systems Manager에서 적절한 자동화, 런북을 정의하고 명령을 실행합니다. IAM 정책을 사용하여 이러한 작업을 수행할 수 있는 사람과 작업이 허용되는 조건을 정의합니다. 프로덕션 환경이 아닌 환경에서 자동화를 철저하게 테스트하세요. 필요한 경우 대화형 방식으로 인스턴스에 액세스하지 않고 자동화를 간접 호출할 수 있습니다.
6. 필요한 경우 AWS Systems Manager Session Manager를 사용하여 인스턴스에 대한 대화형 액세스를 제공합니다. 세션 활동 로깅을 활성화하여 Amazon CloudWatch Logs 또는 Amazon S3에서 감사 기록을 유지 관리합니다.

리소스

관련 모범 사례:

- REL08-BP04 변경 불가능한 인프라를 사용하여 배포

관련 예제:

- Replacing SSH access to reduce management and security overhead with AWS Systems Manager

관련 도구:

- AWS Systems Manager

관련 비디오:

- Controlling User Session Access to Instances in AWS Systems Manager Session Manager

SEC06-BP04 소프트웨어 무결성 검증

암호화 검증을 사용하여 워크로드에서 사용하는 소프트웨어 아티팩트(이미지 포함)의 무결성을 검증합니다. 컴퓨팅 환경 내에서 실행되는 무단 변경을 방지하기 위해 소프트웨어를 암호화 방식으로 서명합니다.

원하는 성과: 모든 아티팩트를 신뢰할 수 있는 소스에서 가져옵니다. 공급업체 웹 사이트 인증서가 검증되었습니다. 다운로드한 아티팩트는 서명을 통해 암호화 방식으로 확인됩니다. 자체 소프트웨어는 컴퓨팅 환경에서 암호화 방식으로 서명되고 확인됩니다.

일반적인 안티 패턴:

- 믿을 수 있는 공급업체 웹 사이트를 신뢰하여 소프트웨어 아티팩트를 가져오지만, 인증서 만료 통지는 무시합니다. 인증서가 유효한지 확인하지 않고 다운로드를 진행합니다.
- 공급업체 웹 사이트 인증서를 검증하지만, 해당 웹 사이트에서 다운로드한 아티팩트를 암호화 방식으로 확인하지는 않습니다.
- 요약 또는 해시에만 의존하여 소프트웨어 무결성을 검증합니다. 해시는 아티팩트가 원본 버전에서 수정되지 않았음을 확인하지만, 소스를 검증하지는 않습니다.
- 자체 배포에서만 사용하는 경우에도 자체 소프트웨어, 코드 또는 라이브러리에 서명하지 않습니다.

이 모범 사례 확립의 이점: 워크로드가 의존하는 아티팩트의 무결성을 검증하면 맬웨어가 컴퓨팅 환경에 침입하는 것을 방지할 수 있습니다. 소프트웨어에 서명하면 컴퓨팅 환경에서 무단 실행되지 않도록 보호하는 데 도움이 됩니다. 코드 서명 및 확인을 통해 소프트웨어 공급망을 보호하세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

운영 체제 이미지, 컨테이너 이미지 및 코드 아티팩트는 요약이나 해시와 같은 무결성 검사가 가능한 상태로 배포되는 경우가 많습니다. 이를 통해 클라이언트는 페이로드의 자체 해시를 계산하고 게시된 것과 동일한지 검증하여 무결성을 확인할 수 있습니다. 이러한 검사는 페이로드가 변조되지 않았음을

확인하는 데 도움이 되지만, 페이로드가 원본 소스(출처)에서 왔는지 검증하지는 않습니다. 출처를 확인하려면 신뢰할 수 있는 기관에서 아티팩트에 디지털 서명하기 위해 발급한 인증서가 필요합니다.

워크로드에서 다운로드한 소프트웨어 또는 아티팩트를 사용하는 경우 제공업체가 디지털 서명 확인을 위한 퍼블릭 키를 제공하는지 확인하세요. AWS에서 당사가 게시하는 소프트웨어에 대한 퍼블릭 키 및 확인 지침을 제공하는 방법의 몇 가지 예는 다음과 같습니다.

- EC2 Image Builder: Verify the signature of the AWSTOE installation download
- AWS Systems Manager: SSM 에이전트의 서명 확인
- Amazon CloudWatch: Verifying the signature of the CloudWatch agent package

SEC06-BP02 강화된 이미지로부터 컴퓨팅 프로비저닝에서 설명한 대로 이미지 획득 및 강화에 사용하는 프로세스에 디지털 서명 검증을 통합합니다.

AWS Signer를 사용하여 서명 검증은 물론, 자체 소프트웨어 및 아티팩트에 대한 자체 코드 서명 수명 주기를 관리하는 데 도움이 될 수 있습니다. AWS Lambda 및 Amazon Elastic Container Registry 모두 코드 및 이미지의 서명 확인을 위해 Signer와의 통합을 제공합니다. 리소스 섹션의 예제를 바탕으로 지속적 통합 및 지속적 전달(CI/CD) 파이프라인에 Signer를 통합하여 서명 검증과 자체 코드 및 이미지 서명을 자동화할 수 있습니다.

리소스

관련 문서:

- Cryptographic Signing for Containers
- Best Practices to help secure your container image build pipeline by using AWS Signer
- Announcing Container Image Signing with AWS Signer and Amazon EKS
- AWS Lambda에 대한 코드 서명 구성
- Best practices and advanced patterns for Lambda code signing
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys

관련 예제:

- Automate Lambda code signing with Amazon CodeCatalyst and AWS Signer
- Signing and Validating OCI Artifacts with AWS Signer

관련 도구:

- AWS Lambda
- AWS Signer
- AWS Certificate Manager
- AWS Key Management Service
- AWS CodeArtifact

SEC06-BP05 컴퓨팅 보호 자동화

컴퓨팅 보호 운영을 자동화하여 사람이 개입할 필요성을 줄입니다. 자동 스캔을 사용하여 컴퓨팅 리소스 내의 잠재적 문제를 탐지하고 자동화된 프로그래밍 방식 응답 또는 플릿 관리 작업을 통해 문제를 해결하세요. CI/CD 프로세스에 자동화를 통합하여 최신 종속성을 갖춘 신뢰할 수 있는 워크로드를 배포하세요.

원하는 성과: 자동화된 시스템이 컴퓨팅 리소스의 모든 스캔 및 패치를 수행합니다. 자동 검증을 사용하여 소프트웨어 이미지와 종속성이 신뢰할 수 있는 소스에서 비롯되었으며 변조되지 않았는지 확인합니다. 워크로드는 자동으로 최신 종속성을 확인하고 AWS 컴퓨팅 환경에서 신뢰성을 확보하도록 서명됩니다. 규정을 준수하지 않는 리소스가 탐지되면 자동 수정이 시작됩니다.

일반적인 안티 패턴:

- 변경 불가능한 인프라의 관행을 따르고 있지만, 프로덕션 시스템의 긴급 패치 적용 또는 교체를 위한 솔루션이 없습니다.
- 자동화를 사용하여 잘못 구성된 리소스를 수정하지만, 수동 재정의 메커니즘은 없습니다. 요구 사항을 조정해야 하는 상황이 발생할 수 있으며, 이러한 변경을 수행할 때까지 자동화를 일시 중단해야 할 수 있습니다.

이 모범 사례 확립의 이점: 자동화를 통해 컴퓨팅 리소스의 무단 액세스 및 사용 위험을 줄일 수 있습니다. 이를 통해 프로덕션 환경에 잘못된 구성이 유입되는 것을 방지하고 잘못된 구성이 발생할 경우 이를 탐지하여 수정할 수 있습니다. 자동화는 또한 무단 액세스 및 컴퓨팅 리소스 사용을 탐지하여 대응 시간을 줄이는 데 도움이 됩니다. 결과적으로 문제로 인한 전체 영향 범위를 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

보안 원칙 사례에 설명된 자동화를 적용하여 컴퓨팅 리소스를 보호할 수 있습니다. SEC06-BP01 취약성 관리 수행에서는 CI/CD 파이프라인에서 그리고 런타임 환경에서 알려진 일반적인 취약성 및 노출(CVE)을 지속적으로 스캔하는 경우에 Amazon Inspector를 사용하는 방법을 설명합니다. AWS Systems Manager를 사용하면 패치를 적용하거나 자동화된 런북을 통해 최신 이미지에서 재배포하여 컴퓨팅 플릿을 최신 소프트웨어 및 라이브러리로 업데이트할 수 있습니다. 이러한 기술을 사용하면 수동 프로세스는 물론 컴퓨팅 리소스에 대한 대화형 액세스의 필요성을 줄일 수 있습니다. 자세한 내용은 SEC06-BP03 수동 관리 및 대화형 액세스 감소를 참조하세요.

또한 자동화는 신뢰할 수 있는 워크로드를 배포하는 데도 중요한 역할을 합니다(SEC06-BP02 강화된 이미지로부터 컴퓨팅 프로비저닝 및 SEC06-BP04 소프트웨어 무결성 검증 참조). EC2 Image Builder, AWS Signer, AWS CodeArtifact, Amazon Elastic Container Registry(ECR)와 같은 서비스를 사용하여 강화 및 승인된 이미지와 코드 종속성을 다운로드, 확인, 구성 및 저장할 수 있습니다. Inspector와 함께 이들 각각은 CI/CD 프로세스에서 역할을 수행할 수 있으므로, 종속성이 신뢰할 수 있는 출처에서 비롯된 최신 상태인 점이 확인된 경우에만 워크로드가 프로덕션에 전달됩니다. 또한 AWS Lambda 및 Amazon Elastic Kubernetes Service(EKS)와 같은 AWS 컴퓨팅 환경에서 워크로드가 실행되기 전에 변조되지 않았는지 확인할 수 있도록 워크로드가 서명됩니다.

이러한 예방적 제어 외에도 컴퓨팅 리소스에 대한 탐지 제어에서도 자동화를 사용할 수 있습니다. 한 가지 예로, AWS Security Hub에서는 NIST 800-53 Rev. 5 표준을 제공합니다. 이 표준은 [EC2.8] EC2 인스턴스가 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용해야 한다는 등의 검사를 포함합니다. IMDSv2는 세션 인증 기술을 사용하여 X-Forwarded-For HTTP 헤더를 포함하는 요청을 차단하고 네트워크 TTL 1을 사용하여 외부 소스에서 발생하는 트래픽을 중지하고 EC2 인스턴스에 대한 정보를 검색합니다. Security Hub의 검사는 EC2 인스턴스가 IMDSv1을 사용하는 시기를 탐지하고 자동 수정을 시작할 수 있습니다. SEC04-BP04 규정 미준수 리소스 관련 문제 해결 시작에서 자동 탐지 및 수정에 대해 자세히 알아보세요.

구현 단계

1. EC2 Image Builder를 사용하여 안전하고 규정을 준수하며 강화된 AMI 생성을 자동화합니다. 기본 AWS 및 APN 파트너 이미지의 Center for Internet Security(CIS) Benchmarks 또는 Security Technical Implementation Guide(STIG) 표준의 제어를 통합하여 이미지를 생성할 수 있습니다.
2. 구성 관리를 자동화합니다. 구성 관리 서비스 또는 도구를 사용하여 컴퓨팅 리소스의 보안 구성을 자동으로 적용하고 검증합니다.
   a. AWS Config를 사용한 자동화된 구성 관리
   b. AWS Security Hub를 사용한 자동화된 보안 및 규정 준수 상태 관리

3. Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 패치 또는 교체를 자동화합니다. AWS Systems Manager Patch Manager는 보안 관련 업데이트와 기타 유형의 업데이트를 모두 사용하여 관리형 인스턴스를 패치하는 프로세스를 자동화합니다. 패치 관리자를 사용하면 운영 체제와 애플리케이션 모두에 패치를 적용할 수 있습니다.
   a. AWS Systems Manager Patch Manager
4. 일반적인 취약성 및 노출(CVE)에 대한 컴퓨팅 리소스 스캔을 자동화하고 빌드 파이프라인에 보안 스캔 솔루션을 포함시킵니다.
   a. Amazon Inspector
   b. ECR 이미지 스캔
5. 컴퓨팅 리소스를 보호하기 위한 자동 맬웨어 및 위협 탐지 기능에 대해 Amazon GuardDuty를 고려하세요. 또한 GuardDuty는 AWS 환경에서 AWS Lambda 함수가 간접 호출될 때 잠재적인 문제를 식별할 수 있습니다.
   a. Amazon GuardDuty
6. AWS 파트너 솔루션을 고려해 보세요. AWS 파트너는 사용자 온프레미스 환경의 기존 제어 솔루션과 동등한 수준이거나, 동일하거나, 통합된 업계 최고 수준의 제품을 제공합니다. 이러한 제품은 기존 AWS 서비스를 보완하여 클라우드 및 온프레미스 환경에 포괄적인 보안 아키텍처와 보다 원활한 환경을 배포할 수 있도록 해줍니다.
   a. 인프라 보안

리소스

관련 모범 사례:

- SEC01-BP06 표준 보안 제어의 배포 자동화

관련 문서:

- Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure

관련 비디오:

- Security best practices for the Amazon EC2 instance metadata service

데이터 보호

Questions

- SEC 7. 데이터는 어떻게 분류하나요?
- SEC 8. 저장 데이터는 어떻게 보호하나요?
- SEC 9. 전송 중 데이터는 어떻게 보호하나요?

SEC 7. 데이터는 어떻게 분류하나요?

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

모범 사례

- SEC07-BP01 데이터 분류 체계 이해
- SEC07-BP02 데이터 민감도에 따라 데이터 보호 제어 적용
- SEC07-BP03 식별 및 분류 자동화
- SEC07-BP04 확장 가능한 데이터 수명 주기 관리 정의

SEC07-BP01 데이터 분류 체계 이해

워크로드에서 처리 중인 데이터의 분류, 처리 요구 사항, 관련 비즈니스 프로세스, 데이터가 저장되는 위치, 데이터 소유자가 누구인지 이해하세요. 데이터 분류 및 처리 체계는 워크로드의 적용 가능한 법률 및 규정 준수 요구 사항과 필요한 데이터 제어 기능을 고려해야 합니다. 데이터를 이해해야 데이터 분류 여정을 시작할 수 있습니다.

원하는 성과: 워크로드에 있는 데이터 유형이 잘 이해되고 문서화되어 있습니다. 분류에 따라 민감한 데이터를 보호하기 위한 적절한 제어 조치가 마련되어 있습니다. 이러한 제어 기능은 누가 어떤 목적으로 데이터에 액세스할 수 있는지는 물론 데이터가 저장되는 위치, 해당 데이터에 대한 암호화 정책 및 암호화 키가 관리되는 방식, 데이터의 수명 주기 및 보존 요구 사항, 적절한 폐기 프로세스, 마련된 백업 및 복구 프로세스, 액세스 감사와 같은 고려 사항을 관리합니다.

일반적인 안티 패턴:

- 데이터 민감도 수준과 처리 요구 사항을 정의하는 공식적인 데이터 분류 정책이 마련되어 있지 않습니다.
- 워크로드 내 데이터의 민감도 수준을 제대로 이해하지 못하고 아키텍처 및 운영 문서에 해당 정보를 포함하지 않습니다.

- 데이터 분류 및 처리 정책에 명시된 대로 데이터의 민감도 및 요구 사항을 기반으로 데이터에 적절한 제어 기능을 적용하지 않습니다.
- 정책 소유자에게 데이터 분류 및 처리 요구 사항에 대한 피드백을 제공하지 않습니다.

이 모범 사례 확립의 이점: 이 방법을 사용하면 워크로드 내에서 데이터의 적절한 처리와 관련된 모호성이 사라집니다. 조직 내 데이터의 민감도 수준과 필수 보호 조치를 정의하는 공식 정책을 적용하면 법규와 기타 사이버 보안 증명 및 인증을 준수하는 데 도움이 될 수 있습니다. 워크로드 소유자는 민감한 데이터가 어디에 저장되고 어떤 보호 제어 기능이 적용되는지 알고 안심할 수 있습니다. 이러한 내용을 문서화하면 신입 팀원의 이해도를 높이고 업무를 배정받은 초반에도 제어 수준을 유지하도록 도울 수 있습니다. 그리고 각 데이터 유형에 대한 제어 규모를 적절하게 조정하여 비용을 절감하는 데도 도움이 될 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드를 설계할 때 민감한 데이터를 직관적으로 보호하는 방법을 고려할 수 있습니다. 예를 들어, 다중 테넌트 애플리케이션에서는 직관적으로 각 테넌트의 데이터를 민감한 것으로 생각하고 한 테넌트가 다른 테넌트의 데이터에 액세스할 수 없도록 보호 기능을 적용합니다. 마찬가지로 관리자만 데이터를 수정할 수 있고 다른 사용자는 읽기 수준의 액세스 권한만 가지거나 전혀 액세스할 수 없도록 액세스 제어를 직관적으로 설계할 수 있습니다.

이러한 데이터 민감도 수준을 데이터 보호 요구 사항과 함께 정의하고 정책에 포함하면 워크로드에 있는 데이터를 공식적으로 식별할 수 있습니다. 그런 다음 올바른 제어 기능이 있는지, 이를 감사할 수 있는지, 데이터가 잘못 처리되는 경우 어떻게 대응해야 적절한지 결정할 수 있습니다.

워크로드 내에서 민감한 데이터가 있는 위치를 식별하는 데 도움이 되도록 데이터 카탈로그를 사용하는 것이 좋습니다. 데이터 카탈로그는 조직의 데이터, 위치, 민감도 수준 및 해당 데이터를 보호하기 위해 마련된 제어를 매핑하는 데이터베이스입니다. 또한 가능한 경우 리소스 태그를 사용하는 것도 고려해 보세요. 예를 들어, 보호 의료 정보(PHI)에서 `Classification`의 태그 키와 `PHI`의 태그 값이 있는 태그 그리고 `Sensitivity`의 태그 키와 `High`의 태그 값이 있는 다른 태그를 적용할 수 있습니다. 그런 다음 AWS Config와 같은 서비스를 사용하여 리소스에서 변경 여부를 모니터링하고 보호 요구 사항을 준수하지 못하게 하는 방식으로 수정된 경우(예: 암호화 설정 변경) 알림을 보낼 수 있습니다. AWS Organizations의 기능인 태그 정책을 사용하여 태그 키의 표준 정의와 허용 가능한 값을 캡처할 수 있습니다. 태그 키 또는 값에 개인 데이터나 민감한 데이터는 포함하지 않는 것이 좋습니다.

구현 단계

1. 조직의 데이터 분류 체계 및 보호 요구 사항을 이해합니다.

2. 워크로드에서 처리하는 민감한 데이터의 유형을 식별합니다.
3. 조직에서 데이터가 있는 위치와 해당 데이터의 민감도 수준에 대한 단일 보기를 제공하는 데이터 카탈로그에서 데이터를 캡처합니다.
4. 가능한 경우 리소스 및 데이터 수준 태그 지정을 사용하여 민감도 수준과 모니터링 및 인시던트 대응에 도움이 될 수 있는 기타 운영 메타데이터로 데이터에 태그를 지정하는 것을 고려해 보세요.
   a. AWS Organizations 태그 정책은 태그 지정 표준을 적용하는 데 사용할 수 있습니다.

리소스

관련 모범 사례:

- SUS04-BP01 데이터 분류 정책 구현

관련 문서:

- Data Classification 백서
- Best Practices for Tagging AWS Resources

관련 예제:

- AWS Organizations Tag Policy Syntax and Examples

관련 도구

- AWS Tag Editor

SEC07-BP02 데이터 민감도에 따라 데이터 보호 제어 적용

분류 정책에 정의된 각 데이터 클래스에 대해 적절한 수준의 제어를 제공하는 데이터 보호 제어 기능을 적용합니다. 이렇게 하면 데이터의 가용성과 사용을 유지하면서 민감한 데이터를 무단 액세스 및 사용으로부터 보호할 수 있습니다.

원하는 성과: 조직의 데이터에 대한 다양한 민감도 수준을 정의하는 분류 정책이 있습니다. 이러한 민감도 수준 각각에 대해 승인된 보관 및 취급 서비스, 위치 및 필수 구성과 관련하여 명확한 지침이 게시되어 있습니다. 필요한 보호 수준 및 관련 비용에 따라 각 수준에 대해 제어 기능을 구현합니다. 데이터가 승인되지 않은 위치에 존재하거나, 무허가 환경에서 처리되거나, 무단 행위자가 액세스하거나, 관

련 서비스의 구성이 규정을 준수하지 않는 경우 등을 탐지할 수 있는 모니터링 및 알림 기능을 갖추고 있습니다.

일반적인 안티 패턴:

- 모든 데이터에 동일한 수준의 보호 제어 기능을 적용합니다. 이로 인해 민감도가 낮은 데이터에 대한 보안 제어가 과도하게 프로비저닝되거나 매우 민감한 데이터에 대한 보호가 불충분해질 수 있습니다.
- 데이터 보호 제어를 정의할 때 보안, 규정 준수 및 비즈니스 팀의 관련 이해관계자를 참여시키지 않습니다.
- 데이터 보호 제어의 구현 및 유지 관리와 관련된 운영 오버헤드 및 비용을 간과합니다.
- 분류 정책을 준수하기 위해 정기적인 데이터 보호 제어 검토를 수행하지 않습니다.
- 저장 상태일 때와 전송 중에 데이터의 위치에 대한 완전한 인벤토리가 없습니다.

이 모범 사례 확립의 이점: 데이터 분류 수준에 맞게 제어를 조정하면 조직은 필요한 경우 더 높은 수준의 제어에 투자할 수 있습니다. 여기에는 보안, 모니터링, 측정, 문제 해결 및 보고에 대한 리소스 확충이 포함될 수 있습니다. 적절한 제어 수단이 적을수록 직원, 고객 또는 구성원을 위한 데이터의 접근성과 완전성을 개선할 수 있습니다. 이 접근 방식을 통해 조직은 데이터 보호 요구 사항을 준수하면서 데이터를 최대한 유연하게 사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

데이터 민감도 수준을 기반으로 데이터 보호 제어 기능을 구현하려면 몇 가지 주요 단계를 거쳐야 합니다. 먼저 워크로드 아키텍처 내의 다양한 데이터 민감도 수준(예: 공개, 내부, 기밀, 제한)을 식별하고 이 데이터를 저장하고 처리하는 위치를 평가합니다. 다음으로 민감도 수준에 따라 데이터 주변의 격리 경계를 정의합니다. 서비스 제어 정책(SCP)을 통해 데이터 민감도 수준별로 허용되는 서비스 및 작업을 제한하여 데이터를 서로 다른 AWS 계정 계정으로 분리하는 것이 좋습니다. 이렇게 하면 강력한 격리 경계를 만들고 최소 권한 원칙을 적용할 수 있습니다.

격리 경계를 정의한 후 데이터 민감도 수준에 따라 적절한 보호 제어 기능을 구현합니다. 암호화, 액세스 제어 및 감사와 같은 관련 제어 기능을 구현하려면 저장 데이터 보호 및 전송 중 데이터 보호에 대한 모범 사례를 참조하세요. 토큰화나 익명화와 같은 기법을 고려하여 데이터의 민감도를 낮춥니다. 토큰화 및 탈토큰화를 위한 중앙 집중식 시스템을 통해 기업 전체에 일관된 데이터 정책을 간편하게 적용할 수 있습니다.

구현된 제어 기능의 효과를 지속적으로 모니터링하고 테스트합니다. 진화하는 조직 데이터 환경과 위협에 발맞춰 데이터 분류 체계, 위험 평가 및 보호 제어 기능을 정기적으로 검토하고 업데이트하세요. 구현된 데이터 보호 제어 기능을 관련 산업 규제, 표준 및 법적 요구 사항에 맞게 조정합니다. 또한, 직원이 데이터 분류 체계와 민감한 데이터의 취급 및 보호에 대한 책임을 이해할 수 있도록 보안 인식 및 교육을 제공합니다.

구현 단계

1. 워크로드 내 데이터의 분류 및 민감도 수준을 식별합니다.
2. 각 수준에 대한 격리 경계를 정의하고 시행 전략을 결정합니다.
3. 액세스, 암호화, 감사, 보존 및 데이터 분류 정책에 필요한 기타 사항을 관리하기 위해 정의한 제어 기능을 평가합니다.
4. 적절한 경우 토큰화 또는 익명화 사용 등 데이터의 민감도 수준을 낮추는 옵션을 평가합니다.
5. 구성된 리소스의 자동화된 테스트 및 모니터링을 사용하여 제어 기능을 확인합니다.

리소스

관련 모범 사례:

- PERF03-BP01 데이터 액세스 및 스토리지 요구 사항을 가장 잘 지원하는 목적별 데이터 스토어 사용
- COST04-BP05 데이터 보존 정책 적용

관련 문서:

- Data Classification 백서
- 보안, 자격 증명 및 규정 준수를 위한 모범 사례
- AWS KMS 모범 사례
- Encryption best practices and features for AWS services

관련 예제:

- Building a serverless tokenization solution to mask sensitive data
- How to use tokenization to improve data security and reduce audit scope

관련 도구:

- AWS Key Management Service (AWS KMS)
- AWS CloudHSM
- AWS Organizations

SEC07-BP03 식별 및 분류 자동화

데이터 식별 및 분류를 자동화하면 올바른 제어를 구현하는 데 도움이 될 수 있습니다. 자동화를 사용하여 수동 결정을 보강하면 인적 오류 및 노출의 위험을 줄일 수 있습니다.

원하는 성과: 분류 및 취급 정책에 따라 적절한 통제가 마련되어 있는지 확인할 수 있습니다. 자동화된 도구 및 서비스는 데이터의 민감도 수준을 식별하고 분류하는 데 도움이 됩니다. 나아가 자동화를 통해 환경을 지속적으로 모니터링하여 데이터가 무단으로 저장되거나 처리되는 경우 이를 감지하고 알림을 보내 시정 조치를 신속하게 취할 수 있습니다.

일반적인 안티 패턴:

- 데이터 식별 및 분류를 위한 수동 프로세스에만 의존하므로, 오류가 발생하기 쉽고 시간이 많이 걸릴 수 있습니다. 이로 인해 특히 데이터 볼륨이 증가하면 데이터 분류의 효율성과 일관성이 떨어질 수 있습니다.
- 조직 전체에 걸쳐 데이터 자산을 추적하고 관리하는 메커니즘이 없습니다.
- 조직 내에서 이동하고 변화하는 데이터를 지속적으로 모니터링하고 분류해야 할 필요성을 간과합니다.

이 모범 사례 확립의 이점: 데이터 식별 및 분류를 자동화하면 데이터 보호 제어를 보다 일관되고 정확하게 적용하여 인적 오류의 위험을 줄일 수 있습니다. 자동화는 또한 민감한 데이터 액세스 및 이동에 대한 가시성을 제공하여 무단 처리를 탐지하고 시정 조치를 취하는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

워크로드의 초기 설계 단계에서 데이터를 분류할 때는 사람이 판단하는 경우가 많지만, 예방 차원에서 테스트 데이터의 식별 및 분류를 자동화하는 시스템을 마련하는 것을 고려해 보세요. 예를 들어, 개발자에게 대표 데이터를 스캔하여 민감도를 결정하는 도구나 서비스를 제공할 수 있습니다. AWS에서 Amazon S3에 데이터세트를 업로드하고 Amazon Macie, Amazon Comprehend 또는 Amazon Comprehend Medical을 사용하여 스캔할 수 있습니다. 마찬가지로 단위 및 통합 테스트의 일환으로 데이터를 스캔하여 민감한 데이터가 예상되지 않는 부분을 찾아내는 것도 고려해 보세요. 이 단계에

서 민감한 데이터에 대한 알림을 통해 프로덕션에 배포하기 전에 보호의 허점을 강조할 수 있습니다. AWS Glue, Amazon SNS, Amazon CloudWatch에서 민감한 데이터 탐지와 같은 기타 기능을 사용하여 PII를 탐지하고 완화 조치를 수행할 수도 있습니다. 자동화된 도구 또는 서비스의 경우, 민감한 데이터를 정의하는 방법을 이해하고 필요에 따라 다른 사람이나 자동화된 솔루션으로 이를 보강하여 허점을 메우세요.

탐지 제어 기능으로 환경을 지속적으로 모니터링하여 민감한 데이터가 규정을 준수하지 않는 방식으로 저장되고 있는지 탐지하세요. 이를 통해 민감한 데이터가 로그 파일로 내보내지거나, 적절히 식별되지 않거나 수정되지 않고 데이터 분석 환경으로 복사되는 등의 상황을 탐지할 수 있습니다. Amazon S3에 저장된 데이터는 Amazon Macie를 사용하여 민감한 데이터가 있는지 지속적으로 모니터링할 수 있습니다.

구현 단계

1. SEC07-BP01에 설명된 조직 내 데이터 분류 체계를 검토합니다.
   a. 조직의 데이터 분류 체계를 이해하면 회사 정책에 맞는 자동 식별 및 분류를 위한 정확한 프로세스를 수립할 수 있습니다.
2. 자동 식별 및 분류를 위해 환경에 대한 초기 스캔을 수행합니다.
   a. 초기에 데이터를 전체적으로 스캔하면 민감한 데이터가 사용자 환경의 어디에 있는지 포괄적으로 이해하는 데 도움이 될 수 있습니다. 처음에 전체 스캔이 필요하지 않거나 비용 때문에 미리 완료할 수 없다면 데이터 샘플링 기술이 성과를 달성하는 데 적합한지 평가하세요. 예를 들어, Amazon Macie를 사용하여 S3 버킷에서 광범위하고 자동화된 민감한 데이터 검색 작업을 수행하도록 구성할 수 있습니다. 이 기능은 샘플링 기술을 사용하여 민감한 데이터가 있는 위치에 대한 예비 분석을 비용 효율적으로 수행합니다. 그런 다음 민감한 데이터 검색 작업을 사용하여 S3 버킷에 대한 심층 분석을 수행할 수 있습니다. 다른 데이터 스토어를 S3로 내보내 Macie에서 스캔할 수도 있습니다.
   b. 스캔으로 식별된 데이터 스토리지 리소스에 대해 SEC07-BP02에 정의된 액세스 제어를 설정합니다.
3. 환경에 대한 지속적인 스캔을 구성합니다.
   a. Macie의 자동화된 중요 데이터 검색 기능을 사용하여 환경을 지속적으로 스캔할 수 있습니다. 민감한 데이터를 저장할 권한이 있는 알려진 S3 버킷은 Macie의 허용 목록을 사용하여 제외할 수 있습니다.
4. 식별 및 분류를 구축 및 테스트 프로세스에 통합합니다.
   a. 워크로드가 개발 중인 동안 개발자가 데이터의 민감도를 스캔하는 데 사용할 수 있는 도구를 식별합니다. 이러한 도구를 통합 테스트의 일부로 사용하여 민감한 데이터가 예상되지 않는 경우 이를 알리고 추가 배포를 방지할 수 있습니다.

5. 승인되지 않은 위치에서 민감한 데이터가 발견될 경우 조치를 취할 수 있는 시스템 또는 런북을 구현합니다.
   a. 자동 수정을 사용하여 데이터에 대한 액세스를 제한합니다. 예를 들어, 속성 기반 액세스 제어(ABAC)를 사용하는 경우 이 데이터를 액세스가 제한된 S3 버킷으로 이동하거나 객체에 태그를 지정할 수 있습니다. 또한 데이터가 감지될 때 마스킹하는 것도 고려해 보세요.
   b. 데이터 보호 및 인시던트 대응 팀에 알려 인시던트의 근본 원인을 조사합니다. 이들이 찾아내는 모든 교훈은 향후 인시던트를 방지하는 데 도움이 될 수 있습니다.

리소스

관련 문서:

- AWS Glue: Detect and process sensitive data
- Using managed data identifiers in Amazon SNS
- Amazon CloudWatch Logs: Help protect sensitive log data with masking

관련 예제:

- Enabling data classification for Amazon RDS database with Macie
- Detecting sensitive data in DynamoDB with Macie

관련 도구:

- Amazon Macie
- Amazon Comprehend
- Amazon Comprehend Medical
- AWS Glue

SEC07-BP04 확장 가능한 데이터 수명 주기 관리 정의

다양한 수준의 데이터 분류 및 처리와 관련된 데이터 수명 주기 요구 사항을 파악하세요. 여기에는 데이터가 환경에 처음 유입되었을 때 데이터를 처리하는 방법, 데이터가 변환되는 방식, 데이터 폐기 규칙이 포함될 수 있습니다. 보존 기간, 액세스, 감사, 출처 추적과 같은 요소를 고려하세요.

원하는 성과: 데이터를 수집 시점 및 시간에 최대한 가깝게 분류합니다. 데이터 분류에 마스킹, 토큰화 또는 민감도 수준을 낮추는 기타 프로세스가 필요한 경우 수집 시점과 시간에 최대한 가깝게 작업을 수행하세요.

더 이상 보관하기에 적절하지 않은 경우 정책에 따라 분류를 기반으로 데이터를 삭제합니다.

일반적인 안티 패턴:

- 다양한 민감도 수준과 액세스 요구 사항을 고려하지 않고 데이터 수명 주기 관리에 대한 획일적인 접근 방식을 구현합니다.
- 사용 가능한 데이터 또는 백업된 데이터의 관점에서만 수명 주기 관리를 고려하고, 모두 고려하지 않습니다.
- 가치나 출처를 확인하지 않고 워크로드에 입력된 데이터가 유효하다고 가정합니다.
- 데이터 백업 및 보호 대신 데이터 내구성에 의존합니다.
- 유용성 및 필수 보존 기간을 초과하여 데이터를 보존합니다.

이 모범 사례 확립의 이점: 잘 정의되고 확장 가능한 데이터 수명 주기 관리 전략은 적절한 제어를 유지하면서 규제를 계속해서 준수하고, 데이터 보안을 개선하며, 스토리지 비용을 최적화하는 데 도움을 줄 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드 내의 데이터는 동적인 경우가 많습니다. 워크로드 환경에 유입될 때 취하는 형태는 비즈니스 로직, 보고, 분석 또는 기계 학습에 저장되거나 사용될 때와 다를 수 있습니다. 또한, 데이터의 가치는 시간이 지남에 따라 변할 수 있습니다. 일부 데이터는 본질적으로 일시적이며, 오래될수록 가치를 잃습니다. 데이터에 대한 이러한 변경이 데이터 분류 체계 및 관련 제어에 따른 평가에 어떤 영향을 미치는지 생각해 보세요. 가능한 경우 Amazon S3 수명 주기 정책 및 Amazon Data Lifecycle Manager와 같은 자동화된 수명 주기 메커니즘을 사용하여 데이터 보존, 아카이빙 및 만료 프로세스를 구성합니다. DynamoDB에 저장된 데이터의 경우 Time To Live(TTL) 기능을 사용하여 항목별 만료 타임스탬프를 정의할 수 있습니다.

사용할 수 있는 데이터와 백업으로 저장된 데이터를 구분합니다. AWS 서비스 전반의 데이터 백업을 자동화하기 위해 AWS Backup 사용을 고려하세요. Amazon EBS 스냅샷은 수명 주기, 데이터 보호, 보호 메커니즘에 대한 액세스를 비롯한 S3 기능을 사용하여 EBS 볼륨을 복사하고 저장하는 방법을 제공합니다. 이 두 가지 메커니즘은 S3 객체 잠금 및 AWS Backup 볼트 잠금으로, 이를 통해 백업에 대한 추가적인 보안 및 제어를 제공할 수 있습니다. 백업에 대한 업무와 액세스 권한을 명확하게 구분하여

관리합니다. 계정 수준에서 백업을 격리하여 이벤트 발생 시 영향을 받는 환경으로부터 분리할 수 있습니다.

수명 주기 관리의 또 다른 측면은 워크로드가 진행되는 동안 데이터 내역을 기록하는 것입니다. 이를 데이터 출처 추적이라고 합니다. 따라서 데이터의 출처, 수행한 변환, 변경한 소유자 또는 프로세스, 변경 시기를 확실하게 알 수 있습니다. 이 기록이 있으면 잠재적 보안 이벤트 발생 시 문제 해결 및 조사에 도움이 됩니다. 예를 들어 Amazon DynamoDB 테이블에 변환에 대한 메타데이터를 로깅할 수 있습니다. 데이터 레이크 내에서 각 데이터 파이프라인 단계의 서로 다른 S3 버킷에 변환된 데이터의 복사본을 보관할 수 있습니다. AWS Glue Data Catalog에 스키마와 타임스탬프 정보를 저장합니다. 그리고 솔루션에 관계없이 최종 사용자의 요구 사항을 고려하여 데이터 출처를 보고하는 데 필요한 적절한 도구를 결정하세요. 이렇게 하면 출처를 가장 잘 추적하는 방법을 결정하는 데 도움이 됩니다.

구현 단계

1. 워크로드의 데이터 유형, 민감도 수준 및 액세스 요구 사항을 분석하여 데이터를 분류하고 적절한 수명 주기 관리 전략을 정의합니다.
2. 법률, 규제 및 조직 요구 사항에 맞는 데이터 보존 정책 및 자동 폐기 프로세스를 설계하고 구현합니다.
3. 변화하는 워크로드 요구 사항 및 규제 변화에 맞춰 데이터 수명 주기 관리 전략, 제어, 정책을 지속적으로 모니터링, 감사 및 조정하기 위한 프로세스와 자동화를 수립합니다.
   a. AWS Config를 사용하여 자동 수명 주기 관리가 활성화되지 않은 리소스 감지

리소스

관련 모범 사례:

- COST04-BP05 데이터 보존 정책 적용
- SUS04-BP03 정책을 사용하여 데이터세트의 수명 주기 관리

관련 문서:

- Data Classification 백서
- AWS Blueprint for Ransomware Defense
- DevOps Guidance: Improve traceability with data provenance tracking

관련 예제:

- How to protect sensitive data for its entire lifecycle in AWS
- Build data lineage for data lakes using AWS Glue, Amazon Neptune, and Spline

관련 도구:

- AWS Backup
- Amazon Data Lifecycle Manager
- AWS Identity and Access Management Access Analyzer

SEC 8. 저장 데이터는 어떻게 보호하나요?

여러 제어를 구현하여 무단 액세스 또는 처리 오류의 위험을 줄여 저장 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-BP01 보안 키 관리 구현

보안 키 관리에는 워크로드의 저장 데이터를 보호하는 데 필요한 키 구성 요소의 저장, 순환, 액세스 제어 및 모니터링이 포함됩니다.

원하는 성과: 확장 가능하고 반복 가능하며 자동화된 키 관리 메커니즘이 있습니다. 메커니즘이 키 구성 요소에 대한 액세스 권한을 최소한으로 제한하고 키 가용성, 기밀성 및 무결성 간에 적절한 균형을 유지합니다. 키에 대한 액세스를 모니터링하고 키 구성 요소의 교체가 필요한 경우 자동화된 프로세스를 사용하여 키를 교체합니다. 인간 작업자가 키 구성 요소에 액세스하도록 허용하지 않습니다.

일반적인 안티 패턴:

- 암호화되지 않은 키 구성 요소에 대한 인적 접근이 가능합니다.
- 사용자 지정 암호화 알고리즘을 생성합니다.
- 키 구성 요소에 액세스할 수 있는 권한이 지나치게 광범위합니다.

이 모범 사례 확립의 이점: 워크로드에 대한 보안 키 관리 메커니즘을 구축하면 무단 액세스로부터 콘텐츠를 보호하는 데 도움이 될 수 있습니다. 또한 데이터를 암호화하기 위한 규제 요구 사항이 적용될 수 있습니다. 효과적인 키 관리 솔루션은 키 구성 요소를 보호하기 위해 해당 규정에 맞는 기술적 메커니즘을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

저장 데이터의 암호화는 기본적인 보안 제어입니다. 이러한 제어를 구현하려면 워크로드에 저장 데이터를 암호화하는 데 사용되는 키 구성 요소를 안전하게 저장하고 관리하는 메커니즘이 필요합니다.

AWS는 AWS Key Management Service(AWS KMS)를 제공하여 내구성이 뛰어나고 안전하며 중복된 AWS KMS 키 스토리지를 제공합니다. 많은 AWS 서비스가 AWS KMS에 통합되어 데이터 암호화를 지원합니다. AWS KMS에서는 FIPS 140-2 레벨 3 검증 하드웨어 보안 모듈을 사용하여 키를 보호합니다. AWS KMS 키를 일반 텍스트로 내보내는 메커니즘은 없습니다.

다중 계정 전략을 사용하여 워크로드를 배포할 때 AWS KMS 키를 사용하는 워크로드와 동일한 계정에 키를 유지해야 합니다. 이 분산 모델에서는 AWS KMS 키 관리에 대한 책임이 사용자의 팀에 있습니다. 다른 사용 사례에서는 조직에서 AWS KMS 키를 중앙 집중식 계정에 저장하도록 선택할 수 있습니다. 이 중앙 집중식 구조에는 워크로드 계정이 중앙 집중식 계정에 저장된 키에 액세스하는 데 필요한 크로스 계정 액세스를 가능하게 하는 추가 정책이 필요하지만 단일 키를 여러 AWS 계정에서 공유하는 사용 사례에서는 더 적합할 수 있습니다.

키 구성 요소의 보관 위치와 관계없이 키 정책 및 IAM 정책을 사용하여 키에 대한 액세스를 엄격하게 제어해야 합니다. 키 정책은 AWS KMS 키에 대한 액세스를 제어하는 기본 방법입니다. 또한 AWS KMS 키 부여는 사용자를 대신하여 데이터를 암호화 및 복호화하는 AWS 서비스에 대한 액세스를 제공할 수 있습니다. AWS KMS 키에 대한 액세스 제어 지침을 검토합니다.

암호화 키 사용을 모니터링하여 비정상적인 액세스 패턴을 탐지해야 합니다. AWS KMS에 저장된 고객 관리형 키와 AWS 관리형 키를 사용하여 수행한 작업은 AWS CloudTrail에 로그인할 수 있으며 정기적으로 검토해야 합니다. 키 폐기 이벤트를 모니터링하는 데 특히 주의를 기울이세요. 키 구성 요소의 우발적 또는 악의적 폐기를 방지하기 위해 키 폐기 이벤트는 키 구성 요소를 즉시 삭제하지 않습니다. AWS KMS에서 키 삭제 시도에는 대기 시간이 적용됩니다. 기본값은 30일, 최솟값은 7일이므로 관리자는 이러한 작업을 검토하고 필요한 경우 요청을 롤백할 시간을 확보할 수 있습니다.

대부분의 AWS 서비스는 사용자에게 투명한 방식으로 AWS KMS를 사용합니다. AWS 관리형 키를 사용할지 아니면 고객 관리형 키를 사용할지 결정하는 것만이 유일한 요구 사항입니다. 워크로드에서 데이터를 암호화하거나 복호화하는 데 AWS KMS를 직접 사용해야 하는 경우 봉투 암호화를 사용하여

데이터를 보호해야 합니다. AWS Encryption SDK에서는 애플리케이션에 클라이언트측 암호화 기본 요소를 제공하여 봉투 암호화를 구현하고 AWS KMS와 통합할 수 있습니다.

구현 단계

1. 키에 적합한 키 관리 옵션(AWS 관리형 또는 고객 관리형)을 결정합니다.
   a. 사용 편의성을 위해 AWS에서는 대부분의 서비스에 대해 AWS 소유 및 AWS 관리형 키를 제공하며, 키 구성 요소나 키 정책을 관리할 필요 없이 저장 시 암호화 기능을 제공합니다.
   b. 고객 관리형 키를 사용할 때는 민첩성, 보안, 데이터 주권, 가용성 사이에서 최상의 균형을 유지할 수 있도록 기본 키 스토어를 고려하세요. 다른 사용 사례에서는 AWS CloudHSM 또는 외부 키 저장소와 함께 사용자 지정 키 저장소를 사용해야 할 수도 있습니다.
2. 워크로드에 사용 중인 서비스 목록을 검토하여 서비스와의 AWS KMS 통합 방식을 파악하세요. 예를 들어, EC2 인스턴스는 암호화된 EBS 볼륨을 사용하여 해당 볼륨에서 생성된 Amazon EBS 스냅샷도 고객 관리형 키를 사용하여 암호화되는지 확인하고 암호화되지 않은 스냅샷 데이터가 우발적으로 공개되는 것을 방지할 수 있습니다.
   a. How AWS services use AWS KMS
   b. AWS 서비스에서 제공하는 암호화 옵션에 대한 자세한 내용은 해당 서비스 사용 설명서 또는 개발자 안내서의 저장된 데이터 암호화 주제를 참조하세요.
3. AWS KMS 구현: AWS KMS는 다양한 AWS 서비스와 애플리케이션에서 간단하게 키를 생성 및 관리하고 암호화 사용을 제어할 수 있습니다.
   a. Getting started: AWS Key Management Service (AWS KMS)
   b. AWS KMS 키에 대한 액세스 제어 모범 사례를 검토합니다.
4. AWS Encryption SDK 고려: 애플리케이션이 클라이언트 측 데이터를 암호화해야 하는 경우 AWS Encryption SDK를 AWS KMS와 통합하세요.
   a. AWS Encryption SDK
5. AWS KMS 키 정책이 지나치게 광범위한지 자동으로 검토하고 알리도록 IAM Access Analyzer를 활성화합니다.
   a. 사용자 지정 정책 확인을 사용하여 리소스 정책 업데이트가 KMS 키에 대한 퍼블릭 액세스 권한을 부여하지 않는지 확인하는 것이 좋습니다.
6. Security Hub를 활성화하여 잘못 구성된 키 정책, 삭제 예정 키 또는 자동 순환이 활성화되지 않은 키가 있는 경우 알림을 받습니다.
7. AWS KMS 키에 적합한 로깅 수준을 결정하세요. 읽기 전용 이벤트를 포함하여 AWS KMS에 대한 직접 호출이 로깅되므로 AWS KMS에 연결된 CloudTrail 로그의 양이 많아질 수 있습니다.

a. 일부 조직에서는 AWS KMS 로깅 활동을 별도의 트레일로 분리하는 것을 선호합니다. 자세한 내용은 AWS KMS 개발자 안내서의 Logging AWS KMS API calls with CloudTrail 섹션을 참조하세요.

리소스

관련 문서:

- AWS Key Management Service
- AWS cryptographic services and tools
- 암호화로 Amazon S3 데이터 보호
- 봉투 암호화
- Digital sovereignty pledge
- Demystifying AWS KMS key operations, bring your own key, custom key store, and ciphertext portability
- AWS Key Management Service cryptographic details

관련 비디오:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS
- AWS data protection: Using locks, keys, signatures, and certificates

관련 예제:

- Implement advanced access control mechanisms using AWS KMS

SEC08-BP02 저장 시 암호화 적용

저장 상태의 프라이빗 데이터를 암호화하여 기밀성을 유지하고 의도하지 않은 데이터 공개 또는 유출에 대한 추가 보호 계층을 제공합니다. 암호화는 먼저 복호화되지 않고는 데이터를 읽거나 액세스할 수 없도록 하여 데이터를 보호합니다. 암호화되지 않은 데이터의 인벤토리를 만들고 제어하여 데이터 노출과 관련된 위험을 완화합니다.

원하는 성과: 저장 시 기본적으로 프라이빗 데이터를 암호화하는 메커니즘이 있습니다. 이러한 메커니즘은 데이터의 기밀성을 유지하는 데 도움이 되며 의도치 않은 데이터 공개 또는 유출에 대한 추가 보

호 계층을 제공합니다. 암호화되지 않은 데이터의 인벤토리를 유지하고 이를 보호하기 위해 마련된 제어를 이해합니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드 내의 데이터 분류에 암호화 키를 매핑합니다. 이 접근 방식은 데이터에 단일 또는 매우 적은 수의 암호화 키를 사용할 때 지나치게 허용적인 액세스 권한을 방지하는 데 도움이 됩니다(SEC07-BP01 데이터 분류 체계 이해 참조).

AWS Key Management Service(AWS KMS)는 많은 AWS 서비스와 통합되어 저장 데이터를 보다 쉽게 암호화할 수 있습니다. 예를 들어 Amazon Elastic Compute Cloud(Amazon EC2)에서 새 EBS 볼륨이 자동으로 암호화되도록 계정에 기본 암호화를 설정할 수 있습니다. 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 Simple Storage Service(Amazon S3)와 같은 서비스는 이제 기본적으로 모든 새 객체를 암호화합니다. 이 구현은 성능에 영향을 주지 않으면서 향상된 보안을 제공합니다.

Amazon Elastic Compute Cloud(Amazon EC2) 또는 Amazon Elastic File System(Amazon EFS)과 같은 기타 서비스는 기본 암호화 설정을 지원합니다. AWS Config 규칙을 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 SDK에서는 봉투 암호화를 사용하여 데이터를 암호화하는 방법을 제공합니다. 래핑 키를 제공하면 AWS Encryption SDK가 암호화하는 각 데이터 객체에 대해 고유한 데

이터 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

- 새로운 Amazon EBS 볼륨에 대해 기본 암호화 구성: AWS에서 제공하는 기본 키 또는 사용자가 생성한 키를 사용하는 옵션을 통해, 새로 생성되는 모든 Amazon EBS 볼륨이 암호화된 형식으로 생성되도록 지정합니다.
- 암호화된 Amazon Machine Image(AMI) 구성: 암호화가 구성된 상태에서 기존 AMI를 복사하면 루트 볼륨 및 스냅샷을 자동으로 암호화합니다.
- Amazon RDS 암호화 구성: 암호화 옵션을 사용하여 Amazon RDS 데이터베이스 클러스터 및 저장된 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 위해 하나의 AWS KMS 키를 생성하고 개발 또는 테스트 데이터 암호화를 위해 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 설명서를 검토하여 서비스의 암호화 옵션을 결정합니다.

리소스

관련 문서:

- AWS 암호화 도구
- AWS Encryption SDK
- AWS KMS Cryptographic Details 백서
- AWS Key Management Service

- AWS cryptographic services and tools
- Amazon EBS Encryption
- Default encryption for Amazon EBS volumes
- Amazon RDS 리소스 암호화
- How do I enable default encryption for an Amazon S3 bucket?
- 암호화로 Amazon S3 데이터 보호

관련 비디오:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP03 저장 데이터 보호 자동화

자동화를 사용하여 저장된 데이터 제어를 검증하고 적용하세요. 자동 스캐닝을 사용하여 데이터 스토리지 솔루션의 잘못된 구성을 탐지하고, 가능하다면 자동화된 프로그래밍 방식 응답을 통해 수정을 수행합니다. CI/CD 프로세스에 자동화를 통합하여 프로덕션 환경에 배포하기 전에 데이터 스토리지 구성 오류를 감지할 수 있습니다.

원하는 성과: 자동화된 시스템이 데이터 스토리지 위치를 스캔하고 모니터링하여 제어 기능의 잘못된 구성, 무단 액세스 및 예상치 못한 사용이 있는지 확인합니다. 잘못 구성된 스토리지 위치를 탐지하면 자동 수정이 시작됩니다. 자동화된 프로세스는 데이터 백업을 생성하고 원본 환경 외부에 변경 불가능한 사본을 저장합니다.

일반적인 안티 패턴:

- 지원되는 경우에 기본 설정으로 암호화를 활성화하는 옵션을 고려하고 있지 않습니다.
- 자동 백업 및 복구 전략을 수립할 때 운영 이벤트 외에 보안 이벤트는 고려하지 않습니다.
- 스토리지 서비스에 대한 공개 액세스 설정을 적용하지 않습니다.
- 저장 데이터를 보호하기 위한 제어 기능을 모니터링하고 감사하지 않습니다.

이 모범 사례 확립의 이점: 자동화는 데이터 스토리지 위치를 잘못 구성할 위험을 방지하는 데 도움이 됩니다. 프로덕션 환경에 잘못된 구성이 유입되는 것을 방지하는 데 도움이 됩니다. 이 모범 사례는 잘못된 구성이 발생할 경우 이를 탐지하고 수정하는 데도 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

자동화는 저장 데이터를 보호하기 위한 관행 전반에 적용되는 주제입니다. SEC01-BP06 표준 보안 제어의 배포 자동화에서는 AWS CloudFormation과 같이 코드형 인프라(IaC) 템플릿을 사용하여 리소스 구성을 캡처하는 방법을 설명합니다. 이러한 템플릿은 버전 제어 시스템에 적용되며 CI/CD 파이프라인을 통해 AWS에 리소스를 배포하는 데 사용됩니다. 이러한 기술은 Amazon S3 버킷의 암호화 설정과 같은 데이터 스토리지 솔루션의 구성을 자동화하는 데도 동일하게 적용됩니다.

AWS CloudFormation Guard의 규칙을 사용하여 IaC 템플릿에서 정의한 설정에 CI/CD 파이프라인의 잘못된 구성이 있는지 확인할 수 있습니다. AWS Config를 통해 CloudFormation 또는 기타 IaC 도구에서 아직 사용할 수 없는 설정에 잘못된 구성이 있는지 모니터링할 수 있습니다. Config가 잘못된 구성에 대해 생성하는 알림은 SEC04-BP04 비준수 리소스에 대한 문제 해결 시작에서 설명한 대로 자동으로 수정할 수 있습니다.

권한 관리 전략의 일부로 자동화를 사용하는 것도 자동화된 데이터 보호의 필수 구성 요소입니다. SEC03-BP02 최소 권한 액세스 부여 및 SEC03-BP04 지속적으로 권한 축소에서는 최소 권한 액세스 정책을 구성하는 방법을 설명합니다. AWS Identity and Access Management Access Analyzer에서는 이 정책을 지속적으로 모니터링하여 권한을 축소할 수 있는 경우 조사 결과를 생성합니다. 권한 모니터링에 대한 자동화 외에도 EBS 볼륨(EC2 인스턴스를 통해), S3 버킷 및 지원되는 Amazon Relational Database Service 데이터베이스에 대한 비정상적인 데이터 액세스 동작을 감시하도록 Amazon GuardDuty를 구성할 수 있습니다.

자동화는 민감한 데이터가 승인되지 않은 위치에 저장되는 시점을 탐지하는 역할도 합니다. SEC07-BP03 식별 및 분류 자동화에서는 Amazon Macie가 S3 버킷에서 예기치 못한 민감한 데이터를 모니터링하고 자동화된 응답을 시작할 수 있는 알림을 생성하는 방법에 대해 설명합니다.

REL09 데이터 백업의 관행에 따라 자동화된 데이터 백업 및 복구 전략을 개발합니다. 데이터 백업 및 복구는 운영 이벤트와 마찬가지로 보안 이벤트 복구에도 중요합니다.

구현 단계

1. IaC 템플릿에서 데이터 스토리지 구성을 캡처합니다. CI/CD 파이프라인의 자동 검사를 사용하여 구성 오류를 감지합니다.
   a. AWS CloudFormation을 IaC 템플릿에 사용할 수 있으며, AWS CloudFormation Guard를 사용하여 템플릿에 잘못된 구성이 있는지 확인할 수 있습니다.
   b. AWS Config를 사용하여 사전 평가 모드에서 규칙을 실행합니다. 이 설정을 사용하면 리소스를 생성하기 전에 CI/CD 파이프라인의 한 단계로 리소스의 규정 준수를 확인할 수 있습니다.
2. 리소스에서 데이터 스토리지 구성 오류를 모니터링합니다.

   a. 데이터 스토리지 리소스에서 제어 구성의 변경 사항을 모니터링하고 잘못된 구성이 감지되면 수정 조치를 간접 호출하는 알림을 생성하도록 AWS Config를 설정합니다.
   b. 자동 수정에 대한 자세한 지침은 SEC04-BP04 비준수 리소스에 대한 수정 시작을 참조하세요.
3. 자동화를 통해 데이터 액세스 권한을 지속적으로 모니터링하고 줄입니다.
   a. IAM Access Analyzer를 지속적으로 실행하여 권한이 축소될 가능성이 있는 경우 알림을 생성할 수 있습니다.
4. 비정상적인 데이터 액세스 동작을 모니터링하고 알림을 보냅니다.
   a. GuardDuty는 EBS 볼륨, S3 버킷, RDS 데이터베이스와 같은 데이터 스토리지 리소스에 대한 기본 액세스 동작과의 편차와 알려진 위협 서명을 모두 감시합니다.
5. 예상치 못한 위치에 저장되는 민감한 데이터를 모니터링하고 알림을 보냅니다.
   a. Amazon Macie를 사용하여 S3 버킷에서 민감한 데이터를 지속적으로 스캔합니다.
6. 안전하고 암호화된 데이터 백업을 자동화합니다.
   a. AWS Backup은 AWS에서 다양한 데이터 소스의 암호화되고 안전한 백업을 생성하는 관리형 서비스입니다. Elastic Disaster Recovery를 사용하면 전체 서버 워크로드를 복사하고 초 단위로 측정된 Recovery Point Objective(RPO)로 지속적인 데이터 보호를 유지 관리할 수 있습니다. 두 서비스가 함께 작동하도록 구성하여 데이터 백업 생성 및 장애 조치 위치에 복사하는 작업을 자동화할 수 있습니다. 이렇게 하면 운영 또는 보안 이벤트의 영향을 받는 경우에도 데이터를 계속 사용할 수 있습니다.

리소스

관련 모범 사례:

- SEC01-BP06 표준 보안 제어의 배포 자동화
- SEC03-BP02 최소 권한 액세스 부여
- SEC03-BP04 지속적으로 권한 축소
- SEC04-BP04 규정 미준수 리소스 관련 문제 해결 시작
- SEC07-BP03 식별 및 분류 자동화
- REL09-BP02 백업 보안 및 암호화
- REL09-BP03 자동으로 데이터 백업 수행

관련 문서:

- AWS Prescriptive Guidance: Automatically encrypt existing and new Amazon EBS volumes

- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework (CSF)

관련 예제:

- How to use AWS Config proactive rules and AWS CloudFormation Hooks to prevent creation of noncompliant cloud resources
- Automate and centrally manage data protection for Amazon S3 with AWS Backup
- AWS re:Invent 2023 - Implement proactive data protection using Amazon EBS snapshots
- AWS re:Invent 2022 - Build and automate for resilience with modern data protection

관련 도구:

- AWS CloudFormation Guard
- AWS CloudFormation Guard Rules Registry
- IAM 액세스 분석기
- Amazon Macie
- AWS Backup
- Elastic Disaster Recovery

SEC08-BP04 액세스 제어 적용

저장 데이터를 보호하려면 격리 및 버전 관리와 같은 메커니즘을 사용하여 액세스 제어를 적용합니다. 최소 권한 및 조건부 액세스 제어를 적용합니다. 데이터에 대한 퍼블릭 액세스 권한 부여를 방지합니다.

원하는 성과: 인증된 사용자만 알아야 할 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

저장 데이터를 보호하는 것은 데이터 무결성, 기밀성 및 규제 요구 사항 준수를 유지하는 데 중요합니다. 액세스 제어, 격리, 조건부 액세스 및 버전 관리를 포함하여 이를 달성하는 데 도움이 되는 여러 제어를 구현할 수 있습니다.

최소 권한 원칙에 따라 액세스 제어를 적용할 수 있습니다. 이 원칙은 사용자와 서비스가 작업을 수행하는 데 필요한 권한만 제공합니다. 여기에는 암호화 키에 대한 액세스가 포함됩니다. AWS Key Management Service(AWS KMS) 정책을 검토하여 부여하는 액세스 수준이 적절하고 관련 조건이 적용되는지 확인합니다.

각 수준에 대해 고유한 AWS 계정을 사용하여 다양한 분류 수준에 따라 데이터를 분리하고 AWS Organizations을 사용하여 이러한 계정을 관리할 수 있습니다. 이러한 격리는 무단 액세스를 방지하고 데이터 노출 위험을 최소화하는 데 도움이 될 수 있습니다.

Amazon S3 버킷 정책에 부여된 액세스 수준을 정기적으로 검토합니다. 절대적으로 필요한 경우가 아니면 공개적으로 읽을 수 있거나 쓸 수 있는 버킷을 사용하지 마세요. 또한 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.

Amazon S3에 저장된 중요 데이터에 대한 버전 관리 및 객체 잠금 메커니즘을 구현합니다. Amazon S3 버전 관리는 이전 버전의 객체를 보존하여 우발적 삭제 또는 덮어쓰기를 했을 때 데이터를 복구합니다. 잠금이 만료될 때까지 루트 사용자라도 객체를 삭제하거나 덮어쓰지 못하게 하는 Amazon S3 Object Lock은 객체에 대한 필수 액세스 제어를 제공합니다. 또한 Amazon S3 Glacier Vault Lock은 Amazon S3 Glacier에 저장된 아카이브에 대해 유사한 기능을 제공합니다.

구현 단계

1. 최소 권한 원칙에 따라 액세스 제어 적용:
   - 사용자 및 서비스에 부여된 액세스 권한을 검토하고 작업을 수행하는 데 필요한 권한만 있는지 확인합니다.
   - AWS Key Management Service(AWS KMS) 정책을 확인하여 암호화 키에 대한 액세스를 검토합니다.

2. 다양한 분류 수준에 따라 데이터 분리:
   - 각 데이터 분류 수준에 대해 고유한 AWS 계정을 사용합니다.
   - AWS Organizations을 사용하여 이러한 계정을 관리합니다.
3. Amazon S3 버킷 및 객체 권한 검토:
   - Amazon S3 버킷 정책에 부여된 액세스 수준을 정기적으로 검토합니다.
   - 절대적으로 필요한 경우가 아니면 공개적으로 읽을 수 있거나 쓸 수 있는 버킷을 사용하지 마세요.
   - AWS Config를 사용하여 공개적으로 사용 가능한 버킷을 감지하는 것을 고려하세요.
   - Amazon CloudFront를 사용하여 Amazon S3의 콘텐츠를 제공합니다.
   - 퍼블릭 액세스를 허용하면 안 되는 버킷은 퍼블릭 액세스가 되지 않도록 적절히 구성되어 있는지 확인합니다.
   - SQS 또는 서드파티 데이터 스토어와 같이 IAM 인증을 사용하는 데이터베이스 및 기타 데이터 소스에 동일한 검토 프로세스를 적용할 수 있습니다.
4. AWS IAM Access Analyzer 사용:
   - Amazon S3 버킷을 분석하고 S3 정책이 외부 엔터티에 대한 액세스 권한을 부여할 때 조사 결과를 생성하도록 AWS IAM Access Analyzer를 구성할 수 있습니다.
5. 버전 관리 및 객체 잠금 메커니즘 구현:
   - Amazon S3 버전 관리를 사용하여 이전 버전의 객체를 보존하여 실수로 삭제하거나 덮어쓰는 것을 방지합니다.
   - 잠금이 만료될 때까지 루트 사용자라도 객체를 삭제하거나 덮어쓰지 못하게 하는 Amazon S3 Object Lock을 사용하여 객체에 대한 필수 액세스 제어를 제공합니다.
   - Amazon S3 Glacier에 저장된 아카이브에 Amazon S3 Glacier Vault Lock을 사용합니다.
6. Amazon S3 Inventory 사용:
   - Amazon S3 Inventory를 사용하여 S3 객체의 복제 및 암호화 상태를 감사하고 보고할 수 있습니다.
7. Amazon EBS 및 AMI 공유 권한 검토:
   - Amazon EBS 및 AMI 공유에 대한 권한을 검토하여 워크로드 외부에 존재하는 AWS 계정과 이미지 및 볼륨이 공유되지 않는다는 것을 확인합니다.
8. AWS Resource Access Manager Shares 정기적으로 검토:
   - AWS Resource Access Manager를 사용하여 Amazon VPC 내에서 AWS Network Firewall 정책, Amazon Route 53 Resolver 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 Cryptographic Details 백서
- Amazon S3 리소스에 대한 액세스 권한 관리 소개
- Overview of managing access to your AWS KMS resources
- AWS Config 규칙
- Amazon S3 + Amazon CloudFront: A Match Made in the Cloud
- 버전 관리 사용
- Amazon S3 객체 잠금을 사용하여 객체 잠금
- Sharing an Amazon EBS Snapshot
- 공유 AMI
- Hosting a single-page application on Amazon S3
- AWS 글로벌 조건 키
- Building a Data Perimeter on AWS

관련 비디오:

- Securing Your Block Storage on AWS

SEC 9. 전송 중 데이터는 어떻게 보호하나요?

여러 제어를 구현하여 무단 액세스 또는 손실의 위험을 줄여 전송 중인 데이터를 보호합니다.

모범 사례

- SEC09-BP01 보안 키 및 인증서 관리 구현
- SEC09-BP02 전송 중 암호화 적용

- SEC09-BP03 네트워크 통신 인증

SEC09-BP01 보안 키 및 인증서 관리 구현

전송 계층 보안(TLS) 인증서는 인터넷과 프라이빗 네트워크에서 네트워크 통신을 보호하고 웹 사이트, 리소스 및 워크로드의 ID를 설정하는 데 사용됩니다.

원하는 성과: 퍼블릭 키 인프라(PKI)에서 인증서를 프로비저닝, 배포, 저장 및 갱신할 수 있는 보안 인증서 관리 시스템. 보안 키 및 인증서 관리 메커니즘은 인증서 개인 키 구성 요소가 공개되는 것을 방지하고 정기적으로 인증서를 자동 갱신합니다. 또한 다른 서비스와 통합하여 워크로드 내부의 머신 리소스에 대한 보안 네트워크 통신 및 ID를 제공합니다. 키 구성 요소는 인적 자격 증명이 절대 접근할 수 없어야 합니다.

일반적인 안티 패턴:

- 인증서 배포 또는 갱신 프로세스 중에 수동 단계를 수행합니다.
- 프라이빗 CA를 설계할 때 인증 기관(CA) 계층 구조에 충분히 주의를 기울이지 않습니다.
- 퍼블릭 리소스에 자체 서명된 인증서를 사용합니다.

이 모범 사례 확립의 이점:

- 자동 배포 및 갱신을 통해 인증서 관리 간소화
- TLS 인증서를 사용하여 전송 중 데이터의 암호화 장려
- 인증 기관이 취한 인증서 작업의 보안 및 감사 가능성 향상
- CA 계층 구조의 여러 계층에서 관리 업무 구성

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

최신 워크로드는 TLS와 같은 PKI 프로토콜을 사용하는 암호화된 네트워크 통신을 광범위하게 사용합니다. PKI 인증서 관리는 복잡할 수 있지만 자동화된 인증서 프로비저닝, 배포 및 갱신을 통해 인증서 관리와 관련된 마찰을 줄일 수 있습니다.

AWS에서는 범용 PKI 인증서를 관리하기 위해 AWS Certificate Manager 및 AWS Private Certificate Authority(AWS Private CA)와 같은 두 가지 서비스를 제공합니다. ACM은 고객이 퍼블릭 워크로드와 프라이빗 AWS 워크로드 모두에서 사용할 인증서를 프로비저닝, 관리 및 배포하는 데 사용하는 기본

서비스입니다. ACM은 AWS Private CA를 사용하여 프라이빗 인증서를 발급하고 다른 많은 AWS관리형 서비스와 통합하여 워크로드에 보안 TLS 인증서를 제공합니다. ACM은 Amazon Trust Services 에서 공개적으로 신뢰할 수 있는 인증서를 발급할 수도 있습니다. ACM의 퍼블릭 인증서는 퍼블릭 워크로드에 사용할 수 있습니다. 최신 브라우저와 운영 체제는 기본적으로 이러한 인증서를 신뢰하기 때문입니다.

AWS Private CA를 사용하면 자체 루트의 CA 또는 하위 CA를 설정하고 API를 통해 TLS 인증서를 발급할 수 있습니다. 이러한 종류의 인증서는 TLS 연결의 클라이언트 측에서 신뢰 체인을 제어하고 관리하는 시나리오에서 사용할 수 있습니다. TLS 사용 사례 외에도 AWS Private CA를 사용하면 사용자 지정 템플릿으로 Kubernetes 포드, Matter 디바이스 제품 증명, 코드 서명 및 기타 사용 사례에 인증서를 발급할 수 있습니다. IAM Roles Anywhere를 사용하여 프라이빗 CA에서 서명한 X.509 인증서를 발급한 온프레미스 워크로드에 임시 IAM 자격 증명을 제공할 수 있습니다.

ACM 및 AWS Private CA 외에도 AWS IoT Core는 PKI 인증서를 IoT 디바이스에 프로비저닝, 관리 및 배포하기 위한 전문 지원을 제공합니다. AWS IoT Core는 대규모 퍼블릭 키 인프라에 적용할 수 있는 IoT 디바이스 온보딩용 전문 메커니즘을 제공합니다.

Amazon API Gateway 및 Elastic Load Balancing과 같은 일부 AWS 서비스는 인증서를 사용하여 애플리케이션 연결을 보호하는 자체 기능을 제공합니다. 예를 들어 API Gateway와 Application Load Balancer(ALB)는 모두 AWS Management Console, CLI 또는 API를 사용하여 생성하고 내보내는 클라이언트 인증서를 사용하여 상호 TLS(mTLS)를 지원합니다.

프라이빗 CA 계층 구조 설정 시 고려 사항

프라이빗 CA를 설정해야 하는 경우 CA 계층 구조를 미리 적절하게 설계할 수 있도록 특별히 주의를 기울이는 것이 중요합니다. 프라이빗 CA 계층 구조를 만들 때는 CA 계층 구조의 각 수준을 별도의 AWS 계정에 배포하는 것이 좋습니다. 이 의도적인 단계는 CA 계층 구조의 각 수준에 대한 노출 영역을 줄여 CloudTrail 로그 데이터에서 이상 징후를 더 쉽게 발견하고 계정 중 하나에 대한 무단 액세스가 발생할 경우 액세스 또는 영향 범위를 줄일 수 있습니다. 루트 CA는 별도의 계정에 있어야 하며 하나 이상의 중간 CA 인증서를 발급하는 데만 사용해야 합니다.

그런 다음 루트 CA 계정과 분리된 계정에 하나 이상의 중간 CA를 생성하여 최종 사용자, 디바이스 또는 기타 워크로드에 대한 인증서를 발급합니다. 마지막으로 루트 CA에서 중간 CA로 인증서를 발급합니다. 그러면 중간 CA가 최종 사용자나 디바이스에 인증서를 발급합니다. 복원력 계획, 크로스 리전 복제, 조직 전반의 CA 공유 등을 포함하여 CA 배포 계획 및 CA 계층 설계에 대한 자세한 내용은 Planning your AWS Private CA deployment를 참조하세요.

구현 단계

1. 사용 사례에 필요한 관련 AWS 서비스 확인:

- 많은 사용 사례에서 AWS Certificate Manager를 사용하여 기존 AWS 퍼블릭 키 인프라를 활용할 수 있습니다. ACM은 웹 서버나 로드 밸런서용 또는 공개적으로 신뢰할 수 있는 인증서를 위한 기타 용도로 TLS 인증서를 배포하는 데 사용할 수 있습니다.
- 자체 프라이빗 인증 기관 계층 구조를 설정해야 하거나 내보낼 수 있는 인증서에 액세스해야 하는 경우 AWS Private CA를 고려하세요. 그런 다음 ACM에서는 AWS Private CA를 사용하여 다양한 유형의 최종 엔터티 인증서를 발급할 수 있습니다.
- 내장된 사물 인터넷(IoT) 디바이스에 대규모로 인증서를 프로비저닝해야 하는 사용 사례의 경우에는 AWS IoT Core를 고려하세요.
- Amazon API Gateway 또는 Application Load Balancer와 같은 서비스에서 네이티브 mTLS 기능을 사용하는 것을 고려하세요.

2. 가능한 경우 자동 인증서 갱신 구현:
   - 통합된 AWS 관리형 서비스와 함께 ACM에서 발급한 인증서에 대해 ACM 관리형 갱신을 사용합니다.
3. 로깅 및 감사 트레일 설정:
   - CloudTrail 로그를 활성화하여 인증 기관이 있는 계정에 대한 액세스를 추적합니다. CloudTrail에서 로그 파일 무결성 검증을 구성하여 로그 데이터의 신뢰성을 확인하는 것이 좋습니다.
   - 프라이빗 CA가 발급 또는 취소한 인증서를 나열하는 감사 보고서를 정기적으로 생성하고 검토합니다. 이러한 보고서는 S3 버킷으로 내보낼 수 있습니다.
   - 프라이빗 CA를 배포할 때는 인증서 폐기 목록(CRL)을 저장할 S3 버킷도 설정해야 합니다. 워크로드 요구 사항에 따라 이 S3 버킷을 구성하는 방법에 대한 지침은 Planning a certificate revocation list (CRL)를 참조하세요.

리소스

관련 모범 사례:

- SEC02-BP02 임시 자격 증명 사용
- SEC08-BP01 보안 키 관리 구현
- SEC09-BP03 네트워크 통신 인증

관련 문서:

- How to host and manage an entire private certificate infrastructure in AWS
- How to secure an enterprise scale ACM Private CA hierarchy for automotive and manufacturing

- Private CA best practices
- How to use AWS RAM to share your ACM Private CA cross-account

관련 비디오:

- Activating AWS Certificate Manager Private CA(워크숍)

관련 예제:

- Private CA 워크숍
- IOT Device Management 워크숍(디바이스 프로비저닝 포함)

관련 도구:

- Plugin to Kubernetes cert-manager to use AWS Private CA

SEC09-BP02 전송 중 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 성과: 데이터에 대한 무단 액세스를 완화하기 위해 리소스와 인터넷 간의 네트워크 트래픽을 암호화합니다. 보안 요구 사항에 따라 내부 AWS 환경 내에서 네트워크 트래픽을 암호화합니다. 보안 TLS 프로토콜 및 암호 세트를 사용하여 전송 중 데이터를 암호화합니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS 서비스는 통신에 TLS를 사용하는 HTTPS 엔드포인트를 제공하여 AWS API와 통신할 때 전송 중 암호화 기능을 제공합니다. 안전하지 않은 HTTP 프로토콜은 보안 그룹을 사용하여 가상 프라이빗 클라우드(VPC)에서 감사 및 차단할 수 있습니다. HTTP 요청은 Amazon CloudFront의 HTTPS로나 Application Load Balancer에서 자동으로 리디렉션될 수도 있습니다. Amazon Simple Storage Service(Amazon S3) 버킷 정책을 사용하여 HTTP를 통해 객체를 업로드하는 기능을 제한하여 객체를 버킷에 업로드하는 데 HTTPS 사용을 효과적으로 적용할 수 있습니다. 컴퓨팅 리소스를 완전하게 제어하여 서비스 간에 전송 중 암호화를 구현할 수 있습니다. 외부 네트워크 또는 AWS Direct Connect로부터 특정 VPC로의 VPN 연결을 사용하여 트래픽을 쉽게 암호화할 수도 있습니다. AWS가 2024년 2월부터 이전 버전의 TLS 사용을 중단했으므로 클라이언트가 최소 TLS 1.2를 사용하여 AWS API를 직접 호출하는지 확인합니다. TLS 1.3을 사용할 것을 권장합니다. 전송 중 암호화에 대한 특별한 요구 사항이 있는 경우 AWS Marketplace에서 서드파티 솔루션을 찾을 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 프로파일을 사용합니다.
- 외부 연결을 위해 VPN 사용: 데이터 프라이버시와 무결성을 모두 지원할 수 있도록 지점 간 또는 네트워크 간 연결에 IPsec VPN 사용을 고려합니다.
- 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: ACM에서는 AWS 서비스에서 사용하도록 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:

- CloudFront에서 HTTPS 사용
- AWS Virtual Private Network를 사용하여 VPC를 원격 네트워크에 연결
- Create an HTTPS listener for your Application Load Balancer
- Tutorial: Configure SSL/TLS on Amazon Linux 2
- SSL/TLS를 사용하여 DB 인스턴스 연결 암호화
- 연결을 위한 보안 옵션 구성

SEC09-BP03 네트워크 통신 인증

전송 계층 보안(TLS) 또는 IPsec과 같은 인증을 지원하는 프로토콜을 사용하여 통신의 자격 증명을 확인합니다.

서비스, 애플리케이션 또는 사용자 간에 통신할 때마다 안전하고 인증된 네트워크 프로토콜을 사용하도록 워크로드를 설계합니다. 인증 및 권한 부여를 지원하는 네트워크 프로토콜을 사용하면 네트워크 흐름을 더 강력하게 제어할 수 있고 무단 액세스의 영향을 줄일 수 있습니다.

원하는 성과: 서비스 간 트래픽 흐름이 잘 정의된 데이터 영역 및 컨트롤 플레인 트래픽 흐름을 보유한 워크로드. 트래픽 흐름은 기술적으로 가능한 경우 인증되고 암호화된 네트워크 프로토콜을 사용합니다.

일반적인 안티 패턴:

- 암호화되지 않았거나 인증되지 않은 트래픽이 워크로드 내에 흐릅니다.
- 여러 사용자 또는 엔터티가 인증 자격 증명을 재사용합니다.
- 액세스 제어 메커니즘으로 네트워크 제어에만 의존합니다.
- 업계 표준 인증 메커니즘에 의존하지 않고 사용자 지정 인증 메커니즘을 구축합니다.
- 서비스 구성 요소 또는 VPC의 다른 리소스 간에 지나치게 허용적인 트래픽이 흐릅니다.

이 모범 사례 확립의 이점:

- 무단 액세스의 영향 범위를 워크로드의 한 부분으로 제한합니다.
- 작업이 인증된 엔터티에 의해서만 수행되도록 더 높은 수준의 보장을 제공합니다.

- 의도된 데이터 전송 인터페이스를 명확하게 정의하고 적용함으로써 서비스의 분리를 개선합니다.
- 요청 어트리뷰션 및 잘 정의된 통신 인터페이스를 통해 모니터링, 로깅 및 인시던트 대응을 개선합니다.
- 네트워크 제어와 인증 및 권한 제어를 결합하여 워크로드에 대한 심층 방어를 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

워크로드의 네트워크 트래픽 패턴은 두 가지 범주로 분류할 수 있습니다.

- 동서 트래픽은 워크로드를 구성하는 서비스 간의 트래픽 흐름을 나타냅니다.
- 남북 트래픽은 워크로드와 소비자 간의 트래픽 흐름을 나타냅니다.

남북 트래픽을 암호화하는 것이 일반적이지만 인증된 프로토콜을 사용하여 동서 트래픽을 보호하는 것은 흔하지 않습니다. 최신 보안 관행에서는 네트워크 설계만으로는 두 엔터티 간에 신뢰할 수 있는 관계를 부여하지 않을 것을 권장합니다. 두 서비스가 공통 네트워크 경계 내에 있을 수 있는 경우에도 이러한 서비스 간의 통신을 암호화 및 인증하고 권한을 부여하는 것이 가장 좋습니다.

예를 들어, AWS 서비스 API는 요청이 시작된 네트워크와 상관없이 AWS Signature Version 4(SigV4) 서명 프로토콜을 사용하여 호출자를 인증합니다. 이 인증을 통해 AWS API는 작업을 요청한 자격 증명을 확인할 수 있으며, 그런 다음 해당 자격 증명을 정책과 결합하여 권한 부여 결정을 내려 작업의 허용 여부를 결정할 수 있습니다.

Amazon VPC Lattice 및 Amazon API Gateway와 같은 서비스를 사용하면 동일한 SigV4 서명 프로토콜을 사용하여 자체 워크로드의 동서 트래픽에 인증 및 권한 부여를 추가할 수 있습니다. AWS 환경 외부의 리소스가 SigV4 기반 인증 및 권한 부여가 필요한 서비스와 통신해야 하는 경우 AWS 이외 리소스에서 AWS Identity and Access Management(IAM) Roles Anywhere를 사용하여 임시 AWS 자격 증명을 얻을 수 있습니다. 이러한 자격 증명을 사용하여 액세스 권한 부여에 SigV4를 사용하는 서비스에 대한 요청에 서명할 수 있습니다.

동서 트래픽을 인증하는 또 다른 일반적인 메커니즘은 TLS 상호 인증(mTLS)입니다. 많은 사물 인터넷(IoT), B2B 애플리케이션 및 마이크로서비스는 mTLS를 사용하여 클라이언트 및 서버 측 X.509 인증서를 모두 사용하여 TLS 통신 양측의 자격 증명을 확인합니다. 이러한 인증서는 AWS Private Certificate Authority(AWS Private CA)에서 발급할 수 있습니다. Amazon API Gateway와 같은 서비스를 사용하여 워크로드 간 또는 워크로드 내부 통신을 위한 mTLS 인증을 제공할 수 있습니다. Application Load Balancer는 내부 또는 외부 대상 워크로드에 대한 mTLS도 지원합니다. mTLS는 TLS 통신 양쪽에 대한 인증 정보를 제공하지만 권한 부여 메커니즘을 제공하지는 않습니다.

마지막으로 OAuth 2.0 및 OIDC(OpenID Connect)는 일반적으로 사용자의 서비스 액세스를 제어하는 데 사용되는 프로토콜이지만 이제는 서비스 간 트래픽에서도 널리 사용되고 있습니다. API Gateway는 JSON 웹 토큰(JWT) 권한 부여자를 제공하여 워크로드가 OIDC 또는 OAuth 2.0 ID 제공업체에서 발급한 JWT를 사용하여 API 경로에 대한 액세스를 제한할 수 있도록 합니다. OAuth2 범위는 기본 권한 부여 결정을 위한 소스로 사용될 수 있지만, 애플리케이션 계층에서 여전히 권한 부여 검사를 구현해야 하며, OAuth2 범위만으로는 더 복잡한 권한 부여 요구 사항을 지원할 수 없습니다.

구현 단계

- 워크로드 네트워크 흐름 정의 및 문서화: 심층 방어 전략을 구현하기 위한 첫 번째 단계는 워크로드의 트래픽 흐름을 정의하는 것입니다.
  - 워크로드를 구성하는 여러 서비스 간에 데이터가 전송되는 방식을 명확하게 정의하는 데이터 흐름도를 만듭니다. 이 다이어그램은 인증된 네트워크 채널을 통해 이러한 흐름을 적용하는 첫 번째 단계입니다.
  - 개발 및 테스트 단계에서 워크로드를 계측하여 데이터 흐름도가 런타임 시 워크로드의 동작을 정확하게 반영하는지 확인합니다.
  - 데이터 흐름도는 SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정에서 설명한 대로 위협 모델링 연습을 수행할 때도 유용할 수 있습니다.
- 네트워크 제어 설정: 데이터 흐름에 맞게 네트워크 제어를 설정할 수 있는 AWS 기능을 고려합니다. 네트워크 경계가 유일한 보안 제어가 되어서는 안 되지만, 네트워크 경계는 워크로드를 보호하기 위한 심층 방어 전략의 한 계층을 제공합니다.
  - 보안 그룹을 사용하여 리소스 간 데이터 흐름을 설정, 정의 및 제한합니다.
  - AWS 및 AWS PrivateLink를 지원하는 서드파티 서비스 모두와 통신하기 위해 AWS PrivateLink 사용을 고려하세요. AWS PrivateLink 인터페이스 엔드포인트를 통해 전송된 데이터는 AWS 네트워크 백본 내에 머물며 퍼블릭 인터넷을 통과하지 않습니다.
- 워크로드의 서비스 전반에 인증 및 권한 부여 구현: 워크로드에서 인증되고 암호화된 트래픽 흐름을 제공하는 데 가장 적합한 AWS 서비스 세트를 선택합니다.
  - 서비스 간 통신을 보호하려면 Amazon VPC Lattice를 고려하세요. VPC Lattice는 인증 정책과 결합된 SigV4 인증을 사용하여 서비스 간 액세스를 제어할 수 있습니다.
  - mTLS를 사용한 서비스 간 통신의 경우 API Gateway, Application Load Balancer를 고려해 보세요. AWS Private CA는 mTLS와 함께 사용할 인증서를 발급할 수 있는 프라이빗 CA 계층 구조를 설정하는 데 사용할 수 있습니다.
  - OAuth 2.0 또는 OIDC를 사용하여 서비스와 통합할 때는 JWT 권한 부여자를 사용하여 API Gateway를 사용하는 것을 고려하세요.

  - 워크로드와 IoT 디바이스 간 통신의 경우 네트워크 트래픽 암호화 및 인증을 위한 여러 옵션을 제공하는 AWS IoT Core를 고려해 보세요.
- 무단 액세스 모니터링: 의도하지 않은 통신 채널, 보호된 리소스에 대한 무단 액세스 시도 및 기타 부적절한 액세스 패턴을 지속적으로 모니터링합니다.
  - VPC Lattice를 사용하여 서비스에 대한 액세스를 관리하는 경우 VPC Lattice 액세스 로그를 활성화하고 모니터링하는 방법을 고려해 보세요. 이러한 액세스 로그에는 요청 엔티티에 대한 정보, 소스 및 대상 VPC를 비롯한 네트워크 정보, 요청 메타데이터가 포함됩니다.
  - VPC 흐름 로그를 사용하여 네트워크 흐름의 메타데이터를 캡처하고 주기적으로 이상 징후를 검토하는 방법을 고려해 보세요.
  - 보안 인시던트 계획, 시뮬레이션 및 대응에 대한 자세한 지침은 AWS Security Incident Response Guide 및 AWS Well-Architected Framework 보안 원칙의 인시던트 대응 섹션을 참조하세요.

리소스

관련 모범 사례:

- SEC03-BP07 퍼블릭 및 크로스 계정 액세스 분석
- SEC02-BP02 임시 자격 증명 사용
- SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정

관련 문서:

- Evaluating access control methods to secure Amazon API Gateway APIs
- REST API에 대한 상호 TLS 인증 구성
- How to secure API Gateway HTTP endpoints with JWT authorizer
- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- AWS Security Incident Response Guide

관련 비디오:

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

관련 예제:

- Amazon VPC Lattice 워크숍
- 제로 트러스트 에피소드 1 - The Phantom Service Perimeter 워크숍

인시던트 대응

질문

- SEC 10. 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구하나요?

SEC 10. 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구하나요?

예방 및 탐지 제어를 사용하더라도 조직은 잠재적 보안 인시던트에 대응하고 그 영향을 완화하기 위한 메커니즘을 구현해야 합니다. 이러한 준비는 인시던트 발생 시 보안팀이 효과적으로 문제를 격리 및 억제하고 문제에 대한 포렌식을 수행하고, 운영을 알려진 정상 상태로 복구하는 능력에 지대한 영향을 미칩니다. 보안 인시던트보다 앞서 도구 및 액세스를 마련하고 게임 데이를 통해 인시던트 대응을 정기적으로 연습한다면 비즈니스 중단을 최소화하면서 복구할 수 있습니다.

모범 사례

- SEC10-BP01 주요 직원과 외부 리소스 파악
- SEC10-BP02 인시던트 관리 계획 개발
- SEC10-BP03 포렌식 역량 확보
- SEC10-BP04 보안 인시던트 대응 플레이북 개발 및 테스트
- SEC10-BP05 액세스 권한 사전 프로비저닝
- SEC10-BP06 도구 사전 배포
- SEC10-BP07 시뮬레이션 실행
- SEC10-BP08 인시던트로부터 학습하기 위한 프레임워크 구축

SEC10-BP01 주요 직원과 외부 리소스 파악

조직이 인시던트에 대응하는 데 도움이 될 수 있는 내부 및 외부 직원, 리소스, 법적 의무를 파악합니다.

원하는 성과: 주요 담당자, 연락처 정보, 보안 이벤트 대응 시 수행하는 역할 목록이 있습니다. 이 정보를 정기적으로 검토하고 내부 및 외부 도구 관점에서 직원 변경 사항을 반영하도록 업데이트합니다. 이러한 정보를 문서화할 때는 보안 파트너, 클라우드 제공업체, 서비스형 소프트웨어(SaaS) 애플리케이션을 비롯한 모든 서드파티 서비스 제공업체 및 공급업체를 고려합니다. 보안 이벤트 중에는 적절한 수준의 책임을 맡고 상황 정보를 알고 있으며 액세스 권한을 가진 직원이 대응하고 복구할 수 있습니다.

일반적인 안티 패턴:

- 보안 이벤트 대응 시 연락처 정보, 역할, 담당 업무가 나와 있는 주요 인력의 최신 목록을 유지 관리하지 않습니다.
- 이벤트에 대응하고 이벤트에서 복구할 때 모두가 인력, 종속성, 인프라, 솔루션을 이해하고 있다고 가정합니다.
- 주요 인프라 또는 애플리케이션 설계를 나타내는 문서 또는 정보 리포지토리가 없습니다.
- 신입 직원이 보안 이벤트 대응에 효과적으로 기여할 수 있도록 돕는 적절한 온보딩 프로세스(예: 이벤트 시뮬레이션 수행)가 없습니다.
- 보안 이벤트 중에 주요 인력이 일시적으로 부재하거나 대응에 실패할 경우에 대비하여 에스컬레이션 경로가 마련되어 있지 않습니다.

이 모범 사례 확립의 이점: 이 방법을 사용하면 이벤트 중에 적합한 담당자와 역할을 식별하는 데 걸리는 분류 및 대응 시간이 단축됩니다. 주요 담당자 및 역할 목록을 업데이트하여 이벤트가 발생한 동안 낭비되는 시간을 최소화함으로써 적절한 인력 배치로 이벤트를 분류하고 복구할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

조직의 주요 담당자 파악: 참여해야 하는 조직 내 담당자의 연락처 목록을 유지 관리합니다. 운영상 변화, 승진, 팀 변경 등 인사이동이 발생하는 경우 관련 정보를 정기적으로 검토하고 업데이트하세요. 이는 인시던트 관리자, 인시던트 대응 담당자, 커뮤니케이션 책임자와 같은 주요 역할에 특히 중요합니다.

- 인시던트 관리자: 인시던트 관리자는 이벤트 대응 과정에서 전반적인 권한을 보유합니다.
- 인시던트 담당자: 인시던트 담당자는 조사 및 수정 활동을 담당합니다. 이러한 담당자는 이벤트 유형에 따라 다를 수 있지만. 보통 영향을 받는 애플리케이션을 다루는 개발자와 운영 팀이 역할을 맡습니다.
- 커뮤니케이션 책임자: 커뮤니케이션 책임자는 내부 및 외부 커뮤니케이션, 특히 공공 기관, 규제 기관 및 고객과의 커뮤니케이션을 담당합니다.
- 온보딩 프로세스: 인시던트 대응 노력에 효과적으로 기여하는 데 필요한 스킬과 지식을 갖추도록 신입 직원을 정기적으로 훈련하고 온보딩합니다. 온보딩 프로세스의 일부로 시뮬레이션 및 실습을 포함하여 준비를 촉진합니다.
- 분야별 전문가(SME): 분산되고 자율적인 팀의 경우 미션 크리티컬 워크로드를 처리할 SME를 파악하는 것이 좋습니다. SME는 이벤트와 관련된 중요 워크로드의 운영 및 데이터 분류에 대한 인사이트를 제공합니다.

예시 테이블 형식:

```
  | Role | Name | Contact Information | Responsibilities |
1 | —- | —- | —- | —- |
2 | Incident Manager | Jane Doe| jane.doe@example.com | Overall authority during
 response |
3 | Incident Responder | John Smith | john.smith@example.com | Investigation and
 remediation |
4 | Communications Lead | Emily Johnson | emily.johnson@example.com | Internal and
 external communications |
5 | Communications Lead | Michael Brown | michael.brown@example.com | Insights on
 critical workloads |
```

AWS Systems Manager Incident Manager 기능을 사용하여 주요 담당자를 포착하고, 대응 계획을 정의하며, 당직 일정을 자동화하고, 에스컬레이션 계획을 생성하는 것을 고려해 보세요. 당직 일정에 따라 모든 직원을 자동화하고 교체하여 워크로드에 대한 책임을 여러 담당자가 분담하도록 합니다. 이를 통해 관련 지표 및 로그를 내보내고 워크로드에 중요한 경보 임곗값을 정의하는 등의 모범 사례를 활용할 수 있습니다.

외부 파트너 식별: 기업은 독립 소프트웨어 개발 판매 회사(ISV), 파트너 및 하청업체가 구축한 도구를 사용하여 고객을 위한 차별화 솔루션을 구축합니다. 인시던트에 대응하고 인시던트로부터 복구하는 데 도움을 줄 수 있는 주요 담당자를 참여시키세요. 지원 사례를 통해 AWS 분야별 전문가의 도움을 빠르게 받으려면 적절한 수준의 지원에 가입하는 것이 좋습니다. 워크로드에 대해 모든 주요 솔루션 제공업체와 유사한 계약을 체결하는 것을 고려하세요. 일부 보안 이벤트의 경우 상장 기업은 관련 공공 기관 및 규제 기관에 이벤트와 영향을 알려야 합니다. 관련 부서 및 담당자의 연락처 정보를 유지 및 업데이트하세요.

구현 단계

1. 인시던트 관리 솔루션을 설정합니다.
   a. 보안 도구 계정에 Incident Manager를 배포하는 것을 고려해 보세요.
2. 인시던트 관리 솔루션에서 담당자를 정의합니다.
   a. 인시던트 발생 시 연락이 안 되는 일이 없도록 각 담당자에 대해 최소 2가지 유형의 연락 채널(예: SMS, 전화 또는 이메일)을 정의합니다.
3. 대응 계획을 정의합니다.
   a. 인시던트 발생 시 가장 적절한 담당자를 식별합니다. 개별 담당자가 아닌 참여 대상 직원의 역할에 맞게 에스컬레이션 계획을 정의합니다. 외부 주체가 인시던트 해결에 직접 관여하지 않았더라도 외부 기관에 알릴 책임이 있는 담당자를 포함시키는 것이 좋습니다.

리소스

관련 모범 사례:

- OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별

관련 문서:

- AWS Security Incident Response Guide

관련 예제:

- AWS customer playbook framework
- Prepare for and respond to security incidents in your AWS environment

관련 도구:

- AWS Systems Manager Incident Manager

관련 비디오:

- Amazon's approach to security during development

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 않을 경우 노출되는 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 Guide의 권장 사항을 따르세요. 마찬가지로 개인 식별 정보(PII)를 저장하는 워크로드를 운영할 때는 데이터 레지던시 및 사용과 관련된 문제를 보호하고 대응하는 방법을 고려합니다.

AWS에서 워크로드에 대한 인시던트 관리 계획을 구축하는 경우, 인시던트 대응에 대한 심층 방어 방식을 구축하기 위해 AWS 공동 책임 모델부터 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. AWS Security Incident Response Guide에서는 클라우드 중심 인시던트 관리 계획을 구축하기 위한 주요 개념과 기본적인 지침을 자세히 설명합니다.

효과적인 인시던트 관리 계획은 클라우드 운영 목표와 함께 끊임없이 반복되고 항상 최신 상태를 유지해야 합니다. 인시던트 관리 계획을 수립 및 개선할 때 아래에서 자세히 설명하는 구현 계획의 사용을 고려해 볼 수 있습니다.

구현 단계

1. 조직 내에서 보안 이벤트를 처리하기 위한 역할과 책임을 정의합니다. 여기에는 다음을 포함한 다양한 부서의 담당자가 참여해야 합니다.
   - 인적 자원(HR)
   - 경영진
   - 법무 부서
   - 애플리케이션 소유자 및 개발자(주제 전문가 또는 SME)
2. 인시던트 발생 시 업무에 대한 책임을 지는 사람, 결과에 대한 책임을 지는 사람, 상의해야 할 사람, 정보를 제공해야 할 사람(RACI)을 명확하게 설명합니다. RACI 차트를 생성하여 빠르고 직접적인 커뮤니케이션을 촉진하고 이벤트의 다양한 단계에서 리더십을 명확하게 설명합니다.
3. 인시던트 중에 애플리케이션 소유자와 개발자(SME)가 영향을 측정하는 데 도움이 되는 귀중한 정보와 컨텍스트를 제공할 수 있으므로 참여시킵니다. 이러한 SME와 관계를 구축하고 실제 인시던트가 발생하기 전에 해당 SME와 인시던트 대응 시나리오를 연습합니다.

4. 신뢰할 수 있는 파트너 또는 외부 전문가가 추가적인 전문성과 관점을 제공할 수 있으므로 조사 또는 대응 프로세스에 참여시킵니다.
5. 인시던트 관리 계획 및 역할을 조직에 적용되는 모든 현지 규정 또는 규정 준수 요구 사항에 맞게 조정합니다.
6. 인시던트 대응 계획을 정기적으로 연습 및 테스트하고 정의된 모든 역할과 책임을 포함합니다. 이렇게 하면 프로세스를 간소화하고 보안 인시던트에 대한 조정되고 효율적인 대응 조치가 있는지 확인할 수 있습니다.
7. 역할, 책임 및 RACI 차트를 정기적으로 또는 조직 구조 또는 요구 사항이 변경될 때 검토 및 업데이트합니다.

AWS 대응 팀 및 지원 이해

- AWS Support
  - 지원는 다양한 플랜을 제공합니다. 이러한 플랜을 통해 AWS 솔루션의 성공과 운영 상태를 지원하는 도구 및 전문 지식에 액세스할 수 있습니다. AWS 환경을 계획, 배포, 최적화하는 데 도움이 되는 기술 지원 및 추가 리소스가 필요한 경우 AWS 사용 사례에 가장 적합한 지원 플랜을 선택할 수 있습니다.
  - AWS Management Console의 지원 센터(로그인이 필요함)를 AWS 리소스에 영향을 미치는 문제에 대한 지원을 받을 수 있는 중앙 연락 창구로 고려하세요. 지원에 대한 액세스는 AWS Identity and Access Management로 제어됩니다. 지원 기능에 액세스하는 방법에 대한 자세한 내용은 Getting started with 지원를 참조하세요.
- AWS 고객 인시던트 대응 팀(CIRT)
  - AWS 고객 인시던트 대응 팀(CIRT)은 24/7로 운영되는 전문 글로벌 AWS 팀으로, AWS 공동 책임 모델의 고객 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 지원 사례를 통해 AWS CIRT의 지원을 요청할 수 있습니다.
- DDoS 대응 지원
  - AWS에서는 AWS Shield를 제공합니다. 이를 통해 AWS에서 실행 중인 웹 애플리케이션을 보호하는 관리형 분산 서비스 거부(DDoS) 보호 서비스를 제공합니다. Shield는 애플리케이션 가동 중지 시간과 대기 시간을 최소화하는 상시 탐지 및 자동 인라인 완화를 제공하므로 지원를 이용하지 않고도 DDoS 보호를 활용할 수 있습니다. Shield에는 AWS Shield Standard 및 AWS Shield

Advanced와 같은 두 가지 티어가 있습니다. 이 두 티어의 차이점에 대해 알아보려면 Shield 기능 설명서를 참조하세요.

- AWS Managed Services(AMS)
  - AWS Managed Services(AMS)에서는 AWS 인프라를 지속적으로 관리하므로 사용자는 애플리케이션에 집중할 수 있습니다. 인프라를 유지 관리하기 위한 모범 사례를 구현함으로써 AMS는 운영 오버헤드와 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 목표와 기능을 간략하게 설명합니다.
- 역할 및 책임: 인시던트 대응 이해관계자를 나열하고 인시던트 발생 시 해당 이해관계자의 역할을 자세히 설명합니다.
- 커뮤니케이션 계획: 연락처 정보 및 인시던트 발생 시 커뮤니케이션 방법을 자세히 설명합니다.
- 커뮤니케이션 방법 백업: 인시던트 커뮤니케이션의 백업으로 대역 외 통신을 사용하는 것이 모범 사례입니다. 안전한 대역 외 통신 채널을 제공하는 애플리케이션의 예는 AWS Wickr입니다.
- 인시던트 대응 단계 및 수행할 조치: 인시던트 대응의 단계(예: 탐지, 분석, 제거, 억제, 복구)를 열거합니다. 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 Detection

관련 문서:

- AWS Security Incident Response Guide
- NIST: Computer Security Incident Handling Guide

SEC10-BP03 포렌식 역량 확보

보안 인시던트에 앞서 보안 이벤트 조사를 지원하기 위한 포렌식 기능을 개발하는 것이 좋습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

기존 온프레미스 포렌식의 개념이 AWS에 적용됩니다. AWS 클라우드에서 포렌식 역량 구축을 시작하는 데 필요한 주요 정보는 Forensic investigation environment strategies in the AWS 클라우드를 참조하세요.

포렌식을 위한 환경과 AWS 계정 계정 구조를 설정한 후에는 네 단계에 걸쳐 포렌식 방법론을 효과적으로 수행하는 데 필요한 기술을 정의하세요.

- 수집: AWS CloudTrail, AWS Config, VPC 흐름 로그, 호스트 수준 로그 등 관련 AWS 로그를 수집합니다. 가능한 경우 영향을 받은 AWS 리소스의 스냅샷, 백업, 메모리 덤프를 수집합니다.
- 검사: 관련 정보를 추출하고 평가하여 수집한 데이터를 검사합니다.
- 분석: 수집된 데이터를 분석하여 인시던트를 이해하고 결론을 도출합니다.
- 보고: 분석 단계의 결과 정보를 제시합니다.

구현 단계

포렌식 환경 준비

AWS Organizations는 AWS 리소스의 성장과 규모 조정에 따라 AWS 환경을 중앙에서 관리하고 제어할 수 있도록 도와줍니다. AWS 조직은 AWS 계정을 통합하여 단일 단위로 관리할 수 있도록 합니다. 조직 단위(OU)를 사용하면 계정을 그룹으로 만들어 단일 유닛으로 관리할 수 있습니다.

인시던트 대응에는 보안 OU 및 포렌식 OU를 포함하는 인시던트 대응 기능을 지원하는 AWS 계정 구조를 갖는 것이 도움이 됩니다. 보안 OU 내에는 다음에 대한 계정이 있어야 합니다.

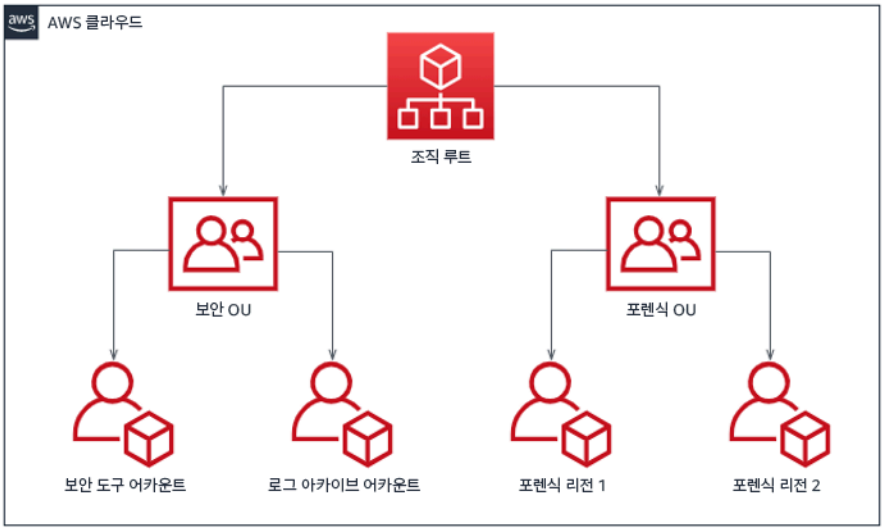
- 로그 보관: 제한된 권한으로 로그 아카이브 AWS 계정에서 로그를 집계합니다.

- 보안 도구: 보안 도구 AWS 계정에서 보안 서비스를 중앙 집중화합니다. 이 계정은 보안 서비스에 대한 위임된 관리자 역할을 합니다.

포렌식 OU 내에서, 비즈니스와 운영 모델에 가장 적합한 것에 따라 운영하는 각 리전에 대해 단일 포렌식 계정 또는 여러 개의 계정을 구현할 수 있는 옵션이 있습니다. 리전별로 포렌식 계정을 생성하면 해당 리전 외부에서 AWS 리소스를 생성하는 것을 차단하고 리소스가 의도하지 않은 리전으로 복사되는 위험을 줄일 수 있습니다. 예를 들어, 미국 동부(버지니아 북부) 리전(`us-east-1`) 및 미국 서부(오레곤)(`us-west-2`)에서만 비즈니스를 운영하는 경우 포렌식 OU에는 두 개의 계정(`us-east-1`에 대한 계정과 `us-west-2`에 대한 계정)이 있습니다.

여러 리전에 대한 포렌식 AWS 계정 계정을 만들 수 있습니다. 데이터 주권 요구 사항을 준수하고 있는지 확인하기 위해 AWS 리소스를 해당 계정으로 복사할 때는 주의해야 합니다. 새 계정을 프로비저닝하는 데는 시간이 걸리므로, 사고 발생 훨씬 전에 포렌식 계정을 생성하고 계측하여 대응 담당자가 대응에 효과적으로 사용할 수 있도록 준비하는 것이 필수적입니다.

다음 다이어그램은 리전별 포렌식 계정이 있는 포렌식 OU를 포함한 샘플 계정 구조를 보여줍니다.



인시던트 대응을 위한 리전별 계정 구조

백업 및 스냅샷 캡처

주요 시스템과 데이터베이스의 백업을 설정하는 것은 보안 인시던트 복구 및 포렌식 용도로 매우 중요합니다. 백업을 설정하면 시스템을 이전의 안전한 상태로 복원할 수 있습니다. AWS에서는 다양한 리소스의 스냅샷을 생성할 수 있습니다. 스냅샷은 해당 리소스의 특정 시점 백업을 제공합니다. 백업 및 복구를 지원할 수 있는 많은 AWS 서비스가 있습니다. 백업 및 복구를 위한 이러한 서비스와 접근 방식

에 대한 자세한 내용은 Backup and Recovery Prescriptive Guidance 및 Use backups to recover from security incidents를 참조하세요.

특히 랜섬웨어와 같은 상황에서는 백업의 보안을 잘 유지하는 것이 중요합니다. 백업 보안을 위한 지침은 Top 10 security best practices for securing backups in AWS를 참조하세요. 백업의 보안을 유지하는 것 외에도 정기적으로 백업 및 복원 프로세스를 테스트하여 보유한 기술과 프로세스가 정상적으로 작동하는지 확인해야 합니다.

포렌식 자동화

보안 이벤트가 발생하는 동안 인시던트 대응팀은 이벤트와 관련된 기간에 정확성을 유지하면서 증거를 신속하게 수집하고 분석할 수 있어야 합니다(예: 특정 이벤트 또는 리소스와 관련된 로그 캡처 또는 Amazon EC2 인스턴스의 메모리 덤프 수집). 특히 많은 인스턴스와 계정에서 관련 증거를 수동으로 수집하는 작업은 인시던트 대응팀에게 어렵고 시간이 많이 소요되는 업무입니다. 또한 수작업으로 수집하는 경우 인적 오류가 발생하기 쉽습니다. 이러한 이유로 포렌식을 위한 자동화를 최대한 개발하고 구현해야 합니다.

AWS는 포렌식을 위한 여러 가지 자동화 리소스를 제공하며 이는 아래 리소스 섹션에 열거되어 있습니다. 다음은 저희가 개발하고 고객이 구현한 포렌식 패턴의 리소스 예제입니다. 시작하기에 유용한 참조 아키텍처가 될 수 있지만, 환경, 요구 사항, 도구, 포렌식 프로세스에 따라 이를 수정하거나 새로운 포렌식 자동화 패턴을 생성하는 것을 고려하세요.

리소스

관련 문서:

- AWS Security Incident Response Guide - Develop Forensics Capabilities
- AWS Security Incident Response Guide - Forensics Resources
- Forensic investigation environment strategies in the AWS 클라우드
- How to automate forensic disk collection in AWS
- AWS Prescriptive Guidance - Automate incident response and forensics

관련 비디오:

- Automating Incident Response and Forensics

관련 예제:

- Automated Incident Response and Forensics Framework

- Automated Forensics Orchestrator for Amazon EC2

SEC10-BP04 보안 인시던트 대응 플레이북 개발 및 테스트

인시던트 대응 프로세스를 준비하는 데 있어 가장 중요한 부분은 플레이북을 개발하는 것입니다. 인시던트 대응 플레이북은 보안 이벤트가 발생했을 때 따라야 할 권장 가이드와 단계를 제공합니다. 명확한 구조와 단계를 갖추면 대응 프로세스가 간소화되고 인적 오류의 가능성이 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

다음과 같은 인시던트 시나리오에 대한 플레이북을 만들어야 합니다.

- 예상되는 인시던트: 예상되는 인시던트에 대한 플레이북을 만들어야 합니다. 여기에는 서비스 거부(DoS), 랜섬웨어, 자격 증명 유출과 같은 위협이 포함됩니다.
- 알려진 보안 조사 결과 또는 알림: 알려진 보안 조사 결과 및 알림(예: Amazon GuardDuty 조사 결과 및 알림)을 해결하기 위한 플레이북을 만들어야 합니다. GuardDuty 조사 결과를 받으면 플레이북은 알림을 잘못 처리하거나 무시하지 않도록 명확한 단계를 제공해야 합니다. 자세한 문제 해결 세부 정보 및 지침은 감지된 GuardDuty 보안 결과 해결을 참조하세요.

플레이북에는 보안 분석가가 잠재적인 보안 사고를 적절히 조사하고 대응하기 위해 완료해야 할 기술 단계가 포함되어야 합니다.

구현 단계

플레이북에 포함할 항목은 다음과 같습니다.

- 플레이북 개요: 이 플레이북은 어떤 위험 또는 인시던트 시나리오를 다루고 있나요? 플레이북의 목표는 무엇인가요?
- 사전 조건: 이 인시던트 시나리오에 어떤 로그, 탐지 메커니즘 및 자동화된 도구가 필요한가요? 예상되는 알림은 무엇인가요?
- 커뮤니케이션 및 에스컬레이션 정보: 누가 관여하며 담당자의 연락처 정보는 무엇인가요? 관련된 각 이해관계자의 책임은 무엇인가요?
- 대응 단계: 인시던트 대응 단계 전반에서 어떤 전술적 단계를 수행해야 하나요? 분석가는 어떤 쿼리를 실행해야 하나요? 원하는 성과를 얻으려면 어떤 코드를 실행해야 하나요?
  - 감지: 어떻게 인시던트를 감지하나요?
  - 분석: 어떻게 영향 범위를 결정하나요?

  - 포함: 범위를 제한하기 위해 어떻게 인시던트를 격리하나요?
  - 근절: 환경에서 위협을 어떻게 제거하나요?
  - 복구: 영향을 받은 시스템이나 리소스를 어떻게 프로덕션 환경으로 복구하나요?
- 예상 결과: 쿼리와 코드가 실행된 후 플레이북의 예상 결과는 무엇인가요?

리소스

관련 Well-Architected 모범 사례:

- SEC10-BP02 인시던트 관리 계획 개발

관련 문서:

- Framework for Incident Response Playbooks
- Develop your own Incident Response Playbooks
- Incident Response Playbook Samples
- Building an AWS incident response runbook using Jupyter playbooks and CloudTrail Lake

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 계정을 변경합니다.
- 적시 권한 승격을 제공할 때 IAM 권한을 직접 조작합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

AWS는 가능한 경우 장기 자격 증명에 대한 의존도를 줄이거나 제거할 것을 권장합니다. 그 대신, 임시 자격 증명 및 적시 권한 에스컬레이션 메커니즘을 사용합니다. 장기 자격 증명은 보안 위험에 노출되기 쉽고, 운영 오버헤드가 증가합니다. 대부분의 관리 작업과 인시던트 대응 작업의 경우 관리 액세스를

위한 임시 에스컬레이션과 함께 ID 페더레이션을 구현하는 것이 좋습니다. 이 모델에서는 사용자가 더 높은 수준의 권한(인시던트 대응 역할 등)을 요청하고, 사용자가 권한 승격에 적합한 경우 요청이 승인자에게 전송됩니다. 요청이 승인되면 사용자는 작업을 완료하는 데 사용할 수 있는 임시 AWS 자격 증명 세트를 받습니다. 이러한 자격 증명이 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 정책을 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.
- 분산 서비스 거부(DDoS) 이벤트 배포 또는 시스템을 사용할 수 없도록 렌더링하는 등의 악의적인 활동.

위의 사례에서는 긴급 break glass 액세스를 구성하여 인시던트에 대한 조사 및 적시 수정이 이루어지도록 해야 합니다. 적절한 권한이 있는 사용자, 그룹 또는 역할을 사용하여 작업을 수행하고 AWS 리소스에 액세스하는 것이 좋습니다. 루트 사용자 자격 증명이 필요한 작업에서만 루트 사용자를 사용합니다. 인시던트 응답자가 AWS 및 기타 관련 시스템에 대한 올바른 수준의 액세스 권한이 있는지 확인할 수 있도록, 전용 계정을 사전 프로비저닝하는 것이 좋습니다. 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 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)를 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 통해 구성할 수 있습니다(AWS 보안 보범 사례의 AWS Organizations SCP 참조). 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 개인 간에 공유되지 않도록 하고, 특정 작업에 필요하지 않는 한 AWS 계정 루트 사용자가 사용되지 않도록 합니다. 루트 사용자가 필요한 경우(예를 들어, 특정 계정에 대한 IAM 액세스를 사용할 수 없는 경우), 사용 가능한 플레이북을 통해 별도의 프로세스를 사용하여 루트 사용자 자격 증명 및 MFA 토큰의 가용성을 확인해야 합니다.

인시던트 대응 역할에 대한 IAM 정책을 구성하려면 IAM Access Analyzer를 사용하여 AWS CloudTrail 로그를 기반으로 정책을 생성하는 방법을 고려하세요. 이를 위해서는 비프로덕션 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 계정을 사용하여 다른 AWS 계정에서 전용 인시던트 대응 IAM 역할을 수임합니다. 이러한 역할은 보안 계정의 사용자만 수임할 수 있도록 구성되어야 하며, 신뢰 관계에서는 직접 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 계정과 IAM 역할의 이름을 모두 명확하게 지정하여 CloudTrail 로그에서 쉽게 찾을 수 있도록 하는 것이 좋습니다. 그 예로는 IAM 계정은 *<USER_ID>*`-BREAK-GLASS`, IAM 역할은 `BREAK-GLASS-ROLE`로 이름을 지정합니다.

CloudTrail은 AWS 계정의 API 활동을 로깅하는 데 사용되며 인시던트 대응 역할의 사용에 대한 알림을 구성하는 데 사용해야 합니다. 루트 키 사용 시 알림 구성에 대한 블로그 게시물을 참조하세요. 인시던트 대응 IAM 역할과 관련된 `AssumeRole` 이벤트를 필터링하기 위해 Amazon CloudWatch 지표 필터를 구성하도록 지침을 수정할 수 있습니다.

```
{ $.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 포함될 수 있습니다. SSH 또는 RDP와 같은 기본 프로토콜을 사용하는 대신 Amazon EC2 인스턴스에 대한 모든 관리 액세스에 AWS Systems Manager Session Manager를 사용하는 것이 좋습니다. 이 액세스는 보안 및 감사 기능이 있는 IAM을 사용하여 제어할 수 있습니다. AWS Systems Manager Run Command 문서를 사용하여 플레이북의 일부를 자동화함으로써 사용자 오류를 줄이고 복구 시간을 단축할 수도 있습니다. 데이터베이스 및 서드파티 도구에 대한 액세스를 위해, AWS Secrets Manager에 액세스 자격 증명을 저장하고 인시던트 응답자 역할에 액세스 권한을 부여하는 것이 좋습니다.

마지막으로, 인시던트 대응 IAM 계정의 관리를 입사, 전근 및 퇴사 프로세스에 추가하고 주기적으로 검토 및 테스트하여 의도한 액세스만 허용되는지 확인해야 합니다.

리소스

관련 문서:

- Managing temporary elevated access to your AWS environment
- AWS Security Incident Response Guide
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- IAM 사용자의 계정 암호 정책 설정
- AWS의 다중 인증(MFA) 사용
- Configuring Cross-Account Access with MFA
- Using IAM Access Analyzer to generate IAM policies
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- How to Receive Notifications When Your AWS Account’s Root Access Keys Are Used
- Create fine-grained session permissions using IAM managed policies
- Break glass access

관련 비디오:

- Automating Incident Response and Forensics in AWS

- DIY guide to runbooks, incident reports, and incident response
- Prepare for and respond to security incidents in your AWS environment

SEC10-BP06 도구 사전 배포

보안 담당자가 조사부터 복구까지 소요되는 시간을 단축할 수 있는 올바른 도구를 미리 배포했는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

보안 대응 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.

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 임곗값 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.

보안 조사 중에 관련 로그를 검토하여 인시던트의 전체 범위와 타임라인을 기록하고 이해할 수 있어야 합니다. 관심 있는 특정 작업이 발생했음을 나타내는 알림 생성에도 로그가 필요합니다. 쿼리 및 검색 메커니즘을 선택, 활성화, 저장 및 설정하고 경보를 설정하는 것이 중요합니다. 또한 로그 데이터를 검색할 수 있는 도구를 제공하는 효과적인 방법은 Amazon Detective입니다.

AWS는 200개 이상의 클라우드 서비스와 수천 개의 기능을 제공합니다. 인시던트 대응 전략을 지원하고 간소화할 수 있는 서비스를 검토하는 것이 좋습니다.

로깅 외에도 태그 지정 전략을 개발하고 구현해야 합니다. 태그 지정은 AWS 리소스의 목적에 대한 컨텍스트를 제공하는 데 도움이 될 수 있습니다. 태그 지정은 자동화를 위해서도 사용할 수 있습니다.

구현 단계

분석 및 알림을 위한 로그 선택 및 설정

인시던트 대응을 위한 로깅 구성에 대한 다음 설명서를 참조하세요.

- 보안 인시던트 분석을 위한 로깅 전략
- SEC04-BP01 서비스 및 애플리케이션 로깅 구성

보안 서비스를 사용하여 탐지 및 대응 지원

AWS는 기본 탐지, 예방 및 대응 기능을 제공하며, 기타 서비스를 사용하여 맞춤형 보안 솔루션을 설계할 수 있습니다. 보안 인시던트 대응과 가장 관련성이 높은 서비스 목록은 클라우드 기능 정의를 참조하세요.

태그 지정 전략 개발 및 구현

AWS 리소스를 사용하는 비즈니스 사용 사례와 관련 내부 이해관계자에 대한 컨텍스트 정보를 얻는 것은 어려울 수 있습니다. 한 가지 방법은 AWS 리소스에 메타데이터를 할당하고 사용자 정의 키와 값으로 구성되는 태그의 형태를 사용하는 것입니다. 태그를 생성하여 리소스를 목적, 소유자, 환경, 처리되는 데이터 유형 및 기타 원하는 기준에 따라 분류할 수 있습니다.

일관된 태그 전략을 사용하면 AWS 리소스에 대한 컨텍스트 정보를 신속하게 식별할 수 있으므로 응답 시간을 단축하고 조직 컨텍스트에 소요되는 시간을 최소화할 수 있습니다. 태그는 응답 자동화를 시작하는 메커니즘으로도 사용할 수 있습니다. 태그 지정 대상에 대한 자세한 내용은 Tagging your AWS resources를 참조하세요. 먼저 조직 전체에 구현할 태그를 정의하는 것이 좋습니다. 그런 다음 이 태그 지정 전략을 구현하고 적용합니다. 구현 및 시행에 대한 자세한 내용은 Implement AWS resource tagging strategy using AWS Tag Policies and Service Control Policies (SCPs)를 참조하세요.

리소스

관련 Well-Architected 모범 사례:

- SEC04-BP01 서비스 및 애플리케이션 로깅 구성
- SEC04-BP02 표준화된 위치에서 로그, 조사 결과 및 지표 캡처

관련 문서:

- 보안 인시던트 분석을 위한 로깅 전략
- 인시던트 대응 클라우드 기능 정의

관련 예제:

- Threat Detection and Response with Amazon GuardDuty and Amazon Detective
- Security Hub 워크숍
- Vulnerability Management with Amazon Inspector

SEC10-BP07 시뮬레이션 실행

시간이 지나면서 조직이 성장하고 발전함에 따라 위협 환경도 변화하므로 인시던트 대응 능력을 지속적으로 검토하는 것이 중요합니다. 시뮬레이션(게임 데이라고도 함)을 실행하는 것도 이 평가를 수행하는 데 사용할 수 있는 방법 중 하나입니다. 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: 시뮬레이션은 다음과 같은 다양한 이점을 제공합니다.

- 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 않을 경우 노출되는 위험 수준: 중간

구현 가이드

시뮬레이션에는 다음과 같은 세 가지 주요 유형이 있습니다.

- 탁상 연습: 시뮬레이션에 대한 탁상 접근 방식은 다양한 인시던트 대응 이해관계자가 참여하여 책임진 역할을 연습하고 확립된 커뮤니케이션 도구과 플레이북을 사용하는 토론 기반 세션입니다. 연습은 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 이해관계자들도 포함될 수 있습니다. 레드 팀은 보안 공격 교육을 받은 침투 테스트 팀 또는 주요 이해관계자로 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 성과에 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 이해관계자 식별: 연습에는 최소한 연습 진행자와 참가자가 필요합니다. 시나리오에 따라 법무, 커뮤니케이션 또는 경영진과 같은 추가 이해관계자가 참여할 수 있습니다.
3. 시니라오 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) 개발: 잘 운영된 영역, 개선이 필요한 영역, 잠재적인 격차를 식별합니다. 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 가이드

관련 비디오:

- AWS GameDay - Security Edition
- Running effective security incident response simulations

SEC10-BP08 인시던트로부터 학습하기 위한 프레임워크 구축

학습한 교훈 프레임워크와 근본 원인 분석 기능을 구현하면 인시던트 대응 능력을 개선하는 데 도움이 될 뿐만 아니라 인시던트 재발을 방지하는 데도 도움이 됩니다. 각 인시던트에서 교훈을 얻음으로써 동일한 실수, 노출 또는 잘못된 구성을 반복하지 않도록 하여 보안 태세를 개선할 뿐만 아니라 사전에 방지 가능한 상황으로 인한 시간 손실을 최소화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

다음 사항을 높은 수준에서 설정하고 달성하는 학습한 교훈 프레임워크를 구현하는 것이 중요합니다.

- 학습한 교훈은 언제 적용하게 되나요?
- 학습한 교훈 과정에는 무엇이 포함되나요?
- 학습한 교훈은 어떻게 수행되나요?
- 누가 어떻게 이 과정에 참여하나요?
- 개선이 필요한 부분은 어떻게 확인할 수 있나요?
- 개선 사항을 효과적으로 추적하고 구현할 수 있도록 어떻게 해야 할까요?

프레임워크는 개인에게 초점을 맞추거나 개인을 비난하는 것이 아니라 도구와 프로세스를 개선하는 데 초점을 맞춰야 합니다.

구현 단계

앞에서 설명한 개략적인 결과 외에도, 프로세스에서 최대한의 가치(실행 가능한 개선으로 이어지는 정보)를 이끌어낼 수 있도록 올바른 질문을 하는 것이 중요합니다. 다음 질문을 고려하면 학습한 교훈 토론을 시작하는 데 도움이 됩니다.

- 어떤 인시던트였나요?
- 인시던트가 언제 처음 확인되었나요?
- 어떻게 식별되었나요?
- 어떤 시스템에서 해당 활동에 대해 경고했나요?
- 어떤 시스템, 서비스 및 데이터가 관련되어 있나요?
- 구체적으로 어떤 일이 발생했나요?
- 어떤 점이 잘 작동했나요?
- 어떤 점이 잘 작동하지 않았나요?
- 인시던트에 대응하기 위해 어떤 프로세스 또는 절차가 실패했거나 조정되지 못했나요?
- 다음 영역에서 개선할 수 있는 사항:
  - 사람
    - 연락이 필요한 직원이 실제로 연락이 가능했고 연락처 목록이 최신 상태였나요?
    - 인시던트에 효과적으로 대응하고 조사하는 데 필요한 교육이나 역량을 갖춘 직원이 없었나요?
    - 적절한 리소스가 준비되어 있고 이용 가능했나요?
  - 프로세스
    - 프로세스와 절차를 준수했나요?
    - 이 (유형의) 인시던트에 대한 프로세스와 절차가 문서화되어 있고 사용 가능했나요?
    - 필요한 프로세스 및 절차가 누락되지는 않았나요?
    - 대응 담당자가 문제를 대응하는 데 필요한 정보에 적시에 액세스할 수 있었나요?
  - 기술
    - 기존 경고 시스템이 활동을 효과적으로 식별하고 경고했나요?
    - 어떻게 하면 탐지 시간을 50%까지 줄일 수 있을까요?
    - 기존 경고 시스템을 개선해야 하나요? 아니면 이 (인시던트 유형) 사고에 대해 새로운 경고 시스템을 구축해야 하나요?
    - 기존 도구로 인시던트를 효과적으로 조사(검색 및 분석)할 수 있었나요?
    - 이 (유형의) 인시던트를 더 빨리 식별하려면 어떻게 해야 할까요?
    - 이 (유형의) 인시던트가 재발하는 것을 방지하려면 어떻게 해야 할까요?
    - 개선 계획의 담당자는 누구이며 개선 계획이 실행되었는지 어떻게 테스트할 예정인가요?
    - 추가 모니터링 또는 예방적 통제 및 프로세스를 구현하고 테스트할 일정은 어떻게 되나요?

이 목록은 모든 것을 포함하지는 않지만, 조직 및 비즈니스 요구 사항이 무엇인지 식별하고 인시던트로부터 가장 효과적으로 학습하고 보안 태세를 지속적으로 개선하기 위해 이를 분석할 수 있는 방법을 식별하기 위한 출발점이 될 수 있습니다. 가장 중요한 것은 인시던트 대응 프로세스, 문서화 및 이해관계자 전반의 기대치에서 학습한 교훈을 표준으로 삼아 통합하는 것부터 시작하는 것입니다.

리소스

관련 문서:

- AWS Security Incident Response Guide - Establish a framework for learning from incidents
- NCSC CAF guidance - Lessons learned

애플리케이션 보안

질문

- SEC 11. 설계, 개발 및 배포 수명 주기 전반에 걸쳐 애플리케이션의 보안 속성을 어떻게 통합하고 검증하시나요?

SEC 11. 설계, 개발 및 배포 수명 주기 전반에 걸쳐 애플리케이션의 보안 속성을 어떻게 통합하고 검증하시나요?

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 코드 검토 수행
- 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

팀에 안전한 개발 및 운영 방식에 대한 교육을 제공하세요. 안전한 고품질 소프트웨어를 구축하는 데 도움이 됩니다. 이 방식을 통해 팀은 개발 수명 주기 초기에 보안 문제를 예방, 탐지 및 해결할 수 있습니다. 위협 모델링, 안전한 코딩 방식, 안전한 구성 및 운영을 위한 서비스 사용을 다루는 교육을 고려해 보세요. 셀프 서비스 리소스를 통해 팀이 교육에 액세스할 수 있도록 하고 지속적인 개선을 위해 정기적으로 피드백을 수집하세요.

원하는 성과: 처음부터 보안을 염두에 두고 소프트웨어를 설계하고 구축하는 데 필요한 지식과 기술을 팀에 제공합니다. 위협 모델링 및 안전한 개발 방식에 대한 교육을 통해 팀은 잠재적 보안 위험과 소프트웨어 개발 수명 주기(SDLC) 동안 이를 완화하는 방법을 깊이 이해할 수 있습니다. 이러한 선제적 보안 접근 방식은 팀 문화의 일부이며, 잠재적인 보안 문제를 조기에 식별하고 해결할 수 있습니다. 따라서 팀은 고품질의 안전한 소프트웨어와 기능을 더 효율적으로 제공하여 전체 제공 일정을 단축합니다. 조직 내에 보안 소유권이 모든 빌더에게 공유되는 협업적이고 포괄적인 보안 문화가 있습니다.

일반적인 안티 패턴:

- 보안 검토가 끝날 때까지 기다린 다음 시스템의 보안 속성을 고려합니다.
- 중앙의 보안 팀에 모든 보안 결정을 맡깁니다.
- SDLC에서 내린 결정이 조직의 전반적인 보안 기대치 또는 정책과 어떤 관련이 있는지에 대해 소통하지 않습니다.
- 보안 검토 프로세스를 너무 늦게 수행합니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

안전한 고품질 소프트웨어를 구축하기 위해 안전한 애플리케이션 개발 및 운영을 위한 일반적인 방식에 대해 팀에 교육을 제공합니다. 이 방식은 팀이 개발 수명 주기 초기에 보안 문제를 예방, 탐지 및 해결하는 데 도움이 될 수 있으며, 결과적으로 제공 일정이 단축될 수 있습니다.

이 방식이 자리 잡히도록 위협 모델링 워크숍과 같은 AWS 리소스를 사용하여 위협 모델링에 대해 팀을 교육하는 것을 고려합니다. 위협 모델링은 팀이 잠재적인 보안 위험을 이해하고 처음부터 보안을 염두에 두고 시스템을 설계하는 데 도움이 될 수 있습니다. 또한 안전한 개발 방식에 대한 AWS 교육 and Certification, 산업 또는 AWS 파트너 교육에 대한 액세스를 제공할 수 있습니다. 대규모 설계, 개발, 보안 및 효율적인 운영에 대한 포괄적인 접근 방식에 대한 자세한 내용은 AWS DevOps Guidance를 참조하세요.

조직의 보안 검토 프로세스를 명확하게 정의하고 소통하며 팀, 보안 팀 및 기타 이해관계자의 책임을 개괄적으로 설명합니다. 보안 요구 사항을 충족하는 방법을 다루는 자습형 지침, 코드 예시, 템플릿을 게시합니다. AWS CloudFormation, AWS 클라우드 개발 키트 (AWS CDK)(AWS CDK) Constructs 및 Service Catalog와 같은 AWS 서비스를 사용하여 사전 승인된 보안 구성을 제공하고 사용자 지정 설정의 필요성을 줄일 수 있습니다.

팀의 보안 검토 프로세스 및 교육 경험에 대해 정기적으로 피드백을 수집하고, 피드백을 바탕으로 지속적으로 개선합니다. 게임 데이 또는 버그 퇴치 캠페인을 수행하여 보안 문제를 식별하고 해결하는 동시에 팀의 스킬을 향상시킵니다.

구현 단계

1. 교육 요구 사항 식별: 설문조사, 코드 검토 또는 팀원과의 논의를 통해 안전한 개발 방식과 관련하여 팀 내 현재 스킬 수준 및 지식 격차를 평가합니다.
2. 교육 계획: 식별된 요구 사항에 따라 위협 모델링, 안전한 코딩 방식, 보안 테스트 및 안전한 배포 방식과 같은 관련 주제를 다루는 교육 계획을 수립합니다. 위협 모델링 워크숍, AWS 교육 and Certification, 산업 또는 AWS 파트너 교육 프로그램과 같은 리소스를 사용합니다.
3. 교육 일정 수립 및 제공: 팀을 위한 정기적인 교육 세션 또는 워크숍 일정을 잡습니다. 팀의 선호도와 시간적 여유에 따라 강사 주도형 또는 자기 주도형일 수 있습니다. 학습을 강화하기 위해 실습을 장려하고 실제 사례를 활용하도록 합니다.
4. 보안 검토 프로세스 정의: 보안 팀 및 기타 이해관계자와 협력하여 애플리케이션의 보안 검토 프로세스를 명확하게 정의합니다. 개발 팀, 보안 팀 및 기타 관련 이해관계자를 포함하여 프로세스에 관련된 각 팀 또는 개인의 책임을 문서화합니다.
5. 셀프 서비스 리소스 생성: 조직의 보안 요구 사항을 충족하는 방법을 보여주는 셀프 서비스 지침, 코드 예시 및 템플릿을 개발합니다. 사전 승인된 보안 구성을 제공하고 사용자 지정 설정의 필요성을 줄이기 위해 CloudFormation AWS CDK Constructs 및 Service Catalog와 같은 AWS 서비스를 고려합니다.
6. 소통 및 교류: 보안 검토 프로세스와 사용 가능한 셀프 서비스 리소스를 팀에 효과적으로 알립니다. 교육 세션 또는 워크숍을 수행하여 팀이 이러한 리소스를 숙지하도록 하고 리소스 사용 방법을 이해하고 있는지 확인합니다.

7. 피드백 수집 및 개선: 팀의 보안 검토 프로세스 및 교육 경험에 대해 정기적으로 피드백을 수집합니다. 이 피드백을 사용하여 개선이 필요한 영역을 식별하고 교육 자료, 셀프 서비스 리소스 및 보안 검토 프로세스를 지속적으로 개선합니다.
8. 보안 연습 수행: 게임 데이 또는 버그 퇴치 캠페인을 주최하여 애플리케이션 내의 보안 문제를 식별하고 해결합니다. 이러한 연습은 잠재적 취약성을 발견하는 데 도움이 될 뿐만 아니라, 팀이 보안 개발 및 운영에 대한 스킬을 강화하는 실용적인 학습 기회 역할을 합니다.
9. 지속적인 학습 및 개선: 팀이 최신 보안 개발 방식, 도구 및 기술을 파악하도록 장려합니다. 진화하는 보안 환경과 모범 사례를 반영하도록 교육 자료와 리소스를 정기적으로 검토하고 업데이트합니다.

리소스

관련 모범 사례:

- SEC11-BP08 보안 소유권이 워크로드 팀에 귀속되도록 프로그램 구축

관련 문서:

- AWS 교육 및 인증
- How to think about cloud security governance
- How to approach threat modeling
- Accelerating training – The AWS Skills Guild
- AWS DevOps Sagas

관련 비디오:

- Proactive security: Considerations and approaches

관련 예제:

- Workshop on threat modeling
- Industry awareness for developers

관련 서비스:

- AWS CloudFormation

- AWS 클라우드 개발 키트 (AWS CDK)(AWS CDK) Constructs
- 서비스 카탈로그

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 성과: 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 디바이스 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 디바이스 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 Group에서는 소스 코드 보안 분석기에 대한 비교 정보를 제공합니다. 여기에는 Byte Code Scanners 및 Binary Code Scanners에 대한 오픈 소스 도구도 포함됩니다.

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 Q Developers는 빌더의 IDE에서 보안 스캔을 포함한 코드 권장 사항을 제공합니다. Amazon CodeGuru Security는 애플리케이션 개발 중에 중요한 문제, 보안 문제 및 발견하기 어려운 버그를 식별할 수 있으며 코드 품질을 개선하기 위한 권장 사항을 제공합니다. 또한 소프트웨어 재료 사양서(SBOM)를 추출하면 소프트웨어 구축에 사용되는 다양한 구성 요소의 세부 정보 및 관계가 포함된 공식 레코드를 추출할 수 있습니다. 이것을 취약성 관리에 반영하고 소프트웨어 또는 구성 요소 종속성과 공급망 위험을 빠르게 식별하는 데 사용할 수 있습니다.

Security for Developers 워크숍에서는 SAST 및 DAST 테스트 방법론이 포함된 릴리스 파이프라인 자동화를 위해 AWS CodeBuild, AWS CodeCommit, AWS CodePipeline과 같은 AWS 개발자 도구를 사용합니다.

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. 이러한 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.
- Automated Security Helper(ASH)는 오픈 소스 코드 보안 스캔 도구의 예입니다.
- 개발자 IDE와 통합된 Amazon Q Developer, 커밋 시 코드 스캔을 위한 Amazon CodeGuru Security와 같은 자동화된 도구를 사용하여 테스트 또는 코드 분석을 수행하면 빌더가 적시에 피드백을 받을 수 있습니다.
- AWS Lambda를 사용하여 구축할 때 Amazon Inspector를 사용하여 함수의 애플리케이션 코드를 스캔할 수 있습니다.
- 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 수정 지침에 연결하면 빌더가 코드 품질을 개선하는 데 도움이 됩니다.
- 자동화된 도구의 결과를 정기적으로 분석하여 다음 자동화, 빌더 교육 또는 인식 캠페인의 우선순위를 지정합니다.
- CI/CD 파이프라인의 일부로 SBOM을 추출하려면 Amazon Inspector SBOM 생성기를 사용하여 아카이브, 컨테이너 이미지, 디렉터리, 로컬 시스템 및 컴파일된 Go 및 Rust 바이너리에 대한 SBOM을 CycloneDX SBOM 형식으로 생성합니다.

리소스

관련 모범 사례:

- DevOps 지침: DL.CR.3 Establish clear completion criteria for code tasks

관련 문서:

- 지속적 전송 및 지속적 배포

- AWS DevOps 컴피던시 파트너
- 애플리케이션 보안을 위한 AWS 보안 컴피턴시 파트너
- Choosing a Well-Architected CI/CD approach
- Secrets detection in Amazon CodeGuru Security

- Amazon CodeGuru Security Detection Library
- Accelerate deployments on AWS with effective governance
- AWS에서 안전하고 간편한 배포를 자동화하는 접근 방법
- How Amazon CodeGuru Security helps you effectively balance security and velocity

관련 비디오:

- Hands-off: Automating continuous delivery pipelines at Amazon
- Automating cross-account CI/CD pipelines
- The Software Development Prcess at Amazon
- Testing software and systems at Amazon

관련 예제:

- Industry awareness for developers
- Automated Security Helper (ASH)
- AWS CodePipeline Governance - Github

SEC11-BP03 정기적인 침투 테스트 시행

정기적으로 소프트웨어 침투 테스트를 시행하세요. 이러한 메커니즘은 자동화된 테스트나 수동 코드 검토로 감지할 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 성과: 침투 테스트는 애플리케이션의 보안 속성을 탐지, 수정 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 서드파티 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 마련합니다. 이 프로세스를 위협 모델에 기반하면 바람직한 방식으로 컨텍스트를 유지 관리할 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 테스트에서는 AWS에서의 침투 테스트에 대한 자세한 지침을 제공합니다.
- Accelerate deployments on AWS with effective governance
- AWS 보안 컴피턴시 파트너
- Modernize your penetration testing architecture on AWS Fargate
- AWS Fault injection Simulator

관련 예제:

- Automate API testing with AWS CodePipeline(GitHub)
- Automated security helper(GitHub)

SEC11-BP04 코드 검토 수행

코드 검토를 구현하여 개발 중인 소프트웨어의 품질과 보안을 확인합니다. 코드 검토에는 원래 코드 작성자 이외의 팀원이 코드에서 잠재적 문제, 취약성, 코딩 표준 및 모범 사례 준수 여부를 검토하도록 하는 것이 포함됩니다. 이 프로세스는 원래 개발자가 간과했을 수 있는 오류, 불일치 및 보안 결함을 포착하는 데 도움이 됩니다. 자동 도구를 사용하여 코드 검토를 지원합니다.

원하는 성과: 작성 중인 소프트웨어의 품질을 높이기 위해 개발 중에 코드 검토를 포함합니다. 코드 검토 중에 식별한 내용을 통해 경험이 부족한 팀원의 스킬을 향상합니다. 자동화 기회를 식별하고 자동화된 도구 및 테스트를 사용하여 코드 검토 프로세스를 지원합니다.

일반적인 안티 패턴:

- 배포 전에 코드 검토를 수행하지 않습니다.
- 같은 사람이 코드를 작성하고 검토합니다.
- 코드 검토를 지원하거나 오케스트레이션하는 데 자동화와 도구를 사용하지 않습니다.
- 빌더가 코드를 검토하기 전에 빌더에게 애플리케이션 보안 교육을 하지 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

코드 검토는 개발 중에 소프트웨어의 품질과 보안을 확인하는 데 도움이 됩니다. 수동 검토에는 원래 코드 작성자 이외의 팀원이 코드에서 잠재적 문제, 취약성, 코딩 표준 및 모범 사례 준수 여부를 검토하도록 하는 것이 포함됩니다. 이 프로세스는 원래 개발자가 간과했을 수 있는 오류, 불일치 및 보안 결함을 포착하는 데 도움이 됩니다.

자동 코드 검토를 수행하기 위해 Amazon CodeGuru Security를 고려합니다. CodeGuru Security는 기계 학습 및 자동 추론을 사용하여 코드를 분석하고 잠재적 보안 취약성 및 코딩 문제를 식별합니다. 자동화된 코드 검토를 기존 코드 리포지토리 및 지속적 통합/지속적 배포(CI/CD) 파이프라인과 통합합니다.

구현 단계

1. 코드 검토 프로세스 수립:
   - 코드를 주 브랜치에 병합하기 전 또는 프로덕션에 배포하기 전과 같이 코드 검토를 해야 하는 시기를 정의합니다.
   - 팀원, 선임 개발자, 보안 전문가와 같이 코드 검토 프로세스에 참여해야 하는 사람을 결정합니다.
   - 사용할 프로세스 및 도구를 포함하여 코드 검토 방법론을 결정합니다.
2. 코드 검토 도구 설정:
   - GitHub Pull 요청 또는 CodeGuru Security와 같이 팀의 요구 사항에 맞는 코드 검토 도구를 평가하고 선택합니다.
   - 선택한 도구를 기존 코드 리포지토리 및 CI/CD 파이프라인과 통합합니다.
   - 최소 검토자 수 및 승인 규칙과 같은 코드 검토 요구 사항을 적용하도록 도구를 구성합니다.
3. 코드 검토 체크리스트 및 지침 정의:
   - 검토해야 할 사항을 설명하는 코드 검토 체크리스트 또는 지침을 생성합니다. 코드 품질, 보안 취약성, 코딩 표준 준수 및 성능과 같은 요소를 고려합니다.

- 체크리스트 또는 지침을 개발 팀과 공유하고 모든 사람이 기대 사항을 이해하고 있는지 확인합니다.

4. 코드 검토 모범 사례에 대해 개발자 교육:
   - 효과적인 코드 검토를 수행하는 방법에 대해 팀에 교육을 제공합니다.
   - 검토 중에 찾아야 할 애플리케이션 보안 원칙과 일반적인 취약성에 대해 팀을 교육합니다.
   - 지식 공유를 장려하고 프로그래밍 세션을 병행하여 경험이 부족한 팀원의 스킬을 향상합니다.
5. 코드 검토 프로세스 구현:
   - Pull 요청 생성 및 검토자 할당과 같은 코드 검토 단계를 개발 워크플로에 통합합니다.
   - 코드 변경을 병합 또는 배포하기 전에 코드 검토를 거치도록 합니다.
   - 검토 프로세스 중에 열린 커뮤니케이션과 건설적인 피드백을 장려합니다.
6. 모니터링 및 개선:
   - 코드 검토 프로세스의 효과를 정기적으로 검토하고 팀으로부터 피드백을 수집합니다.
   - 코드 검토 프로세스를 간소화하기 위한 자동화 또는 도구 개선 기회를 식별합니다.
   - 알게 된 내용 및 업계 모범 사례를 기반으로 코드 검토 체크리스트 또는 지침을 지속적으로 업데이트하고 개선합니다.
7. 코드 검토 문화 조성:
   - 코드 품질과 보안을 유지하기 위해 코드 검토의 중요성을 강조합니다.
   - 코드 검토 프로세스에서 얻은 성공과 배운 내용을 축하하고 기념합니다.
   - 개발자가 편안하게 피드백을 주고받을 수 있는 협력적이고 지지하는 분위기를 장려합니다.

리소스

관련 모범 사례:

- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화

관련 문서:

- DevOps 지침: DL.CR.2 Perform peer review for code changes
- About pull requests in GitHub

관련 예제:

- Automate code reviews with Amazon CodeGuru Security
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Security CLI

관련 비디오:

- Continuous improvement of code quality with Amazon CodeGuru Security

SEC11-BP05 패키지 및 종속성 서비스의 중앙 집중화

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 중앙 집중화된 소스에서 얻을 수 있습니다.

원하는 성과: 작성하는 코드 외에도 외부 소프트웨어 패키지에서 워크로드를 구축합니다. 이렇게 하면 JSON 구문 분석기 또는 암호화 라이브러리와 같이 반복적으로 사용되는 기능을 더 간단하게 구현할 수 있습니다. 이러한 패키지 및 종속성에 대한 소스를 중앙 집중화하여 보안 팀이 사용하기 전에 검증할 수 있도록 합니다. 수동 및 자동 테스트 흐름과 함께 이 전략을 사용하여 개발하는 소프트웨어의 품질에 대한 신뢰도를 높입니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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:

- DevOps 지침: DL.CS.2 Sign code artifacts after each build
- Supply chain Levels for Software Artifacts (SLSA)

관련 예제:

- Accelerate deployments on AWS with effective governance
- Tighten your package security with CodeArtifact Package Origin Control toolkit
- Multi Region Package Publishing Pipeline(GitHub)
- Publishing Node.js Modules on AWS CodeArtifact using AWS CodePipeline(GitHub)
- AWS CDK Java CodeArtifact Pipeline Sample(GitHub)
- Distribute private .NET NuGet packages with AWS CodeArtifact(GitHub)

관련 비디오:

- Proactive security: Considerations and approaches
- The AWS Philosophy of Security (re:Invent 2017)
- When security, safety, and urgency all matter: Handling Log4Shell

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 성과: 테스트하는 워크로드의 버전은 배포하는 버전이며 배포는 매번 일관되게 수행됩니다. 워크로드 구성을 외부화하여 변경 없이 다양한 환경에 배포할 수 있습니다. 암호화된 서명 소프트웨어 패키지를 사용하여 환경 간에 변경된 내용이 없음을 확인합니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

강력하고 신뢰할 수 있는 애플리케이션 인프라를 유지하기 위해 안전하고 자동화된 배포 방식을 구현합니다. 이 방식에는 프로덕션 환경에서 지속적인 인적 액세스 제거, 배포를 위해 CI/CD 도구 사용, 환경별 구성 데이터 외부화가 포함됩니다. 이 접근 방식을 따르면 보안을 강화하고 인적 오류의 위험을 줄이며 배포 프로세스를 간소화할 수 있습니다.

AWS 계정 구조를 구축하여 프로덕션 환경에서 지속적인 인적 액세스를 제거할 수 있습니다. 이 방식은 무단 변경 또는 우발적 수정의 위험을 최소화하여 프로덕션 시스템의 무결성을 개선합니다. 사람의 직접적인 액세스 대신 AWS CodeBuild 및 AWS CodePipeline과 같은 CI/CD 도구를 사용하여 배포를 수행할 수 있습니다. 이러한 서비스를 사용하여 구축, 테스트 및 배포 프로세스를 자동화하여 수동 개입을 줄이고 일관성을 높일 수 있습니다.

보안 및 추적성을 더욱 강화하기 위해 애플리케이션 패키지를 테스트한 후 애플리케이션 패키지에 서명하고 배포 중에 이러한 서명을 검증할 수 있습니다. 이렇게 하려면 AWS Signer 또는 AWS Key Management Service(AWS KMS)와 같은 암호화 도구를 사용합니다. 패키지에 서명하고 확인하면 인증되고 검증된 코드만 환경에 배포할 수 있습니다.

또한 팀은 워크로드를 아키텍팅하여 AWS Systems Manager Parameter Store와 같은 외부 소스에서 환경별 구성 데이터를 얻을 수 있습니다. 이 방식은 애플리케이션 코드를 구성 데이터와 분리하므로 애플리케이션 코드 자체를 수정하지 않고도 독립적으로 구성을 관리하고 업데이트할 수 있습니다.

인프라 프로비저닝 및 관리를 간소화하려면 AWS CloudFormation 또는 AWS CDK와 같은 코드형 인프라(IaC) 도구를 사용하는 것이 좋습니다. 이러한 도구를 사용하여 인프라를 코드로 정의하여 다양한 환경에서 배포의 일관성과 반복성을 개선할 수 있습니다.

소프트웨어의 성공적인 배포를 검증하기 위해 카나리 배포를 고려합니다. 카나리 배포에는 전체 프로덕션 환경에 배포하기 전에 인스턴스 또는 사용자 하위 집합에 대한 변경 사항을 롤아웃하는 작업이 포함됩니다. 그런 다음 변경의 영향을 모니터링하고 필요한 경우 롤백하여 문제가 확산될 위험을 최소화할 수 있습니다.

Organizing Your AWS Environment Using Multiple Accounts 백서에 설명된 권장 사항을 따릅니다. 이 백서는 환경을 고유한 AWS 계정으로 분리(예: 개발, 스테이징 및 프로덕션)하는 방법에 대한 지침을 제공합니다. 이 방식은 보안 및 격리를 더욱 강화합니다.

구현 단계

1. AWS 계정 구조 설정:
   - Organizing Your AWS Environment Using Multiple Accounts 백서의 지침에 따라 다양한 환경(예: 개발, 스테이징 및 프로덕션)에 대해 별도의 AWS 계정을 생성합니다.
   - 각 계정에 대한 적절한 액세스 제어 및 권한을 구성하여 프로덕션 환경에 대한 사람의 직접적인 액세스를 제한합니다.
2. CI/CD 파이프라인 구현:
   - AWS CodeBuild 및 AWS CodePipeline과 같은 서비스를 사용하여 CI/CD 파이프라인을 설정합니다.
   - 각 환경에 애플리케이션 코드를 자동으로 구축, 테스트 및 배포하도록 파이프라인을 구성합니다.

- 버전 관리 및 코드 관리를 위해 코드 리포지토리를 CI/CD 파이프라인과 통합합니다.

3. 애플리케이션 패키지에 서명 및 확인:
   - AWS Signer 또는 AWS Key Management Service(AWS KMS)를 사용하여 애플리케이션 패키지를 테스트하고 검증한 후 서명합니다.
   - 대상 환경에 배포하기 전에 애플리케이션 패키지의 서명을 확인하도록 배포 프로세스를 구성합니다.
4. 구성 데이터 외부화:
   - 환경별 구성 데이터를 AWS Systems Manager Parameter Store에 저장합니다.
   - 배포 또는 런타임 중에 Parameter Store에서 구성 데이터를 검색하도록 애플리케이션 코드를 수정합니다.
5. 코드형 인프라(IaC) 구현:
   - AWS CloudFormation 또는 AWS CDK와 같은 IaC 도구를 사용하여 인프라를 코드로 정의하고 관리합니다.
   - CloudFormation 템플릿 또는 CDK 스크립트를 생성하여 애플리케이션에 필요한 AWS 리소스를 프로비저닝하고 구성합니다.
   - IaC를 CI/CD 파이프라인과 통합하여 애플리케이션 코드 변경과 함께 인프라 변경 사항을 자동으로 배포합니다.
6. 카나리 배포 구현:
   - 전체 프로덕션 환경에 배포하기 전에 인스턴스 또는 사용자 하위 집합에 변경 사항이 롤아웃되는 카나리 배포를 지원하도록 배포 프로세스를 구성합니다.
   - AWS CodeDeploy 또는 AWS ECS와 같은 서비스를 사용하여 카나리 배포를 관리하고 변경의 영향을 모니터링합니다.
   - 카나리 배포 중에 문제가 감지되면 롤백 메커니즘을 구현하여 이전의 안정적인 버전으로 되돌립니다.
7. 모니터링 및 감사:
   - 모니터링 및 로깅 메커니즘을 설정하여 배포, 애플리케이션 성능 및 인프라 변경을 추적합니다.
   - Amazon CloudWatch 및 AWS CloudTrail과 같은 서비스를 사용하여 로그 및 지표를 수집하고 분석합니다.
   - 감사 및 규정 준수 검사를 구현하여 보안 모범 사례 및 규제 요구 사항 준수 여부를 확인합니다.
8. 지속적 개선:
   - 배포 방식을 정기적으로 검토 및 업데이트하고 이전 배포에서 얻은 피드백과 배운 내용을 반영합니다.

- 배포 프로세스를 최대한 자동화하여 수동 개입과 잠재적인 인적 오류를 줄입니다.
- 여러 분야의 팀원으로 구성된 팀(예: 운영 또는 보안)과 협력하여 배포 방식을 조정하고 지속적으로 개선합니다.

이러한 단계를 따르면 AWS 환경에서 안전하고 자동화된 배포 방식을 구현하여 보안을 강화하고 인적 오류의 위험을 줄이며 배포 프로세스를 간소화할 수 있습니다.

리소스

관련 모범 사례:

- SEC11-BP02 개발 및 릴리스 수명 주기를 통한 테스트 자동화
- DL.CI.2 Trigger builds automatically upon source code modifications

관련 문서:

- Accelerate deployments on AWS with effective governance
- 안전하고 간편한 배포 자동화
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys
- Code Signing, a Trust and Integrity Control for AWS Lambda

관련 비디오:

- Hands-off: Automating continuous delivery pipelines at Amazon

관련 예제:

- Blue/Green deployments with AWS Fargate

SEC11-BP07 정기적으로 파이프라인의 보안 속성 평가

특히 권한 분리에 주의를 기울여 Well-Architected 보안 원칙을 파이프라인에 적용하세요. 파이프라인 인프라의 보안 속성을 정기적으로 평가합니다. 파이프라인 자체 보안을 효율적으로 관리하면 파이프라인을 통과하는 소프트웨어의 보안을 보장할 수 있습니다.

원하는 성과: 소프트웨어를 구축하고 배포하는 데 사용하는 파이프라인이 환경의 다른 워크로드와 동일한 권장 사례를 따릅니다. 파이프라인에서 구현하는 테스트는 이를 사용하는 팀에서 편집할 수 없습니다. 파이프라인에는 임시 자격 증명을 사용하여 수행하는 배포에 필요한 권한만 부여합니다. 파이프라인이 잘못된 환경에 배포되지 않도록 보호 조치를 구현합니다. 구축 환경의 무결성을 검증할 수 있도록 상태를 내보내도록 파이프라인을 구성합니다.

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 자격 증명을 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

배포 파이프라인은 소프트웨어 개발 수명 주기의 중요한 구성 요소이며 환경의 다른 워크로드와 동일한 보안 원칙 및 방식을 따라야 합니다. 여기에는 적절한 액세스 제어 구현, 입력 내용 검증, CI/CD 인프라와 관련된 권한의 정기적인 검토 및 감사가 포함됩니다.

애플리케이션 구축 및 배포를 담당하는 팀이 파이프라인에 구현된 보안 테스트 및 검사를 편집하거나 우회할 수 없는지 확인합니다. 이렇게 우려되는 사항이 발생하지 않도록 분리하면 구축 및 배포 프로세스의 무결성을 유지하는 데 도움이 됩니다.

먼저 AWS 배포 파이프라인 참조 아키텍처를 사용하는 것이 좋습니다. 이 참조 아키텍처는 AWS에서 CI/CD 파이프라인을 구축하기 위한 안전하고 확장 가능한 기반을 제공합니다.

또한 AWS Identity and Access Management Access Analyzer와 같은 서비스를 사용하여 파이프라인 권한 모두에 대해 최소 권한 IAM 정책을 생성하고 파이프라인의 한 단계로 워크로드 권한을 확인할 수 있습니다. 이렇게 하면 파이프라인과 워크로드에 특정 기능에 필요한 권한만 있는지 확인할 수 있으므로 무단 액세스 또는 무단 작업의 위험이 줄어듭니다.

구현 단계

- AWS 배포 파이프라인 참조 아키텍처부터 시작합니다.
- 파이프라인에 대한 최소 권한 IAM 정책을 프로그래밍 방식으로 생성하기 위해 AWS IAM Access Analyzer 사용을 고려하세요.
- 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(SNS)와 같은 대상으로 데이터를 라우팅할 수 있습니다.

리소스

관련 문서:

- AWS Deployment Pipelines Reference Architecture
- AWS CodePipeline 모니터링
- AWS CodePipeline의 보안 모범 사례

관련 예제:

- DevOps monitoring dashboard(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 성과: 팀에 보안 소유권과 의사 결정이 포함되어 있습니다. 보안에 대해 생각하는 방법을 팀에 교육했거나 소속된 인력 또는 관련 보안 인력으로 팀을 강화했습니다. 그 결과 팀은 개발 주기 초반에 더 양질의 보안 결정을 내립니다.

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 사례가 확립되지 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
- How to think about cloud security governance
- How AWS built the Security Guardians program, a mechanism to distribute security ownership
- How to build a Security Guardians program to distribute security ownership

관련 비디오:

- Proactive security: Considerations and approaches
- AppSec tooling and culture tips from AWS and Toyota Motor North America

신뢰성

신뢰성 원칙에서는 워크로드의 기능이 필요한 때에 기능을 정확하고 일관되게 수행하는 역량에 대해 다룹니다. 구현에 대한 권장 가이드는 신뢰성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 기본
- 워크로드 아키텍처
- 변경 관리
- 장애 관리

기본

Questions

- REL 1. 서비스 할당량과 제약 조건은 어떻게 관리하나요?
- REL 2. 네트워크 토폴로지는 어떻게 계획하나요?

REL 1. 서비스 할당량과 제약 조건은 어떻게 관리하나요?

클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 비율을 제한하여 서비스가 남용되지 않도록 하기 위해 존재합니다. 또한 리소스에도 제약이 따릅니다. 예를 들면, 광섬유 케이블을 통해 비트를 전송할 수 있는 속도나 물리적 디스크의 스토리지 용량 등이 있습니다.

모범 사례

- REL01-BP01 서비스 할당량 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 서비스 할당량 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 성과: 고객은 서비스 성능 저하 또는 중단을 일으킬 수 있는 서비스 할당량 및 제약 조건에 도달했는지 확인하기 위해 주요 지표, 인프라 검토 및 자동화 수정 단계를 모니터링하기 위한 적절한 지침을 구현하여 AWS 계정 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 규모를 조정하거나 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

Service Quotas은 한 곳에서 250개가 넘는 AWS 서비스에 대한 할당량을 관리할 수 있도록 도와주는 AWS 서비스입니다. 할당량 값을 조회하는 것에 더해 Service Quotas 콘솔 또는 AWS SDK를 사용하여 할당량 증가를 요청하고 추적할 수 있습니다. AWS Trusted Advisor에서는 일부 서비스의 특정 측면에 대한 사용량 및 할당량을 표시하는 서비스 할당량 확인 기능을 제공합니다. 서비스별 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. 해당하는 서비스에서 구성으로 설정되는 일부 제한으로는 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 지원에 문의하여 완화 방법이 있는지 확인하세요.

서비스 할당량은 리전에 따라 다를 수 있으며 특성상 글로벌로 적용될 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 리전에서 사용되는 DL Amazon EC2 인스턴스의 수를 제한합니다. Auto Scaling 그룹(ASG)을 사용하는 트래픽 규모 조정 이벤트 중에 이 한도에 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 처리량이 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 지원에 문의하여 완화 방법이 있는지 확인하세요. 고정 할당량 조정에 대한 자세한 내용은 REL01-BP03 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용 섹션을 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 요청하며, 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 3,000개의 서비스 할당량 이름이 있습니다.
- AWS 계정에서 사용되는 AWS 리소스를 모두 찾기 위해 AWS Config를 사용하여 할당량 분석을 보강합니다.

- AWS CloudFormation 데이터를 사용하여 사용된 AWS 리소스를 확인하세요. AWS Management Console 또는 `list-stack-resources`AWS CLI 명령으로 생성된 리소스를 살펴보세요. 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 사용합니다.
- 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 필요하다는 사실이 확인되면 필요한 할당량 변경을 계산하는 엔지니어링 절차를 수립합니다(REL01-BP05 할당량 관리 자동화 참조).
- 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 조건의 경우:

- 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 임곗값을 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework 신뢰성 원칙: 가용성
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS Limit Monitor on AWS 질문
- Amazon EC2 서비스 한도
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?

- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- 워크로드의 API 스로틀링 관리 및 모니터링
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

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 성과: 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

서비스 할당량은 계정별로 추적됩니다. 다른 언급이 없는 한, 각 할당량은 AWS 리전별로 다릅니다. 프로덕션 환경에 더해 적용 가능한 모든 비프로덕션 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 조정되는 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.

서비스 할당량 드리프트는 지정된 특정 할당량에 대한 서비스 할당량 제한이 모든 리전이 아니라 한 리전에서 변경되는 조건으로, 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% 임곗값 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 서비스 할당량 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework 신뢰성 원칙: 가용성
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS Limit Monitor on AWS 질문
- Amazon EC2 서비스 한도
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas

- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

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 간접 호출 페이로드 크기, API 게이트웨이의 제한 버스트 속도 및 데이터베이스에 대한 동시 사용자 연결 등이 여기에 포함됩니다.

원하는 성과: 애플리케이션 또는 서비스가 정상 조건 및 트래픽이 많은 조건에서 예상대로 수행됩니다. 해당 리소스의 고정 제약 조건 또는 서비스 할당량의 한계 내에서 작동하도록 설계되었습니다.

일반적인 안티 패턴:

- 규모를 조정할 때 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 조정할 수 없거나 수정할 수 없는 설계를 선택합니다. 예를 들어, SQS 페이로드 크기는 256KB입니다.

- 높은 트래픽 이벤트 동안 위험에 처할 수 있는 서비스 할당량의 임곗값을 모니터링하고 경고하도록 관찰성이 설계 및 구현되지 않았습니다.

이 모범 사례 확립의 이점: 애플리케이션이 중단이나 성능 저하 없이 예상되는 모든 서비스 부하 수준에서 실행되는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 표시된 경우 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 서비스에서는 대체 위치에서 이러한 제한을 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 서비스 할당량 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework 신뢰성 원칙: 가용성
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS Limit Monitor on AWS 질문
- Amazon EC2 서비스 한도
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas 사용 설명서
- Quota Monitor for AWS

- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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 성과: 관리 및 모니터링하는 액티브 자동화된 시스템을 배포합니다. 이러한 운영 솔루션은 할당량 사용량 임곗값에 거의 도달하고 있는지 확인합니다. 이러한 문제는 요청된 할당량 변경에 의해 사전에 해결됩니다.

일반적인 안티 패턴:

- 서비스 할당량 임곗값을 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 관리에 템플릿을 활용하지 않습니다.

이 모범 사례 확립의 이점: AWS 서비스 할당량을 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 간접 호출할 수 있습니다. CloudWatch Logs의 지표 필터를 사용하여 로그의 패턴을 검색하고 추출하여 사용량이 할당량 임곗값에 근접하는지 여부를 확인할 수도 있습니다.

구현 단계

모니터링의 경우:

- 현재 리소스 사용 내역(버킷, 인스턴스)을 파악합니다. 현재 리소스 사용 내역을 모두 보려면 Amazon EC2 `DescribeInstances` API와 같은 서비스 API를 사용합니다.
- 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 클라우드 개발 키트 (AWS CDK)
- AWS Service Quotas를 사용합니다. 한 곳에서 250개가 넘는 AWS 서비스에 대한 할당량을 관리할 수 있도록 도와주는 AWS 서비스입니다.
- Trusted Advisor 서비스 한도를 사용하여 다양한 임곗값에서 현재 서비스 한도를 모니터링합니다.
- 서비스 할당량 내역(콘솔 또는 AWS CLI)을 사용하여 리전별 증가를 확인합니다.
- 필요한 경우 각 리전 및 각 계정의 서비스 할당량 변경 사항을 비교하여 동등성을 생성합니다.

관리의 경우:

- 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 서비스 할당량 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework 신뢰성 원칙: 가용성
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS Limit Monitor on AWS 질문
- Amazon EC2 서비스 한도
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads

- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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

AWS 서비스에서 한도라고도 하는 서비스 할당량은 AWS 계정 계정의 리소스에 대한 최댓값입니다. 각 AWS 서비스는 할당량 세트와 기본값을 정의합니다. 필요한 모든 리소스에 대한 액세스를 워크로드에 제공하려면 서비스 할당량 값을 늘려야 할 수 있습니다.

AWS 리소스의 워크로드 소비가 증가하면 워크로드 안정성이 위협받고 할당량이 초과되면 사용자 경험에 영향을 미칠 수 있습니다. 워크로드가 한도에 가까워지면 알림을 보내고 할당량 증가 요청을 자동으로 생성하는 것을 고려하는 도구를 구현합니다.

원하는 성과: 각 AWS 계정 및 리전에서 실행되는 워크로드에 맞게 할당량이 적절하게 구성되어 있습니다.

일반적인 안티 패턴:

- 워크로드 요구 사항을 충족하도록 할당량을 적절하게 고려하고 조정하지 못합니다.
- 스프레드시트와 같이 더 이상 효용이 없을 수 있는 방법을 사용하여 할당량 및 사용량을 추적합니다.
- 정기적인 일정에 따라서만 서비스 한도를 업데이트합니다.
- 조직은 기존 할당량을 검토하고 필요한 경우 서비스 할당량 증가를 요청하는 운영 프로세스가 부족합니다.

이 모범 사례 확립의 이점:

- 향상된 워크로드 복원력: AWS 리소스 할당량을 초과하여 발생하는 오류를 방지합니다.
- 간소화된 재해 복구: 다른 AWS 리전에서 DR을 설정하는 동안 기본 리전에 구축된 자동 할당량 관리 메커니즘을 재사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS Service Quotas 콘솔, AWS Command Line Interface(AWS CLI) 및 AWS SDK 등의 메커니즘을 사용하여 현재 할당량을 보고 진행 중인 할당량 소비를 추적합니다. 구성 관리 데이터베이스(CMDB) 및 IT 서비스 관리(ITSM) 시스템을 AWS Service Quota API과 통합할 수도 있습니다.

할당량 사용량이 정의된 임계값에 도달하면 자동 알림을 생성하고 알림을 받을 때 할당량 증가 요청을 제출하는 프로세스를 정의합니다. 기본 워크로드가 비즈니스에 중요한 경우 할당량 증가 요청을 자동화할 수 있지만 성장 피드백 루프와 같은 런어웨이 작업의 위험을 방지하기 위해 자동화를 신중하게 테스트할 수 있습니다.

비교적 작은 할당량 증가는 종종 자동으로 승인됩니다. 더 큰 할당량 요청은 AWS 지원에서 수동으로 처리해야 할 수 있으며 검토 및 처리하는 데 추가 시간이 걸릴 수 있습니다. 여러 요청 또는 대규모 증가 요청을 처리하는 데 추가 시간이 드는 것을 감안합니다.

구현 단계

- 서비스 할당량에 대한 자동 모니터링을 구현하고 워크로드의 리소스 사용률이 할당량 한도에 도달하면 알림을 발행합니다. 예를 들어 AWS용 Quota Monitor는 서비스 할당량에 대한 자동 모니터링을

제공할 수 있습니다. 이 도구는 AWS Organizations과 통합되고 Cloudformation StackSets를 사용하여 배포하므로 새 계정이 생성되면 자동으로 모니터링됩니다.

- Service Quotas 요청 템플릿 또는 AWS Control Tower와 같은 기능을 사용하여 새 계정에 대한 Service Quotas 설정을 간소화합니다.
- 모든 AWS 계정 및 리전에 대한 현재 서비스 할당량 사용 대시보드를 구축하고 할당량 초과를 방지하기 위해 필요에 따라 참조합니다. Cloud Intelligence Dashboards의 일부인 Trusted Advisor Organizational(TAO) Dashboard를 사용하면 이러한 대시보드를 빠르게 시작할 수 있습니다.
- 서비스 한도 증가 요청을 추적합니다. Consolidated Insights from Multiple Accounts(CIMA)는 모든 요청에 대한 조직 수준 보기를 제공할 수 있습니다.
- 비프로덕션 계정에서 할당량 임계값을 낮게 설정하여 알림 생성 및 할당량 증가 요청 자동화를 테스트합니다. 프로덕션 계정에서 이러한 테스트를 수행하지 마세요.

리소스

관련 모범 사례:

- OPS10-BP07 이벤트 대응 자동화

관련 문서:

- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS의 Quota Monitor 솔루션 - AWS 솔루션
- What is Service Quotas?
- What is Service Quotas request templates?

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

관련 도구:

- Quota Monitor for AWS

REL01-BP06 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인

이 문서에서는 리소스 할당량과 사용량 사이에서 일정 간격을 유지하는 방법과 리소스 할당량이 조직에 어떤 이점을 줄 수 있는지 설명합니다. 리소스 사용을 완료한 후에도 사용량 할당량에서는 해당 리소스를 계속 고려할 수 있습니다. 이로 인해 리소스에서 장애가 실패하거나 리소스에 액세스하지 못할 수 있습니다. 액세스할 수 없는 리소스와 대체 리소스가 중복되는 부분이 할당량에 반영되는지 확인하여 리소스 장애를 방지합니다. 이 차이를 계산할 때 네트워크 장애, 가용 영역 장애 또는 리전 장애와 같은 사용 사례를 고려합니다.

원하는 성과: 리소스 또는 리소스 액세스 가능성에서 발생하는 작거나 큰 장애는 현재 서비스 임곗값 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 복구하는 전략을 구현합니다. 추가 리소스를 만들어 서비스 한도를 소진하지 않으면서 장애 조치 조건을 수용할 수 있도록 액세스할 수 없는 리소스를 대체하는 전략이 한 가지 예입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

할당량 한도를 평가할 때 일부 성능 저하로 인해 발생할 수 있는 장애 조치 사례를 고려합니다. 다음 장애 조치 사례를 고려합니다.

- 장애가 발생했거나 액세스할 수 없는 VPC.
- 액세스할 수 없는 서브넷.

- 리소스 액세스 가능성에 영향을 미치는 성능이 저하된 가용 영역.
- 네트워킹 경로 또는 수신 및 송신 지점이 차단되거나 변경됩니다.
- 리소스 액세스 가능성에 영향을 미치는 성능이 저하된 리전.
- 리전 또는 가용 영역에서 발생한 장애로 영향을 받는 리소스의 하위 세트.

장애 조치 결정은 비즈니스에 미치는 영향이 크게 다를 수 있으므로 상황마다 다릅니다. 애플리케이션 또는 서비스 장애 조치를 결정하기 전에 장애 조치 위치에서 리소스의 용량 계획 및 해당 리소스의 할당량을 해결합니다.

각 서비스의 할당량을 검토할 때 정상적인 활동 피크보다 높은 상황을 고려합니다. 이러한 피크는 네트워킹 또는 권한으로 인해 액세스할 수 없지만 여전히 활성 상태인 리소스와 관련이 있을 수 있습니다. 종료되지 않은 활성 리소스는 여전히 서비스 할당량 한도에 포함됩니다.

구현 단계

- 장애 조치와 액세스 가능성 손실을 수용할 수 있도록 서비스 할당량과 최대 사용량 사이에서 일정 간격을 유지합니다.
- 서비스 할당량을 결정합니다. 일반적인 배포 패턴, 가용성 요구 사항, 사용량 증가를 고려합니다.
- 필요한 경우 할당량 증가를 요청합니다. 할당량 증가 요청에 대한 대기 시간을 예상합니다.
- 신뢰성 요구 사항(9의 개수로도 표현)을 확인합니다.
- 구성 요소, 가용 영역 또는 리전의 손실과 같은 잠재적 장애 시나리오를 파악합니다.
- 배포 방법(예: canary, 블루/그린, 레드/블랙, 롤링)을 설정합니다.
- 현재 할당향 한도에 적절한 버퍼를 포함합니다. 예를 들어 버퍼는 15%일 수 있습니다.
- 적절한 경우 정적 안정성(영역 및 리전)에 대한 계산을 포함합니다.
- 사용량 증가 계획을 세우고 사용 추세를 모니터링합니다.
- 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 올바른 양과 유형의 리소스를 확보하여 가장 중요한 비즈니스 일정에서 잠재적 위험을 줄일 수 있는 유용한 전략이 될 수 있습니다.

리소스

관련 모범 사례:

- REL01-BP01 서비스 할당량 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Well-Architected Framework 신뢰성 원칙: 가용성
- AWS Service Quotas(이전의 서비스 한도)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS Limit Monitor on AWS 질문
- Amazon EC2 서비스 한도
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas 사용 설명서
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?
- APN 파트너: 구성 관리를 지원할 수 있는 파트너
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS

- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

관련 비디오:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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 계획하나요?

워크로드는 여러 환경에 존재하는 경우가 많습니다. 여기에는 여러 클라우드 환경(퍼블릭 액세스 가능 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 중복 연결 프로비저닝
- REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인
- REL02-BP04 다대다 메시보다 허브 앤 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 성과: 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, Elastic Load Balancing(ELB) 모두 고가용성 퍼블릭 엔드포인트를 제공합니다. Amazon Route 53은 퍼블릭 엔드포인트 주소를 확인할 수 있는지 검증하기 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.
- 분산 서비스 거부(DDoS) 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 같은 가용성과 확장성이 뛰어난 도메인 이름 시스템 (DNS) 을 사용하여 도메인의 DNS 레코드를 관리하세요. Amazon Route 53에서 제공하는 상태 확인을 사용할 수도 있습니다. 상태 확인은 애플리케이션이 도달 가능하고 사용 가능하며 작동하는지 확인하고, 웹 페이지 또는 특정 URL 요청과 같은 사용자 행동을 모방하는 방식으로 설정할 수 있습니다. 장애가 발생하면 Amazon Route 53은 DNS 확인 요청에 응답하고 트래픽을 정상 상태인 엔드포인트로만 보냅니다. Amazon Route 53에서 제공하는 지리적 DNS 및 지연 시간 기반 라우팅 기능을 사용할 수도 있습니다.

워크로드 자체의 고가용성 여부를 확인하려면 Elastic Load Balancing(ELB)을 사용합니다. Amazon Route 53을 사용하여 트래픽을 대상 컴퓨팅 인스턴스로 분산하는 ELB로 트래픽 대상을 지정할 수 있습니다. 서버리스 솔루션을 위해 AWS Lambda와 함께 Amazon API Gateway를 사용할 수도 있습니다. 고객은 여러 AWS 리전에서 워크로드를 실행할 수도 있습니다. 멀티사이트 액티브/액티브 패턴을 사용하면 워크로드가 여러 리전의 트래픽을 처리할 수 있습니다. 다중 사이트 액티브/액티브 패턴을 사용하면 워크로드가 활성 리전의 트래픽을 지원합니다. 그동안 데이터는 보조 리전으로 복제되고 기본 리전에서 장애 발생 시 활성화됩니다. 그런 다음, Route 53 상태 확인을 통해 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 가

용성도 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 AWS에서는 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 규모의 공격(예: UDP 플러드), 상태 고갈 공격(예: TCP SYN 플러드)에 대한 추가 보호를 제공하고 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 공격으로부터 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 configuring Amazon Route 53 as your DNS service를 참조하세요.
2. 상태 확인 설정: Route 53을 사용할 때 정상 대상만 확인할 수 있는지 확인합니다. Route 53 상태 확인 생성 및 DNS 장애 조치 구성 작업부터 시작합니다. 상태 확인을 설정할 때 다음 측면을 고려해야 합니다.
   a. Amazon Route 53이 상태 확인의 정상 여부를 판단하는 방법
   b. 상태 확인의 생성, 업데이트 및 삭제
   c. 상태 확인의 상태 모니터링 및 알림 수신
   d. Amazon Route 53 DNS 모범 사례
3. DNS 서비스를 엔드포인트에 연결.

   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 전송합니다.
4. 콘텐츠 전송 네트워크를 결정합니다.
   a. 사용자에게 더 가까운 엣지 로케이션을 사용하여 콘텐츠를 제공하려면 먼저 CloudFront가 콘텐츠를 제공하는 방법을 이해합니다.
   b. 간단한 CloudFront 배포로 시작합니다. 그런 다음 CloudFront는 콘텐츠를 제공할 위치와 콘텐츠 제공을 추적 및 관리하는 방법에 대한 세부 정보를 확인합니다. CloudFront 배포를 설정할 때 다음 측면을 이해하고 고려해야 합니다.
      i. 캐싱과 CloudFront 엣지 로케이션의 작동 방식
      ii. CloudFront 캐시에서 직접 처리되는 요청의 비율 증가(캐시 적중률)
      iii. Amazon CloudFront Origin Shield 사용
      iv. CloudFront 오리진 장애 조치를 통한 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 이해하려면 AWS WAF의 작동 방식을 검토하고 애플리케이션 계층 HTTP POST 및 GET 플러드로부터 보호를 구현할 준비가 되면 Getting started with AWS WAF를 검토하세요. CloudFront와 함께 AWS WAF를 사용할 수도 있습니다.(how AWS WAF works with Amazon CloudFront features 참조).
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 복원력이 뛰어난 DDoS 아키텍처 예제를 검토할 수 있습니다. DDoS 공격으로부터 워크로드와 퍼블릭 엔드포인트를 보호하려면 Getting started with AWS Shield Advanced를 검토하세요.

리소스

관련 모범 사례:

- REL10-BP01 워크로드를 여러 위치에 배포
- REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 플레인 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
- AWS Global Accelerator란 무엇입니까?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- Elastic Load Balancing이란 무엇인가요?
- Network Connectivity capability - Establishing Your Cloud Foundations
- Amazon API Gateway란 무엇입니까?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?
- Amazon Application Recovery Controller란 무엇입니까?
- DNS 장애 조치에 대한 사용자 지정 상태 확인 구성

관련 비디오:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

관련 예제:

- Amazon ARC(Application Recovery Controller)를 사용한 재해 복구
- AWS Global Accelerator 워크숍

REL02-BP02 클라우드와 온프레미스 환경의 프라이빗 네트워크 간에 중복 연결 프로비저닝

클라우드와 온프레미스 환경의 프라이빗 네트워크 간 연결에 중복 구성을 구현하여 연결 복원력을 확보합니다. 이는 두 개 이상의 링크와 트래픽 경로를 배포하여 네트워크 장애 발생 시 연결을 유지함으로써 달성할 수 있습니다.

일반적인 안티 패턴:

- 하나의 네트워크 연결에만 의존하므로 단일 장애 지점이 발생합니다.
- 하나의 VPN 터널만 사용하거나 동일한 가용 영역에서 끝나는 여러 터널을 사용합니다.
- VPN 연결을 위해 하나의 ISP에 의존하므로 ISP 중단 시 완전한 장애가 발생할 수 있습니다.
- 네트워크 중단 발생 시 트래픽을 다시 라우팅하는 데 중요한 BGP와 같은 동적 라우팅 프로토콜을 구현하지 않습니다.
- VPN 터널의 대역폭 제한을 무시하고 백업 기능을 과대평가합니다.

이 모범 사례 확립의 이점: 클라우드 환경과 기업 또는 온프레미스 환경 간에 중복 연결을 구현하면 두 환경 간의 종속 서비스가 신뢰할 수 있는 기반에서 통신할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS Direct Connect를 사용하여 온프레미스 네트워크를 AWS에 연결하는 경우 둘 이상의 온프레미스 위치와 둘 이상의 AWS Direct Connect 위치에 있는 개별 디바이스에서 끝나는 별도의 연결을 사용하여 최대 네트워크 복원력(99.99% SLA)을 달성할 수 있습니다. 이 토폴로지는 디바이스 장애, 연결 문제 및 완전한 위치 중단에 대한 복원력을 제공합니다. 또는 여러 위치에 대한 두 개의 개별 연결(각 온프레미스 위치가 단일 Direct Connect 위치에 연결됨)을 사용하여 높은 복원력(SLA 99.9%)을 달성할 수 있습니다. 이 접근 방식은 광케이블 절단이나 디바이스 장애로 인한 연결 중단을 방지하고 전체 위치 장애를 완화하는 데 도움이 됩니다. AWS Direct Connect 복원 도구는 AWS Direct Connect 토폴로지 설계에 도움이 될 수 있습니다.

또한 기본 AWS Direct Connect 연결에 대한 비용 효율적인 백업으로 AWS Transit Gateway에서 끝나는 AWS Site-to-Site VPN을 고려할 수 있습니다. 이 설정을 사용하면 여러 VPN 터널에서 Equal-Cost Multipath(ECMP) 라우팅이 가능하므로 각 VPN 터널이 1.25Gbps로 제한되어 있더라도 최대 50Gbps의 처리량이 가능합니다. 하지만 네트워크 중단을 최소화하고 안정적인 연결을 제공하는 데는 AWS Direct Connect가 여전히 가장 효과적인 선택이라는 점에 유의해야 합니다.

인터넷을 통해 VPN을 사용하여 클라우드 환경을 온프레미스 데이터 센터에 연결하는 경우, 단일 사이트 간 VPN 연결의 일부로 두 개의 VPN 터널을 구성합니다. 각 터널은 고가용성을 위해 서로 다른 가용 영역에서 끝나야 하며, 이중화된 하드웨어를 사용하여 온프레미스 디바이스 장애를 방지해야 합니다. 또한 단일 ISP 중단으로 인해 VPN 연결이 완전히 중단되는 것을 방지하려면 온프레미스 위치에서 다양한 인터넷 서비스 제공업체(ISP)의 여러 인터넷 연결을 고려합니다. 다양한 라우팅과 인프라를 갖춘 ISP, 특히 AWS 엔드포인트에 대한 별도의 물리적 경로가 있는 ISP를 선택하면 연결 가용성이 높아집니다.

다중 AWS Direct Connect 연결 및 다중 VPN 터널(또는 둘의 조합)을 통한 물리적 이중화 외에도 Border Gateway Protocol(BGP) 동적 라우팅을 구현하는 것도 중요합니다. 동적 BGP는 실시간 네트워크 상태 및 구성된 정책을 기반으로 경로 간에 트래픽을 자동으로 다시 라우팅합니다. 이러한 동적 동작은 링크 또는 네트워크 장애 발생 시 네트워크 가용성과 서비스 연속성을 유지하는 데 특히 유용합니다. 이는 대체 경로를 빠르게 선택하여 네트워크의 복원력과 신뢰성을 향상시킵니다.

구현 단계

- AWS 및 온프레미스 환경 간에 가용성이 뛰어난 연결을 확보합니다.
  - 별도로 배포된 프라이빗 네트워크 간에 여러 AWS Direct Connect 연결 또는 VPN 터널을 사용합니다.
  - 가용성을 높이려는 경우에는 여러 AWS Direct Connect 위치를 사용합니다.
  - 여러 AWS 리전을 사용하는 경우 2개 이상의 위치에서 중복 구성을 생성합니다.
- 가능하면 AWS Transit Gateway를 사용하여 VPN 연결을 종료합니다.
- AWS Marketplace 어플라이언스를 평가하여 VPN을 종료하거나 SD-WAN을 AWS로 확장합니다. AWS Marketplace 어플라이언스를 사용하는 경우 다른 가용 영역에서 고가용성을 위해 중복 인스턴스를 배포합니다.
- 온프레미스 환경에 중복 연결을 제공합니다.
  - 가용성 요구 사항을 충족하려면 여러 AWS 리전에 대한 중복 연결이 필요할 수 있습니다.
  - AWS Direct Connect Resiliency Toolkit을 사용하여 시작합니다.

리소스

관련 문서:

- AWS Direct Connect Resiliency Recommendations
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Routing policies and BGP communities
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 백서
- 확장 가능하고 안전한 다중 VPC AWS 네트워크 인프라 구축
- Using redundant Site-to-Site VPN connections to provide failover

- Using the AWS Direct Connect Resiliency Toolkit to get started
- VPC 엔드포인트 및 VPC 엔드포인트 서비스(AWS PrivateLink)
- Amazon VPC란 무엇인가?
- What is a transit gateway?
- AWS Site-to-Site VPN란 무엇입니까?
- Direct Connect 게이트웨이 사용

관련 비디오:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs

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 준수하세요.

- 리전당 두 개 이상의 VPC에 대한 IP 주소 공간을 허용합니다.
- VPC 내에서 여러 가용 영역을 포함할 수 있도록 여러 서브넷을 위한 공간을 허용합니다.
- 사용되지 않은 CIDR 블록 공간은 향후 확장을 위해 VPC 내에 남겨둡니다.
- 기계 학습용 스팟 플릿, Amazon EMR 클러스터 또는 Amazon Redshift 클러스터 등 사용할 수 있는 임시 Amazon EC2 인스턴스 플릿의 요구 사항을 충족할 IP 주소 공간이 있는지 확인합니다. 각 Kubernetes 포드에는 기본적으로 VPC CIDR 블록의 라우팅 가능한 주소가 할당되므로 Amazon Elastic Kubernetes Service(Amazon EKS)와 같은 Kubernetes 클러스터도 비슷하게 고려해야 합니다.
- 참고로 각 서브넷 CIDR 블록에서 처음 4개의 IP 주소와 마지막 IP 주소는 예약되므로 사용할 수 없습니다.
- 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 가장 큰 VPC CIDR 블록은 /16이고 가장 작은 블록은 /28입니다.
- 다른 연결된 네트워크(VPC, 온프레미스 또는 기타 클라우드 제공업체)를 고려하고 중복되지 않는 IP 주소 공간을 확보하세요. 자세한 내용은 REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용를 참조하세요.

원하는 성과: 확장 가능한 IP 서브넷은 향후 성장을 수용하고 불필요한 낭비를 방지하는 데 도움이 될 수 있습니다.

일반적인 안티 패턴:

- 향후 성장을 고려하지 않으면 CIDR 블록이 너무 작아지고 재구성이 필요하여 가동 중단이 발생할 수 있습니다.
- Elastic Load Balancer가 사용할 수 있는 IP 주소 수를 잘못 추정합니다.
- 트래픽이 많은 여러 로드 밸런서를 동일한 서브넷에 배포
- IP 주소 사용량 모니터링에 실패하면서 자동 규모 조정 메커니즘을 사용합니다.
- CIDR 범위가 미래 성장 기대치를 훨씬 상회하는 지나치게 큰 것으로 정의하면 주소 범위가 겹치는 다른 네트워크와의 피어링이 어려울 수 있습니다.

이 모범 사례 확립의 이점: 이렇게 하면 워크로드의 증가를 수용하고 확장 시 가용성을 계속 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

성장, 규정 준수 및 다른 제품과 통합을 수용할 수 있도록 네트워크 계획을 수립합니다. 성장은 과소 평가될 수 있고 규정 준수는 변경될 수 있으며 적절한 계획 없이는 프라이빗 네트워크 연결을 구현하기가 어려울 수 있습니다.

- 서비스 요구 사항, 지연 시간, 규정, 재해 복구(DR) 요구 사항을 기준으로 관련 AWS 계정 및 리전을 선택합니다.
- 리전별 VPC 배포에 대한 요구 사항을 파악합니다.
- VPC 크기를 파악합니다.
  - 다중 VPC 연결을 배포할 것인지 여부를 결정합니다.
    - What Is a Transit Gateway?
    - Single Region Multi-VPC Connectivity

- 규정 요구 사항에 따라 분리된 네트워킹이 필요한지 결정합니다.
- 현재 및 미래의 요구 사항을 수용할 수 있도록 적절한 크기의 CIDR 블록으로 VPC를 만드세요.
  - 성장 전망을 알 수 없는 경우 향후 다시 구성할 필요가 없도록 더 큰 CIDR 블록을 사용하는 것이 좋습니다.
- 이중 스택 VPC의 일부로 서브넷에 IPv6 주소 지정 지정을 고려하세요. IPv6은 대규모 IPv4 주소를 필요로 하는 임시 인스턴스 또는 컨테이너를 포함하는 프라이빗 서브넷에서 사용하기에 적합합니다.

리소스

관련 Well-Architected 모범 사례:

- REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 백서
- 다중 데이터 센터 HA 네트워크 연결
- Single Region Multi-VPC Connectivity
- Amazon VPC란 무엇인가?
- AWS에서의 IPv6
- IPv6 on reference architectures
- Amazon Elastic Kubernetes Service launches IPv6 support
- VPC - Classic Load Balancer에 대한 권장 사항
- 가용 영역 서브넷 - Application Load Balancer
- 가용 영역 - Network Load Balancer

관련 비디오:

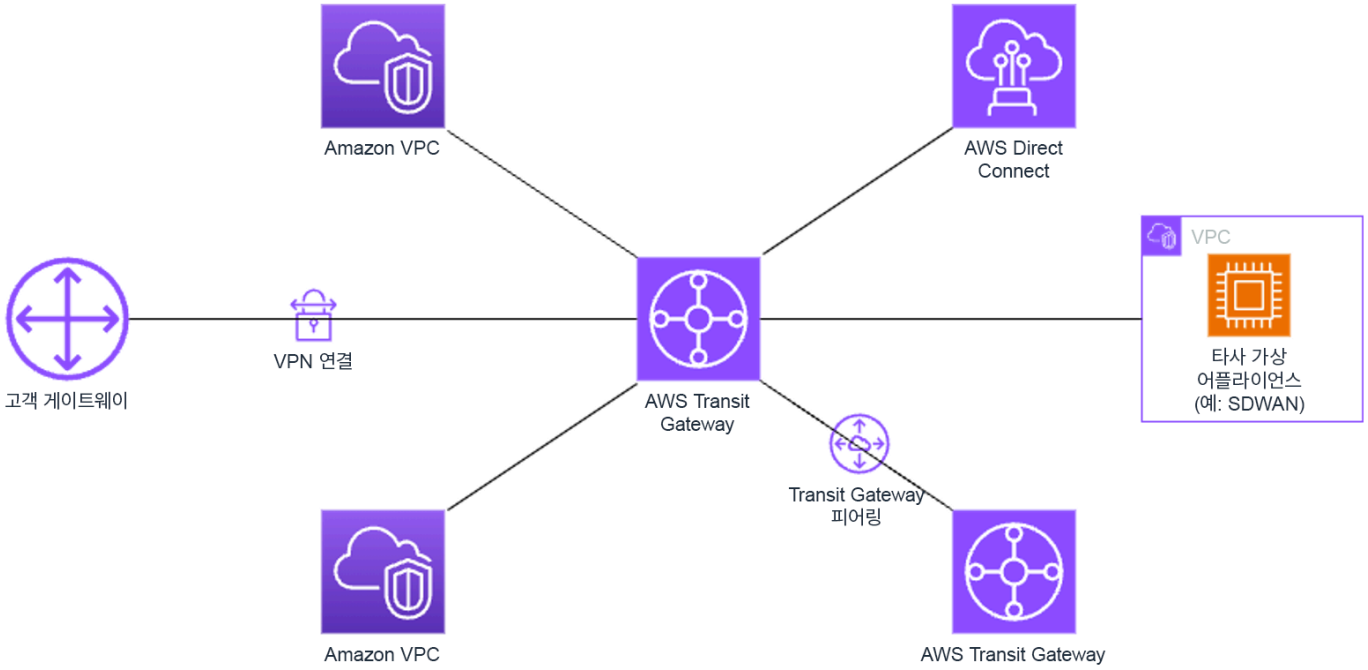
- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

REL02-BP04 다대다 메시보다 허브 앤 스포크 토폴로지 선호

Virtual Private Cloud(VPC) 및 온프레미스 네트워크와 같은 여러 프라이빗 네트워크를 연결할 때는 메시 토폴로지보다 허브 앤 스포크 토폴로지를 선택하세요. 메시 토폴로지는 각 네트워크가 다른 네트워크에 직접 연결되어 복잡성과 관리 오버헤드가 증가하지만 허브 앤 스포크 아키텍처는 단일 허브를 통해 연결을 중앙 집중화합니다. 이러한 중앙 집중화로 네트워크 구조가 단순해지고 운영성, 확장성과 제어 가능성이 높아집니다.

AWS Transit Gateway는 AWS에서 허브 앤 스포크 네트워크를 구축할 수 있도록 설계되었으며 확장성과 가용성이 뛰어난 관리형 서비스입니다. 네트워크의 중앙 허브 역할을 하면서 네트워크 세분화, 중앙 집중식 라우팅, 클라우드와 온프레미스 환경 모두에 대한 간소화된 연결을 제공합니다. 다음 그림은 AWS Transit Gateway를 사용하여 허브 앤 스포크 토폴로지를 빌드하는 방법을 보여줍니다.



원하는 성과: 중앙 허브를 통해 가상 프라이빗 클라우드(VPC)와 온프레미스 네트워크를 연결했습니다. 확장성이 뛰어난 클라우드 라우터 역할을 하는 허브를 통해 피어링 연결을 구성합니다. 복잡한 피어링 관계를 사용할 필요가 없으므로 라우팅이 간소화됩니다. 네트워크 간 트래픽은 암호화되며 네트워크를 격리할 수 있습니다.

일반적인 안티 패턴:

- 복잡한 네트워크 피어링 규칙을 구축합니다.
- 네트워크 간에 서로 통신해서는 안 되는 경로를 제공합니다(예: 상호 의존성이 없는 별도의 워크로드).
- 허브 인스턴스의 거버넌스가 비효율적입니다.

이 모범 사례 확립의 이점: 연결된 네트워크 수가 증가함에 따라 메시 연결의 관리 및 확장은 점점 더 어려워집니다. 메시 아키텍처는 추가 인프라 구성 요소, 구성 요구 사항 및 배포 고려 사항과 같은 추가 문제를 야기합니다. 또한 메시는 데이터 플레인 및 컨트롤 플레인 구성 요소를 관리하고 모니터링하기 위한 추가 오버헤드를 도입합니다. 메시 아키텍처의 고가용성을 제공하는 방법, 메시 상태 및 성능을 모니터링하는 방법, 메시 구성 요소의 업그레이드를 처리하는 방법을 고려해야 합니다.

반면 허브 앤 스포크 모델은 여러 네트워크에서 중앙 집중식 트래픽 라우팅을 설정합니다. 데이터 플레인 및 컨트롤 플레인 구성 요소의 관리 및 모니터링에 대한 보다 간단한 접근 방식을 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Network Services 계정이 없는 경우 계정을 생성합니다. 조직의 Network Services 계정에 허브를 배치합니다. 이 접근 방식을 사용하면 네트워크 엔지니어가 허브를 중앙에서 관리할 수 있습니다.

허브 앤 스포크 모델의 허브는 가상 프라이빗 클라우드(VPC)와 온프레미스 네트워크 간에 흐르는 트래픽에 대한 가상 라우터 역할을 합니다. 이 접근 방식은 네트워크 복잡성을 줄이고 네트워킹 문제를 더 쉽게 해결할 수 있도록 합니다.

상호 연결하려는 VPC, AWS Direct Connect 및 Site-to-Site VPN 연결을 포함하여 네트워크 설계를 고려합니다.

각 Transit Gateway VPC 연결에 대해 별도의 서브넷을 사용하는 것을 고려하세요. 서브넷별로 작은 CIDR(예: /28)을 사용하여 컴퓨팅 리소스를 위한 주소 공간을 더 많이 확보하세요. 또한 하나의 네트워크 ACL을 만들어 허브에 연결된 모든 서브넷과 연결합니다. 인바운드 및 아웃바운드 방향 모두에서 네트워크 ACL을 열어 둡니다.

통신해야 하는 네트워크 간에만 라우팅이 제공되도록 라우팅 테이블을 설계하고 구현합니다. 서로 통신해서는 안 되는 네트워크 간 경로를 생략합니다(예: 상호 종속성이 없는 별도의 워크로드 간).

구현 단계

1. 네트워크를 계획합니다. 연결할 네트워크를 결정하고 중복 CIDR 범위를 공유하지 않는지 확인합니다.

2. AWS Transit Gateway를 생성하고 VPC를 연결합니다.
3. 필요한 경우 VPN 연결 또는 Direct Connect 게이트웨이를 생성하여 Transit Gateway와 연결합니다.
4. 연결된 VPC와 기타 연결 간의 트래픽이 라우팅되는 방식을 Transit Gateway 라우팅 테이블의 구성을 통해 정의합니다.
5. 성능 및 비용 최적화를 위해 필요에 따라 구성을 모니터링하고 조정하는 데 Amazon CloudWatch를 사용합니다.

리소스

관련 모범 사례:

- REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인
- REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용

관련 문서:

- What Is a Transit Gateway?
- 전송 게이트웨이 설계 모범 사례
- 확장 가능하고 안전한 다중 VPC AWS 네트워크 인프라 구축
- Building a global network using AWS Transit Gateway Inter-Region peering
- Amazon Virtual Private Cloud 연결 옵션
- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Marketplace for Network Infrastructure

관련 비디오:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities

관련 워크숍:

- AWS Transit Gateway Workshop

REL02-BP05 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용

피어링되거나, Transit Gateway를 통해 연결되거나, VPN을 통해 연결된 경우 각 VPC의 IP 주소 범위가 중첩되지 않아야 합니다. VPC와 온프레미스 환경 간의 IP 주소 충돌 또는 사용하는 다른 클라우드 제공업체와의 IP 주소 충돌을 방지해야 합니다. 필요한 경우 프라이빗 IP 주소 범위를 할당할 수 있어야 합니다. IP 주소 관리(IPAM) 시스템이 이러한 작업을 자동화하는 데 도움이 될 수 있습니다.

원하는 성과:

- VPC, 온프레미스 환경 또는 기타 클라우드 제공업체 간의 IP 주소 범위 충돌이 없습니다.
- 적절한 IP 주소 관리를 통해 네트워크 요구 사항이 성장하고 변화함에 따라 네트워크 인프라를 쉽게 확장할 수 있습니다.

일반적인 안티 패턴:

- VPC에서 온프레미스, 기업 네트워크 또는 기타 클라우드 제공업체와 동일한 IP 범위를 사용합니다.
- 워크로드를 배포하는 데 사용되는 VPC의 IP 범위를 추적하지 않습니다.
- 스프레드시트와 같은 수동 IP 주소 관리 프로세스를 사용합니다.
- CIDR 블록의 크기를 너무 크거나 작게 지정하여 IP 주소가 낭비되거나 워크로드를 위한 주소 공간이 부족합니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Amazon VPC IP Address Manager와 같은 IPAM을 사용하여 CIDR 사용을 모니터링하고 관리합니다. AWS Marketplace에서도 여러 IPAM을 사용할 수 있습니다. AWS에서 잠재적인 사용량을 평가하고, 기존 VPC에 CIDR 범위를 추가하고, VPC를 생성하여 사용량을 계획에 맞춰 늘릴 수 있습니다.

구현 단계

- 현재 CIDR 소비량을 파악합니다(예: VPC, 서브넷 등).
  - 서비스 API를 사용하여 현재 CIDR 소비량을 파악합니다.

  - Amazon VPC IP 주소 관리자를 사용하여 리소스를 검색합니다.
- 현재 서브넷 사용량을 파악합니다.
  - 서비스 API 작업을 사용하여 각 리전의 VPC당 서브넷을 수집합니다.
  - Amazon VPC IP 주소 관리자를 사용하여 리소스를 검색합니다.
- 현재 사용량을 기록합니다.
- 중첩되지 않는 IP 범위를 생성했는지 판단합니다.
- 여유 용량을 계산합니다.
- 중첩된 IP 범위를 파악합니다. 새 주소 범위로 마이그레이션하거나 중첩되는 범위를 연결해야 하는 경우 프라이빗 NAT 게이트웨이 또는 AWS PrivateLink와 같은 기술 사용을 고려할 수 있습니다.

리소스

관련 모범 사례:

- 네트워크 보호

관련 문서:

- APN 파트너: 네트워킹 계획을 지원할 수 있는 파트너
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options 백서
- 다중 데이터 센터 HA 네트워크 연결
- Connecting Networks with Overlapping IP Ranges
- Amazon VPC란 무엇인가?
- IPAM이란?

관련 비디오:

- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs
- AWS re:Invent 2023 - Ready for what’s next? Designing networks for growth and flexibility
- AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS

워크로드 아키텍처

Questions

- REL 3. 워크로드 서비스 아키텍처는 어떻게 설계하나요?
- REL 4. 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계하나요?
- REL 5. 분산 시스템에서 장애를 완화하거나 견딜 수 있도록 상호 작용을 설계하려면 어떻게 해야 하나요?

REL 3. 워크로드 서비스 아키텍처는 어떻게 설계하나요?

서비스 지향 아키텍처(SOA) 또는 마이크로서비스 아키텍처를 사용하여 확장성과 신뢰성이 뛰어난 워크로드를 구축합니다. 서비스 지향 아키텍처(SOA)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 마이크로서비스로 나눌 수 있을지 신중하게 고려합니다. 가능한 경우 애플리케이션 요구 사항에 따라 서비스 지향 아키텍처(SOA)와 마이크로서비스의 결합으로 마무리될 수도 있습니다. 상태 비저장일 수 있는 워크로드는 마이크로서비스로 배포할 수 있습니다.

원하는 성과: 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 Star는 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 확립의 이점:

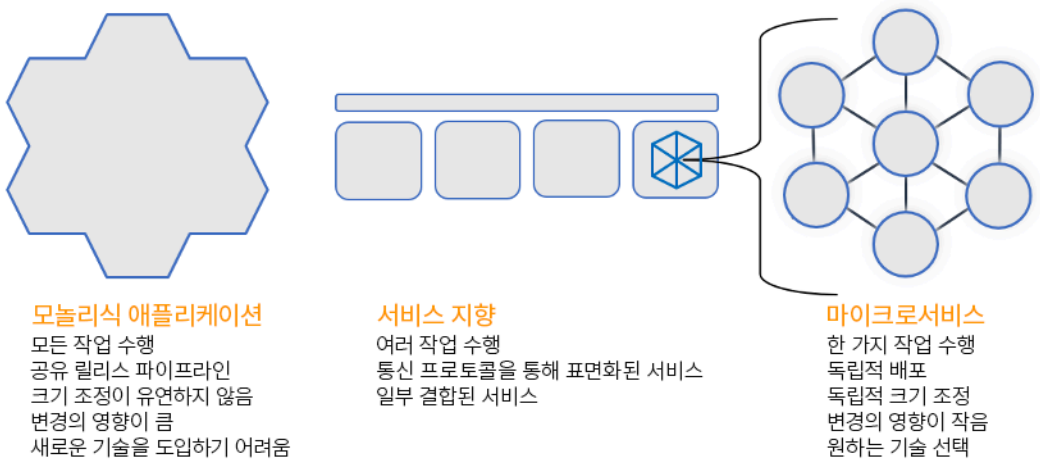
- 보다 구체적인 세그먼트를 사용하면 민첩성, 조직의 유연성 및 확장성이 향상됩니다.
- 서비스 중단의 영향이 감소합니다.
- 애플리케이션 구성 요소가 원자성이 더 큰 세그먼트화로 지원할 수 있는 여러 가능성 요구 사항을 가질 수 있습니다.
- 워크로드를 지원하는 팀의 책임이 잘 정의됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드를 분할하는 방법에 따라 아키텍처 유형을 선택합니다. 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 신뢰할 수 있는 아키텍처로 선호되지만 특히 마이크로서비스 아키텍처를 배포하는 경우 장단점을 고려해야 합니다.

한 가지 주요 장단점은 분산 컴퓨팅 아키텍처가 구축되었지만 사용자 지연 시간 요구 사항을 충족하기가 더 어려워지며, 사용자 상호 작용을 디버그하고 추적하는 과정이 더 복잡해진다는 점입니다. AWS X-Ray를 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 신뢰할 수 있는 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 신뢰성을 실현해야 합니다. 자세한 내용은 AWS에서 마이크로서비스 구현을 참조하세요.
- 모놀리식을 더 작은 구성 요소로 리팩터링하려면 Strangler Fig 패턴 준수를 고려하세요. 여기에는 특정 애플리케이션 구성 요소를 새 애플리케이션 및 서비스로 점차적으로 교체하는 작업이 포함됩니다. AWS Migration Hub Refactor Spaces는 증분 리팩터링의 시작점 역할을 합니다. 자세한 내용은 Seamlessly migrate on-premises legacy workloads using a strangler pattern을 참조하세요.
- 마이크로서비스를 구현하려면 이러한 분산된 서비스가 서로 통신하기 위한 서비스 검색 메커니즘이 필요할 수 있습니다. AWS App Mesh를 서비스 지향 아키텍처와 함께 사용하면 신뢰할 있는 서비스 검색 및 액세스를 제공할 수 있습니다. 동적 DNS 기반 서비스 검색에는 AWS Cloud Map도 사용할 수 있습니다.
- 모놀리식에서 SOA로 마이그레이션하는 경우 Amazon MQ는 클라우드에서 레거시 애플리케이션을 다시 설계할 때 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 프

로세스 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?
- Bounded Context(도메인 중심 설계의 중심 패턴)
- AWS에서 마이크로서비스 구현
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- AWS의 마이크로서비스
- AWS App Mesh란 무엇입니까?

관련 예제:

- Iterative App Modernization 워크숍

관련 비디오:

- Delivering Excellence with Microservices on AWS

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 성과: 엔지니어와 비즈니스 이해관계자가 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 이전됩니다. 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:

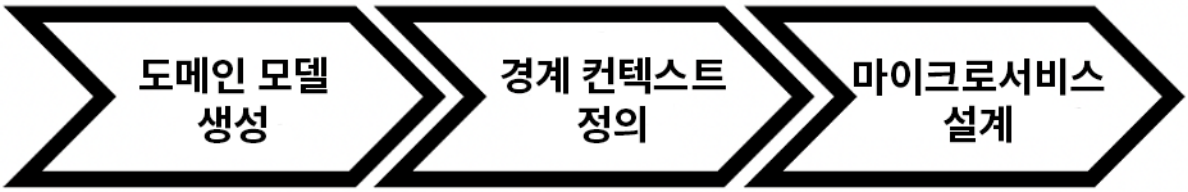
- 팀이 특정 비즈니스 도메인 대신 UI 및 UX, 미들웨어 또는 데이터베이스와 같은 특정 기술 도메인을 중심으로 구성됩니다.
- 애플리케이션이 여러 도메인 책임에 걸쳐 있습니다. 여러 경계 컨텍스트에 걸쳐 있는 서비스는 유지 관리가 더 어렵고 더 많은 테스트 작업이 필요하며 소프트웨어 업데이트에 여러 도메인 팀이 참여해야 할 수 있습니다.
- 도메인 엔터티 라이브러리와 같은 도메인 종속성은 서비스 간에 공유되므로 한 서비스 도메인을 변경하려면 다른 서비스 도메인도 변경해야 합니다.
- 서비스 계약과 비즈니스 로직은 엔터티를 공통적이고 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 않을 경우 노출되는 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 기반 설계

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 컨텍스트를 사용하여 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.
- Decomposing monoliths into microservices에서는 마이크로서비스 리팩터링 패턴을 설명합니다. 비즈니스 역량, 하위 도메인 또는 트랜잭션별 분해 패턴을 사용하는 것은 도메인 중심 접근 방식과 부합됩니다.
- 버블 컨텍스트와 같은 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 계층 또는 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 시작합니다. 테스트 기반 개발(TDD)과 동작 기반 개발(BDD)은 팀이 서비스가 비즈니스 문제 해결에 초점을 맞출 수 있도록 도와줍니다.
- 비즈니스 도메인 요구 사항 및 마이크로서비스 아키텍처를 가장 잘 충족하는 AWS 서비스를 선택합니다.

  - AWS 서버리스를 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.
- Building hexagonal architectures on AWS에서는 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
- How to break a Monolith into Microservices
- Getting Started with DDD when Surrounded by Legacy Systems
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Building hexagonal architectures on AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Test-driven development
- Behavior-driven development

관련 예제:

- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop)

관련 도구:

- AWS 클라우드 데이터베이스
- AWS의 서버리스
- AWS의 컨테이너

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 성과: 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.

일반적인 안티 패턴:

- 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 계약을 개발할 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 계약

은 API에 대한 명확한 자체 문서화 인터페이스를 제공하고 서로 다른 시스템과 프로그래밍 언어 간에 상호 운용성을 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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(Amazon API Gateway, AWS AppSync, Amazon EventBridge 등)를 아키텍처에 통합하여 애플리케이션에서 API 서비스 계약을 사용합니다. Amazon API Gateway를 사용하면 기본 AWS 서비스 및 기타 웹 서비스와 직접 통합할 수 있습니다. API Gateway는 OpenAPI 사양 및 버전 관리를 지원합니다. AWS AppSync는 관리형 GraphQL 엔드포인트로, 쿼리, 변형, 구독을 위한 서비스 인터페이스를 정의하는 GraphQL 스키마를 정의하여 구성하는 엔드포인트입니다. 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 정의를 가져오면 API 생성을 단순화하고 AWS Serverless Application Model 및 AWS 클라우드 개발 키트 (AWS CDK)와 같은 AWS의 코드형 인프라 도구와 통합될 수 있습니다.
  - API 정의를 내보래면 API 테스트 도구와의 통합을 단순화하고 서비스 소비자에게 통합 사양을 제공합니다.
- AWS AppSync로 GraphQL 스키마를 정의하여 GraphQL API를 정의하고 관리할 수 있습니다. 이를 통해 계약 인터페이스를 생성하고 복잡한 REST 모델, 여러 데이터베이스 테이블 또는 레거시 서비스와의 상호 작용을 단순화할 수 있습니다.
- AWS AppSync와 통합된 AWS Amplify 프로젝트는 애플리케이션에서 사용할 강력한 형식의 JavaScript 쿼리 파일과 Amazon DynamoDB 테이블용 AWS 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 Gateway를 사용하면 요청 및 응답 매핑을 통해 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 직접 호출 제어 및 제한
- REL05-BP05 클라이언트 제한 시간 설정

관련 문서:

- 애플리케이션 프로그래밍 인터페이스(API)란 무엇인가요?
- Implementing Microservices on AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- AWS의 마이크로서비스
- OpenAPI에 대한 API Gateway 확장 작업
- OpenAPI-Specification
- GraphQL: Schemas and Types

- Amazon EventBridge code bindings

관련 예제:

- Amazon API Gateway: OpenAPI를 사용하여 REST API 구성
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI
- Modern application integration patterns in a serverless age: API Gateway Service Integration
- Implementing header-based API Gateway versioning with Amazon CloudFront
- AWS AppSync: Building a client application

관련 비디오:

- Using OpenAPI in AWS SAM to manage API Gateway

관련 도구:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

REL 4. 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계하나요?

분산 시스템은 통신 네트워크를 사용하여 서버 또는 서비스와 같은 구성 요소를 상호 연결합니다. 이러한 네트워크에서 데이터 손실이나 지연 시간이 발생하더라도 워크로드는 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 이러한 모범 사례는 장애를 예방하고 평균 고장 간격(MTBF)을 개선합니다.

모범 사례

- REL04-BP01 사용 중인 분산 시스템의 종류 파악
- REL04-BP02 느슨하게 결합된 종속성 구현
- REL04-BP03 일정한 작업 처리
- REL04-BP04 변경 작업에 멱등성 부여

REL04-BP01 사용 중인 분산 시스템의 종류 파악

분산 시스템은 동기, 비동기 또는 배치 방식일 수 있습니다. 동기 시스템은 HTTP/S, REST 또는 원격 프로시저 직접 호출(RPC) 프로토콜을 사용하여 동기식 요청 및 응답 직접 호출을 수행함으로써 요청을 최대한 빨리 처리하고 서로 통신해야 합니다. 비동기 시스템은 개별 시스템을 결합하지 않고 중개 서비스를 통해 비동기식으로 데이터를 교환하여 서로 통신합니다. 배치 시스템은 대량의 입력 데이터를 수신하고, 사람의 개입 없이 자동화된 데이터 프로세스를 실행하며, 출력 데이터를 생성합니다.

원하는 성과: 동기, 비동기 및 배치 종속성과 효과적으로 상호 작용하는 워크로드를 설계합니다.

일반적인 안티 패턴:

- 워크로드가 종속성의 응답에 대해 무기한 대기하므로 요청이 수신되었는지 알지 못한 채로 워크로드 클라이언트가 제한 시간을 초과할 수 있습니다.
- 워크로드가 서로를 동기식으로 호출하는 종속 시스템 체인을 사용합니다. 이를 위해서는 전체 체인이 성공하기 전에 각 시스템을 사용할 수 있어야 하고 요청을 성공적으로 처리해야 합니다. 이로 인해 동작과 전체적인 가용성이 불안정질 수 있습니다.
- 워크로드가 종속성과 비동기적으로 통신하며, 중복 메시지를 수신할 수 있는 경우가 많지만 정확히 한 번 보장된 메시지 전달이라는 개념을 사용합니다.
- 워크로드가 적절한 배치 일정 예약 도구를 사용하지 않으며 동일한 배치 작업을 동시에 실행하도록 허용합니다.

이 모범 사례 확립의 이점: 특정 워크로드에서 동기, 비동기, 배치 중에 하나 이상의 통신 스타일을 구현하는 것이 일반적입니다. 이 모범 사례는 각 통신 스타일과 관련된 다양한 장단점을 식별하여 종속성에 중단이 발생했을 때 워크로드가 견딜 수 있도록 하는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

다음 섹션에는 각 종속성 구현에 대한 일반적인 지침과 구체적인 구현 지침이 모두 포함되어 있습니다.

일반 지침

- 종속성이 제공하는 성능 및 신뢰성 서비스 수준 목표(SLO)가 워크로드의 성능 및 신뢰성 요구 사항을 충족하는지 확인하세요.
- AWS 관찰성 서비스로 응답 시간과 오류율을 모니터링하여 종속성이 워크로드에 필요한 수준의 서비스를 제공하고 있는지 확인하세요.

- 워크로드가 종속성과 통신할 때 직면할 수 있는 잠재적 문제를 식별하세요. 분산 시스템에는 아키텍처 복잡성, 운영 부담 및 비용을 증가시킬 수 있는 다양한 문제가 있습니다. 일반적인 문제로는 지연 시간, 네트워크 장애, 데이터 손실, 규모 조정, 데이터 복제 지연 등이 있습니다.
- 강력한 오류 처리 및 로깅을 구현하면 종속성 문제가 발생할 때 문제를 해결하는 데 도움이 됩니다.

동기식 종속성

동기식 통신에서는 워크로드가 종속성에 요청을 보내고 응답을 기다리는 작업을 차단합니다. 종속성은 요청을 수신하면 최대한 바로 처리하려고 시도하고 워크로드에 응답을 다시 보냅니다. 동기식 통신의 중요한 문제는 일시적 결합이 발생하여 워크로드와 해당 종속성을 동시에 사용할 수 있어야 한다는 것입니다. 워크로드가 종속성과 동기적으로 통신해야 하는 경우 다음 지침을 고려하세요.

- 워크로드가 단일 기능을 수행하기 위해 다수의 동기 종속성에 의존해서는 안 됩니다. 이렇게 여러 종속성이 체인을 이루면 요청을 성공적으로 완료하기 위해 경로상의 모든 종속성을 사용할 수 있어야 하기 때문에 전반적인 불안정성이 높아집니다.
- 종속성이 비정상이거나 사용할 수 없는 경우 오류 처리 및 재시도 전략을 결정하세요. 바이모달 동작은 사용하지 마세요. 바이모달 동작은 정상 모드와 장애 모드에서 워크로드가 서로 다른 동작을 보일 때를 말합니다. 바이모달 동작에 대한 자세한 내용은 REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지를 참조하세요.
- 빠른 실패가 워크로드를 기다리게 하는 것보다 낫다는 점을 명심하세요. 예를 들어, AWS Lambda 개발자 안내서에서는 Lambda 함수를 간접 호출할 때 재시도 및 실패를 처리하는 방법을 설명합니다.
- 워크로드가 종속성을 직접 호출할 때 제한 시간을 설정하세요. 이 기법을 사용하면 응답을 너무 오래 기다리거나 무한정 기다리지 않아도 됩니다. 이 주제에 대한 유용한 논의는 Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications를 참조하세요.
- 단일 요청을 처리하기 위해 워크로드에서 종속성에 수행하는 직접 호출 수를 최소화하세요. 둘 사이에 직접 호출이 너무 많으면 결합과 지연 시간이 늘어납니다.

비동기식 종속성

워크로드를 종속성에서 일시적으로 분리하려면 이 둘이 비동기적으로 통신해야 합니다. 비동기식 접근 방식을 사용하면 종속성 또는 종속성 체인이 응답을 보낼 때까지 기다릴 필요 없이 워크로드가 다른 처리를 계속 진행할 수 있습니다.

워크로드가 종속성과 비동기적으로 통신해야 하는 경우 다음 지침을 고려하세요.

- 사용 사례와 요구 사항에 따라 메시징을 사용할지 아니면 이벤트 스트리밍을 사용할지 결정하세요. 메시징을 사용하면 메시지 브로커를 통해 메시지를 주고받는 방식으로 워크로드가 종속 항목과 통신할 수 있습니다. 이벤트 스트리밍을 사용하면 워크로드와 해당 종속성이 스트리밍 서비스를 사용하여 가능한 한 빨리 처리해야 하는 연속 데이터 스트림으로 전달되는 이벤트를 게시하고 구독할 수 있습니다.

- 메시징과 이벤트 스트리밍은 메시지를 다르게 처리하므로 다음을 기준으로 장단점을 고려하여 결정해야 합니다.
  - 메시지 우선순위: 메시지 브로커는 우선순위가 높은 메시지를 일반 메시지보다 먼저 처리할 수 있습니다. 이벤트 스트리밍에서는 모든 메시지의 우선순위가 동일합니다.
  - 메시지 소비: 메시지 브로커는 소비자가 메시지를 받을 수 있도록 보장합니다. 이벤트 스트리밍 소비자는 마지막으로 읽은 메시지를 추적해야 합니다.
  - 메시지 정렬: 메시징의 경우 선입선출(FIFO) 방식을 사용하지 않는 한 메시지를 전송된 순서대로 정확하게 수신하지 못합니다. 이벤트 스트리밍은 항상 데이터가 생성된 순서를 지킵니다.
  - 메시지 삭제: 메시징의 경우 소비자가 메시지를 처리한 후 메시지를 삭제해야 합니다. 이벤트 스트리밍 서비스는 메시지를 스트림에 추가하고 메시지 보존 기간이 만료될 때까지 메시지가 스트림에 남아 있습니다. 이 삭제 정책 때문에 이벤트 스트리밍은 메시지 재생에 적합합니다.
- 종속성이 작업을 완료하는 시점을 워크로드가 어떻게 인식하는지 정의하세요. 예를 들어, 워크로드가 Lambda 함수를 비동기식으로 간접 호출하면 Lambda는 이벤트를 대기열에 배치하고 추가 정보 없이 성공 응답을 반환합니다. 처리가 완료되면 Lambda 함수는 결과를 대상으로 전송할 수 있으며, 성공 또는 실패에 따라 구성할 수 있습니다.
- 멱등성을 활용하여 중복 메시지를 처리하도록 워크로드를 구축하세요. 멱등성이란 동일한 메시지에 대해 워크로드가 두 번 이상 생성되더라도 워크로드의 결과가 변경되지 않는 것을 의미합니다. 네트워크 장애가 발생하거나 확인이 수신되지 않은 경우 메시징 또는 스트리밍 서비스가 메시지를 다시 전송한다는 점을 명심합니다.
- 워크로드가 종속성에서 응답을 받지 못하는 경우 요청을 다시 제출해야 합니다. 다른 요청을 처리할 수 있도록 워크로드의 CPU, 메모리 및 네트워크 리소스를 보존하려면 재시도 횟수를 제한하는 것이 좋습니다. AWS Lambda 설명서에는 비동기 호출의 오류를 처리하는 방법이 나와 있습니다.
- 적절한 관찰성, 디버깅, 추적 도구를 활용하여 워크로드와 종속성의 비동기식 통신을 관리하고 운영하세요. Amazon CloudWatch를 사용하여 메시징 및 이벤트 스트리밍 서비스를 모니터링할 수 있습니다. 또한 AWS X-Ray로 워크로드를 계측하여 문제 해결에 필요한 인사이트를 빠르게 얻을 수 있습니다.

배치 종속성

배치 시스템은 입력 데이터를 가져와 일련의 작업을 시작하고 수동 개입 없이 일부 출력 데이터를 생성합니다. 데이터 크기에 따라 작업 실행이 몇 분에서 경우에 따라 며칠까지 지속될 수 있습니다. 워크로드가 배치 종속성과 통신할 때는 다음 지침을 고려하세요.

- 워크로드에서 배치 작업을 실행해야 하는 기간을 정의하세요. 워크로드에서 반복 패턴을 설정하여 배치 시스템을 간접 호출할 수 있습니다(예: 매시간 또는 매월 말).
- 데이터 입력 및 처리된 데이터 출력의 위치를 결정합니다. 대규모로 워크로드에서 파일을 읽고 쓸 수 있는 Amazon Simple Storage Services(S3), Amazon Elastic File System(Amazon EFS), Amazon FSx for Lustre 중에서 스토리지 서비스를 선택합니다.
- 워크로드에서 여러 배치 작업을 간접 호출해야 하는 경우 AWS Step Functions를 활용하여 AWS 또는 온프레미스에서 실행되는 배치 작업의 오케스트레이션을 간소화할 수 있습니다. 이 샘플 프로젝트에서는 Step Functions, AWS Batch, Lambda를 사용한 배치 작업의 오케스트레이션을 보여줍니다.
- 배치 작업을 모니터링하여 완료하는 데 예상 외로 오래 걸리는 작업과 같은 이상 현상이 없는지 확인합니다. CloudWatch Container Insights와 같은 도구를 사용하여 AWS Batch 환경 및 작업을 모니터링할 수 있습니다. 이 경우 워크로드는 다음 작업이 시작되지 못하도록 멈추고 관련 직원에게 예외를 알립니다.

리소스

관련 문서:

- AWS 클라우드 운영: 모니터링 및 관찰성
- Amazon Builders' Library: 분산 시스템의 도전 과제
- REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지
- AWS Lambda 개발자 안내서: AWS Lambda의 오류 처리 및 자동 재시도
- Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications
- AWS 메시징
- 스트리밍 데이터란 무엇인가요?
- AWS Lambda 개발자 안내서: 비동기 호출
- Amazon Simple Queue Service FAQ: FIFO 대기열
- Amazon Kinesis Data Streams 개발자 안내서: Handling Duplicate Records
- Amazon Simple Queue Service 개발자 안내서: Available CloudWatch metrics for Amazon SQS

- Amazon Kinesis Data Streams 개발자 안내서: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch
- AWS X-Ray 개발자 안내서: AWS X-Ray concepts
- AWS Samples on GitHub: AWS Step functions Complex Orchestrator App
- AWS Batch 사용 설명서: AWS Batch CloudWatch Container Insights

관련 비디오:

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

관련 도구:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service(S3)
- Amazon Elastic File System(Amazon EFS)
- Amazon FSx for Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 느슨하게 결합된 종속성 구현

대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 느슨히 결합됩니다. 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.

대기열 처리 시스템, 스트리밍 시스템, 워크플로와 같은 종속성을 분리하면 시스템에서 변경 또는 장애의 영향을 최소화하는 데 도움이 됩니다. 이러한 분리를 통해 구성 요소의 동작이 이와 종속된 다른 구성 요소에 영향을 주지 않으므로 복원력과 민첩성이 향상됩니다.

강한 결합으로 구성된 시스템에서는 한 구성 요소를 변경하면 해당 구성 요소를 사용하는 다른 구성 요소도 변경해야 할 수 있으므로 결과적으로 모든 구성 요소의 성능이 저하될 수 있습니다. 느슨한 결합에서는 이 종속성이 분리되므로 종속 구성 요소에서는 버전이 지정되고 게시된 인터페이스만 알면 됩

니다. 종속성 간에 느스한 결합을 구현하면 한 구성 요소의 장애가 다른 구성 요소에 영향을 미치지 않도록 분리됩니다.

느슨한 결합을 사용하면 다른 종속 구성 요소에 미치는 위험을 최소화하면서 구성 요소의 코드를 수정하거나 기능을 추가할 수 있습니다. 또한 구성 요소 수준에서 세분화된 복원력을 지원하므로 종속성의 기본 구현을 스케일 아웃하거나 변경할 수도 있습니다.

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 않고 일반적으로 내구성이 있는 중간 스토리지 계층(예: Amazon SQS 대기열, 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 대기열 및 AWS Step Functions는 느슨한 결합을 위한 중간 계층을 추가할 수 있는 두 가지 방법의 예입니다. AWS 클라우드에서는 Amazon EventBridge를 사용하여 이벤트 기반 아키텍처를 구축할 수도 있습니다. 그러면 클라이언트가 의존하는 서비스(이벤트 소비자)에서 클라이언트(이벤트 생산자)를 추상화할 수 있습니다. Amazon Simple Notification Service(SNS)는 높은 처리량의 푸시 기반 다대다 메시징이 필요할 때 효과적인 솔루션입니다. Amazon SNS 주제를 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.

원하는 성과: 느슨하게 결합된 종속성을 구현하면 장애 노출 영역을 구성 요소 수준으로 최소화할 수 있으므로 문제를 진단하고 해결하는 데 도움이 됩니다. 또한 개발 주기를 간소화하여 팀이 이를 사용하는 다른 구성 요소의 성능에 영향을 주지 않고 모듈 수준에서 변경 사항을 구현할 수 있습니다. 이 접근 방식은 리소스 요구 사항 및 구성 요소 활용도를 기반으로 구성 요소 수준에서 스케일 아웃할 수 있는 기능을 제공하여 비용 효율성에 기여합니다.

일반적인 안티 패턴:

- 모놀리식 워크로드를 배포합니다.
- 장애 조치 또는 비동기식 요청 처리 기능 없이 워크로드 티어 사이에서 API 직접 호출.

- 공유 데이터를 사용하는 강한 결합. 느슨하게 결합된 시스템은 공유 데이터베이스나 다른 형태의 밀결합된 데이터 스토리지를 통한 데이터 공유를 피해야 합니다. 그러지 않으면 밀결합이 다시 도입되어 확장성이 저해될 수 있습니다.
- 배압을 무시합니다. 워크로드는 구성 요소가 동일한 속도로 데이터를 처리할 수 없을 때 수신 데이터의 속도를 늦추거나 중지할 수 있어야 합니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

느슨하게 결합된 종속성을 구현합니다. 느슨하게 결합된 애플리케이션을 구축할 수 있는 솔루션은 다양합니다. 여기에는 완전관리형 대기열, 자동화된 워크플로, 이벤트 대응, API 등을 구현하기 위한 서비스가 포함됩니다. 이러한 서비스는 구성 요소의 동작을 다른 구성 요소로부터 분리하고 복원력과 민첩성을 높이는 데 도움이 될 수 있습니다.

- 이벤트 기반 아키텍처 구축: Amazon EventBridge를 사용하면 느슨하게 결합되고 분산된 이벤트 기반 아키텍처를 구축할 수 있습니다.
- 분산 시스템에서 대기열 구현: Amazon Simple Queue Service(Amazon SQS)를 사용하여 분산 시스템을 통합하고 분리할 수 있습니다.
- 구성 요소를 마이크로서비스로 컨테이너화: 마이크로서비스를 사용하면 잘 정의된 API를 통해 통신하는 독립적인 소규모 구성 요소로 구성된 애플리케이션을 구축할 수 있습니다. Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS)는 컨테이너를 빠르게 시작하는 데 도움을 줄 수 있습니다.
- Step Functions로 워크플로 관리: Step Functions는 여러 AWS 서비스를 유연한 워크플로로 조율할 수 있도록 도와줍니다.
- 게시 및 구독(pub/sub) 메시징 아키텍처 활용: Amazon Simple Notification Service(SNS)는 게시자(생산자)가 구독자(소비자)에게 메시지를 전달하도록 합니다.

구현 단계

- 이벤트 기반 아키텍처의 구성 요소는 이벤트로 인해 시작됩니다. 이벤트는 사용자가 장바구니에 품목을 추가하는 것과 같이 시스템에서 발생하는 작업입니다. 작업이 성공하면 시스템의 다음 구성 요소를 작동시키는 이벤트가 생성됩니다.

  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge
- 분산 메시징 시스템에는 대기열 기반 아키텍처를 위해 구현해야 하는 세 가지 주요 부분이 있습니다. 분산 시스템의 구성 요소, 분리에 사용되는 대기열(Amazon SQS 서버에 분산됨), 대기열의 메시지가 포함됩니다. 일반적인 시스템에는 메시지를 대기열로 보내는 생산자와 대기열에서 메시지를 수신하는 소비자가 있습니다. 대기열은 중복성을 위해 여러 Amazon SQS 서버에 메시지를 저장합니다.
  - Basic Amazon SQS architecture
  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- 마이크로서비스를 잘 활용하면 개별 팀이 느슨하게 결합된 구성 요소를 관리하므로 유지 관리 가능성과 확장성이 향상됩니다. 또한 변경 시 동작을 단일 구성 요소로 분리할 수 있습니다.
  - Implementing Microservices on AWS
  - Let's Architect! Architecting microservices with containers
- AWS Step Functions를 사용하면 분산 애플리케이션을 구축하고, 프로세스를 자동화하며, 마이크로서비스를 오케스트레이션하는 등의 작업을 수행할 수 있습니다. 여러 구성 요소를 자동화된 워크플로로 오케스트레이션하면 애플리케이션의 종속성을 분리할 수 있습니다.
  - Create a Serverless Workflow with AWS Step Functions and AWS Lambda
  - AWS Step Functions 시작하기

리소스

관련 문서:

- Amazon EC2: Ensuring Idempotency
- Amazon Builders' Library: 분산 시스템의 도전 과제
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- Amazon EventBridge란 무엇인가요?
- What Is Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Queue-Based Architecture

관련 비디오:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

REL04-BP03 일정한 작업 처리

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 발생한 서버가 없든 모든 서버에서 장애가 발생하든, 상태 확인 시스템은 대규모의 급속한 변경이 없는 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 대신 상태 확인 시스템에서 매번 현재 상태의 전체 스냅샷을 전송해야 합니다. 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 처리하여 최종 사용자가 어떻게 엔드포인트에 라우팅되는지 판단합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

- 대규모 로드가 급속도로 변경될 때 시스템에서 장애가 발생하지 않도록 일정한 작업을 처리합니다.
- 느슨하게 결합된 종속성을 구현합니다. 대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 느슨히 결합됩니다. 느슨한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.

- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
  - 100,000개의 서버를 모니터링하는 상태 확인 시스템의 예제에서 성공 또는 실패 횟수와 관계없이 페이로드 크기가 일정하게 유지되도록 워크로드를 엔지니어링합니다.

리소스

관련 문서:

- Amazon EC2: Ensuring Idempotency
- Amazon Builders' Library: 분산 시스템의 도전 과제
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee

관련 비디오:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 변경 작업에 멱등성 부여

멱등성이 있는 서비스는 각 요청이 정확히 한 번만 처리되도록 합니다. 이렇게 하면 다수의 동일한 요청에서 단일 요청과 동일한 결과가 나옵니다. 이렇게 하면 클라이언트가 요청이 오류로 여러 번 처리된다는 염려 없이 재시도를 시행할 수 있습니다. 이를 위해 클라이언트는 멱등성 토큰을 사용하여 API 요청을 실행할 수 있으며 요청이 반복될 때마다 이 토큰이 사용됩니다. 멱등성이 있는 서비스 API는 시스템의 기본 상태가 변경되더라도 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) 수행하기가 상대적으로 간단합니다. 작업이 정확히 한 번 수행되도록

하여 다수의 동일한 요청에서 단일 요청과 동일한 결과를 얻기는 더 어렵습니다. API에서 멱등성 토큰을 사용하면 서비스가 중복 레코드를 생성할 필요가 없고 부작용 없이 변경 요청을 한 번 이상 수신할 수 있습니다.

원하는 성과: 모든 구성 요소 및 서비스에서 멱등성을 보장하기 위해 일관되고 잘 문서화되고 널리 채택된 접근 방식을 사용합니다.

일반적인 안티 패턴:

- 필요하지 않은 경우에도 무차별적으로 멱등성을 적용합니다.
- 멱등성을 구현하기 위한 지나치게 복잡한 로직을 도입합니다.
- 타임스탬프를 멱등성의 키로 사용합니다. 이렇게 하면 클럭 스큐 또는 동일한 타임스탬프를 사용하여 변경 사항을 적용하는 여러 클라이언트로 인해 부정확해질 수 있습니다.
- 멱등성을 위해 전체 페이로드를 저장합니다. 이 접근 방식에서는 모든 요청에 대한 전체 데이터 페이로드를 저장하고 새 요청마다 덮어씁니다. 이로 인해 성능이 저하되고 확장성에 영향을 미칠 수 있습니다.
- 서비스 간에 키를 일관되지 않게 생성합니다. 일관된 키가 없으면 서비스가 중복 요청을 인식하지 못하여 의도하지 않은 결과가 발생할 수 있습니다.

이 모범 사례 확립의 이점:

- 확장성 향상: 시스템은 추가 로직 또는 복잡한 상태 관리를 수행할 필요 없이 재시도 및 중복 요청을 처리할 수 있습니다.
- 향상된 신뢰성: 멱등성은 서비스가 일관된 방식으로 여러 동일한 요청을 처리하도록 지원하므로 의도하지 않은 부작용 또는 중복 레코드의 위험이 줄어듭니다. 이는 네트워크 장애 및 재시도가 일반적인 분산 시스템에서 특히 중요합니다.
- 데이터 일관성 개선: 동일한 요청이 동일한 응답을 생성하기 때문에 멱등성은 분산 시스템에서 데이터 일관성을 유지하는 데 도움이 됩니다. 이는 트랜잭션 및 작업의 무결성을 유지하는 데 필수적입니다.
- 오류 처리: 멱등성 토큰을 사용하면 오류 처리가 더 간단해집니다. 클라이언트가 문제로 인해 응답을 받지 못하는 경우 동일한 멱등성 토큰으로 요청을 안전하게 다시 보낼 수 있습니다.
- 운영 투명성: 멱등성을 통해 모니터링 및 로깅을 개선할 수 있습니다. 서비스는 멱등성 토큰으로 요청을 로깅할 수 있으므로 문제를 더 쉽게 추적하고 디버깅할 수 있습니다.
- 간소화된 API 계약: 클라이언트와 서버 측 시스템 간의 계약을 간소화하고 잘못된 데이터 처리에 대한 두려움을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

분산 시스템에서 작업을 최대 한 번(클라이언트가 한 번만 요청) 또는 최소 한 번(클라이언트가 성공이 확인될 때까지 계속 요청) 수행하기가 상대적으로 간단합니다. 그러나 정확히 한 번 동작을 구현하기는 어렵습니다. 이를 위해 클라이언트는 각 요청에 대한 멱등성 토큰을 생성하고 제공해야 합니다.

멱등성 토큰을 사용하면 서비스가 새 요청과 반복된 요청을 구분할 수 있습니다. 서비스가 멱등성 토큰이 포함된 요청을 수신하면 토큰이 이미 사용되었는지 확인합니다. 토큰이 사용된 경우 서비스는 저장된 응답을 검색하고 반환합니다. 새 토큰인 경우 서비스는 요청을 처리하고 토큰과 함께 응답을 저장한 다음, 응답을 반환합니다. 이 메커니즘은 모든 응답에 멱등성을 부여하여 분산 시스템의 신뢰성과 일관성을 향상시킵니다.

멱등성은 이벤트 기반 아키텍처의 중요한 동작이기도 합니다. 이러한 아키텍처는 일반적으로 Amazon SQS, Amazon MQ, Amazon Kinesis Streams 또는 Amazon Managed Streaming for Apache Kafka(Amazon MSK)와 같은 메시지 대기열에 의해 지원됩니다. 경우에 따라 한 번만 게시된 메시지가 실수로 두 번 이상 전달될 수 있습니다. 게시자가 메시지에 멱등성 토큰을 생성하고 포함할 때 수신된 중복 메시지를 처리해도 동일한 메시지에 대해 반복 작업이 발생하지 않도록 요청합니다. 소비자는 수신된 각 토큰을 추적하고 중복 토큰이 포함된 메시지를 무시해야 합니다.

또한 서비스와 소비자는 수신된 멱등성 토큰이 호출하는 다운스트림 서비스에 멱등성 토큰을 전달해야 합니다. 처리 체인의 모든 다운스트림 서비스는 메시지를 두 번 이상 처리할 때의 부작용을 방지하기 위해 멱등성을 구현해야 할 책임이 있습니다.

구현 단계

1. 멱등성이 있는 작업 식별

   어떤 작업에 멱등성이 필요한지 결정합니다. 여기에는 일반적으로 POST, PUT 및 DELETE HTTP 메서드와 데이터베이스 삽입, 업데이트 또는 삭제 작업이 포함됩니다. 읽기 전용 쿼리와 같이 상태를 변경하지 않는 작업은 부작용이 없는 한 일반적으로 멱등성이 필요하지 않습니다.

2. 고유한 식별자 사용

   발신자가 보낸 각 멱등성이 있는 작업 요청에 고유한 토큰을 포함합니다. 요청에 직접 포함하거나 메타데이터(예: HTTP 헤더)의 일부로 포함하면 됩니다. 이렇게 하면 수신자가 중복된 요청 또는 작업을 인식하고 처리할 수 있습니다. 토큰에 일반적으로 사용되는 식별자에는 Universally Unique Identifiers(UUID) 및 K-Sortable Unique Identifiers(KSUID)가 있습니다.

3. 상태 추적 및 관리

워크로드에서 각 작업 또는 요청의 상태를 유지 관리합니다. 이는 데이터베이스, 캐시 또는 기타 영구 스토어에 멱등성 토큰과 해당 상태(예: 보류, 완료 또는 실패)를 저장하여 달성할 수 있습니다. 이 상태 정보를 통해 워크로드는 중복 요청 또는 작업을 식별하고 처리할 수 있습니다.

잠금, 트랜잭션 또는 낙관적 동시성 제어와 같이 필요한 경우 적절한 동시성 제어 메커니즘을 사용하여 일관성과 원자성을 유지합니다. 여기에는 멱등성 토큰을 기록하고 요청 서비스와 관련된 모든 변경 작업을 실행하는 프로세스가 포함됩니다. 이렇게 하면 레이스 조건을 방지하고 멱등성이 있는 작업이 올바르게 실행되는지 확인할 수 있습니다.

스토리지 및 성능을 관리하기 위해 데이터 저장소에서 오래된 멱등성 토큰을 정기적으로 제거합니다. 스토리지 시스템이 지원하는 경우 데이터에 만료 타임스탬프(종종 Time To Live, 즉 TTL 값이라고 함)를 사용하는 것이 좋습니다. 멱등성 토큰 재사용 가능성은 시간이 지남에 따라 줄어듭니다.

일반적으로 멱등성 토큰 및 관련 상태를 저장하는 데 흔히 사용되는 AWS 스토리지 옵션은 다음과 같습니다.

- Amazon DynamoDB: DynamoDB는 지연 시간이 짧은 성능과 고가용성을 제공하는 NoSQL 데이터베이스 서비스이므로 멱등성 관련 데이터의 스토리지로 적합합니다. DynamoDB의 키-값 및 문서 데이터 모델을 사용하면 멱등성 토큰 및 관련 상태 정보를 효율적으로 저장하고 검색할 수 있습니다. DynamoDB는 애플리케이션이 멱등성 토큰을 삽입할 때 TTL 값을 설정하는 경우 멱등성 토큰을 자동으로 만료시킬 수도 있습니다.
- Amazon ElastiCache: ElastiCache는 처리량이 높고 지연 시간이 짧으며 비용이 저렴한 멱등성 토큰을 저장할 수 있습니다. 애플리케이션이 멱등성 토큰을 삽입할 때 TTL 값을 설정하는 경우 ElastiCache(Redis)와 ElastiCache(Memcached) 모두 멱등성 토큰을 자동으로 만료시킬 수 있습니다.
- Amazon Relational Database Service(Amazon RDS): Amazon RDS를 사용하여 멱등성 토큰 및 관련 상태 정보를 저장할 수 있습니다. 특히 애플리케이션이 이미 다른 목적으로 관계형 데이터베이스를 사용하는 경우 더욱 그렇습니다.
- Amazon Simple Storage Service(Amazon S3): Amazon S3는 확장성과 내구성이 뛰어난 객체 스토리지 서비스로, 멱등성 토큰 및 관련 메타데이터를 저장하는 데 사용할 수 있습니다. S3의 버전 관리 기능은 멱등성이 있는 작업 상태를 유지 관리하는 데 특히 유용할 수 있습니다. 스토리지 서비스의 선택은 일반적으로 멱등성 관련 데이터의 볼륨, 필요한 성능 특성, 내구성 및 가용성에 대한 요구, 멱등성 메커니즘이 전체 워크로드 아키텍처와 통합되는 방식과 같은 요인에 따라 달라집니다.

4. 멱등성이 있는 작업 구현

API 및 워크로드 구성 요소가 멱등성이 있도록 설계합니다. 워크로드 구성 요소에 멱등성 검사를 통합합니다. 요청을 처리하거나 작업을 수행하기 전에 고유 식별자가 이미 처리되었는지 확인합니다. 이미 처리된 경우 작업을 다시 실행하는 대신 이전 결과를 반환합니다. 예를 들어 클라이언트가 사용자를 생성하라는 요청을 보내는 경우 동일한 고유 식별자를 가진 사용자가 이미 존재하는지 확인합니다. 사용자가 있는 경우 새 사용자 정보를 생성하는 대신 기존 사용자 정보를 반환해야 합니다. 마찬가지로 대기열 소비자가 중복된 멱등성 토큰이 포함된 메시지를 받는 경우 소비자는 메시지를 무시해야 합니다.

요청의 멱등성을 검증하는 포괄적인 테스트 제품군을 생성합니다. 성공적인 요청, 실패한 요청 및 중복 요청과 같은 다양한 시나리오를 포함해야 합니다.

워크로드가 AWS Lambda 함수를 활용하는 경우 Powertools for AWS Lambda를 고려하세요. Powertools for AWS Lambda는 AWS Lambda 함수를 사용할 때 서버리스 모범 사례를 구현하고 개발자 속도를 높이기 위한 개발자 도구 모음입니다. 특히 Lambda 함수를 재시도해도 안전한 멱등성 작업으로 변환하는 유틸리티를 제공합니다.

5. 멱등성을 명확하게 전달

   API 및 워크로드 구성 요소를 문서화하여 작업의 멱등성을 명확하게 전달합니다. 이를 통해 클라이언트는 기대되는 동작 및 워크로드와 안정적으로 상호 작용하는 방법을 이해할 수 있습니다.

6. 모니터링 및 감사

   모니터링 및 감사 메커니즘을 구현하여 예상치 못한 응답 변경 또는 과도한 중복 요청 처리와 같이 응답의 멱등성과 관련된 문제를 탐지합니다. 이렇게 하면 워크로드의 문제 또는 예상치 못한 동작을 감지하고 조사하는 데 도움이 될 수 있습니다.

리소스

관련 모범 사례:

- REL05-BP03 재시도 직접 호출 제어 및 제한
- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP03 알림 전송(실시간 처리 및 경보)
- REL08-BP02 배포의 일부로 기능 테스트 통합

관련 문서:

- The Amazon Builders' Library: Making retries safe with idempotent APIs

- Amazon Builders' Library: 분산 시스템의 도전 과제
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- Amazon Elastic Container Service: Ensuring idempotency
- How do I make my Lambda function idempotent?
- Ensuring idempotency in Amazon EC2 API requests

관련 비디오:

- Building Distributed Applications with Event-driven Architecture - AWS Online Tech Talks
- AWS re:Invent 2,023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2,023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2,023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2,018 - Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2,019 - Moving to event-driven architectures (SVS308)

관련 도구:

- Idempotency with AWS Lambda Powertools (Java)
- Idempotency with AWS Lambda Powertools (Python)
- AWS Lambda Powertools GitHub 페이지

REL 5. 분산 시스템에서 장애를 완화하거나 견딜 수 있도록 상호 작용을 설계하려면 어떻게 해야 하나요?

분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 신뢰할 수 있는 상태로 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 평균 복구 시간(MTTR)이 개선됩니다.

모범 사례

- REL05-BP01 관련 하드 종속성을 소프트 종속성으로 변환하는 단계적 성능 저하 구현

- REL05-BP02 요청 제한
- REL05-BP03 재시도 직접 호출 제어 및 제한
- REL05-BP04 빠른 실패 및 대기열 제한
- REL05-BP05 클라이언트 제한 시간 설정
- REL05-BP06 가능한 경우 시스템을 상태 비저장으로 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 성과: 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 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 직접 호출자 또는 고객에게 제공할 수 있는 방법을 마련합니다. 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. 예시:

- 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 직접 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 추가적인 직접 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.

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 예제 참조).
2. 다운스트림 시스템이 높은 부하로 인해 요청 처리 불가: 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 직접 호출 실패를 모니터링할 수 있습니다. 많은 수의 직접 호출이 실패하면 다운스트림 시스템으로의 추가 요청 전송이 중단되고 다운스트림 시스템을 다시 사용할 수 있는지 여부를 테스트하기 위한 직접 호출이 가끔 전송됩니다.
3. 파라미터 스토어 사용 불가: 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 작동하지 않는 기타 종속성 사용 불가: 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 인스턴스 사용 불가: 거의 모든 관계형 데이터베이스와 마찬가지로 Amazon Relational Database Service는 기본 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 Serverless를 사용하면 이러한 문제를 부분적으로 완화할 수 있습니다. 고가용성 요구 사항이 매우 높은 경우 기본 라이터에

게 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 기본 항목을 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 (summarizes Circuit Breaker from “Release It!” book)
- Error Retries and Exponential Backoff in AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software”
- 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 비디오:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders’ Library (DOP328)

REL05-BP02 요청 제한

예상치 못한 수요 증가로 인한 리소스 고갈을 완화하기 위해 요청을 제한합니다. 스로틀링 속도 이하의 요청은 처리되지만 정의된 한도를 초과한 요청은 거부되고 요청이 스로틀링되었음을 알리는 메시지가 표시됩니다.

원하는 성과: 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 스케일링되는 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 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 처리량은 충분하면서 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 찾습니다.
- 설정하려는 제한 또는 증가를 허용하려는 제한을 테스트하고 테스트된 제한을 문서화합니다.

- 테스트에서 설정한 범위를 초과하여 제한을 늘리지 않습니다. 제한을 늘릴 때는 증가를 적용하기 전에 프로비저닝된 리소스가 이미 테스트 시나리오의 리소스와 동일하거나 더 큰지 확인합니다.

리소스

관련 모범 사례:

- REL04-BP03 일정한 작업 처리
- REL05-BP03 재시도 직접 호출 제어 및 제한

관련 문서:

- Amazon API Gateway: 처리량 향상을 위해 API 요청 제한
- AWS WAF: Rate-based rule statement
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source
- AWS Lambda: 최대 동시성

관련 예제:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python token-bucket
- Node token-bucket
- .NET System Threading Rate Limiting

관련 비디오:

- Implementing GraphQL API security best practices with AWS AppSync

관련 도구:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS

- Amazon Kinesis
- AWS WAF
- Virtual Waiting Room on AWS

REL05-BP03 재시도 직접 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 성과: 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 서드파티 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 직접 호출을 재시도하여 중복 결과 등 예상치 못한 부작용을 초래합니다.

이 모범 사례 확립의 이점: 재시도는 요청이 실패했을 때 클라이언트가 원하는 성과를 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 않을 경우 노출되는 위험 수준: 높음

구현 가이드

재시도 직접 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 직접 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 멱등성을 유지하는지 확인합니다. 재시도를 구현한 후에는 반드시 테스트를 거치고 프로덕션 환경에서 정기적으로 실행해야 합니다.
- AWS 서비스 API를 직접 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:

- REL04-BP04 변경 작업에 멱등성 부여
- REL05-BP02 요청 제한
- REL05-BP04 빠른 실패 및 대기열 제한

- REL05-BP05 클라이언트 제한 시간 설정
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

관련 문서:

- Error Retries and Exponential Backoff in AWS
- Amazon Builders' Library: 시간 제한, 재시도 및 지터를 사용한 백오프
- Exponential Backoff and Jitter
- Making retries safe with idempotent APIs

관련 예제:

- Spring Retry
- Resilience4j Retry

관련 비디오:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

관련 도구:

- AWS SDKs and Tools: Retry behavior
- AWS Command Line Interface: AWS CLI retries

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 버퍼링합니다. 하지만 긴 대기열 백로그가 쌓이도록 두지 않습니다. 그러면 클라이언트가 이미 포기한 무효 요청을 처리할 수 있습니다.

원하는 성과: 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 스파이크

를 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 않을 경우 노출되는 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 강

력한 아키텍처 기법입니다. Amazon CloudWatch에서는 장애를 모니터링하고 경보를 알리는 기능을 제공합니다. 시스템에 장애가 발생한 것으로 확인되면 손상된 리소스를 제거하는 등 완화 전략을 실행할 수 있습니다. 시스템에서 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 지표를 구현하고 이를 사용하여 시스템 문제를 명시적으로 경고합니다. Amazon CloudWatch를 사용하면 애플리케이션 로그 패턴 및 SDK 계측을 기반으로 지표와 경보를 생성할 수 있습니다.
- 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 수준에 관한 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 직접 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
- Fail Fast
- 내 Amazon SQS 대기열에서 메시지 백로그가 증가하는 것을 방지하려면 어떻게 해야 하나요?
- Elastic Load Balancing: Zonal Shift
- Amazon Application Recovery Controller: 트래픽 장애 조치에 대한 라우팅 컨트롤

관련 예제:

- Enterprise Integration Patterns: Dead Letter Channel

관련 비디오:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

관련 도구:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 클라이언트 제한 시간 설정

연결 및 요청에 대한 시간 제한을 적절하게 설정하고 체계적으로 확인합니다. 워크로드 세부 사항을 인식하지 못하므로 기본값에 의존하지 않습니다.

원하는 성과: 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 차단기 개방을 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 직접 호출의 시간 제한 오류를 처리하기 위한 패턴을 간과합니다.
- 서비스 직접 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 직접 호출 시간 제한이 구성되고 시스템이 시간 제한을 정상적으로 처리하도록 설계되어 원격 직접 호출이 비정상적으로 느리게 응답하고 서비스 클라이언트에서 시간 제한 오류를 정상적으로 처리할 때 리소스가 절약됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

서비스 종속성 직접 호출과 일반적으로 프로세스 전체의 모든 직접 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 고려하세요.

- 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 제한 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 Functions에서는 AWS 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 직접 호출에 대한 시간 제한을 구성하고 기본 제공되는 언어 시간 제한 기능 또는 오픈소스 시간 제한 라이브러리를 활용합니다.
- 워크로드가 AWS SDK를 사용하여 호출하는 경우 설명서에서 언어별 시간 제한 구성을 검토합니다.
  - 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`을 적용하여 AWS 서비스에 대한 일회성 AWS CLI 명령을 제어합니다.
- 오류 시나리오를 사전에 처리할 수 있도록 원격 서비스 직접 호출의 시간 제한을 모니터링하고 지속적인 오류에 대한 경고를 설정합니다.
- 직접 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.
- Lambda 함수에서 시간 제한을 구성합니다.
- API Gateway 클라이언트는 시간 제한을 처리할 때 자체 재시도를 구현해야 합니다. API Gateway는 다운스트림 통합에 대해 50밀리초~29초의 통합 시간 제한을 지원하고 통합 요청의 제한 시간이 초과되면 재시도하지 않습니다.
- 제한 시간을 초과할 때 원격 직접 호출을 방지하기 위해 회로 차단기 패턴을 구현합니다. 직접 호출이 실패하지 않도록 회로를 개방하고 직접 호출이 정상적으로 응답할 때 회로를 폐쇄합니다.
- 컨테이너 기반 워크로드의 경우 기본 제공 시간 제한 및 회로 차단기를 활용하는 App Mesh Envoy 기능을 검토하세요.
- 특히 AWS Step Functions 기본 SDK 및 지원되는 Step Functions 통합을 ㅈ기접 호출하여 워크로드를 간소화하는 경우 AWS를 사용하여 원격 서비스 직접 호출을 위한 로우 코드 회로 차단기를 구축합니다.

리소스

관련 모범 사례:

- REL05-BP03 재시도 직접 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: Retries and Timeouts
- Amazon Builders' Library: 시간 제한, 재시도 및 지터를 사용한 백오프

- Amazon API Gateway 할당량 및 중요 정보
- AWS Command Line Interface: Command line options
- AWS SDK for Java 2.x: Configure API Timeouts
- AWS Botocore using the config object and Config Reference
- AWS SDK for .NET: 재시도 및 제한 시간
- AWS Lambda: Lambda 함수 옵션 구성

관련 예제:

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB
- Martin Fowler: CircuitBreaker

관련 도구:

- AWS SDKs
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 가능한 경우 시스템을 상태 비저장으로 설계

시스템은 상태를 필요로 하지 않거나, 서로 다른 클라이언트 요청 간에 디스크 및 메모리에 로컬로 저장된 데이터에 종속성이 없도록 상태를 오프로드해야 합니다. 이렇게 하면 가용성에 미치는 영향 없이 서버를 대체할 수 있습니다.

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.

상태 비저장으로 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 수평적 스케일링이 가능합니다.

이 모범 사례 확립의 이점: 상태 비저장으로 설계된 시스템은 수평적 스케일링에 더 잘 적응하므로 변동하는 트래픽과 수요에 따라 용량을 추가하거나 제거할 수 있습니다. 또한 기본적으로 장애에 대한 복원력이 뛰어나며 애플리케이션 개발에 유연성과 민첩성을 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

애플리케이션을 상태 비저장으로 설계합니다. 상태 비저장 애플리케이션은 수평적 스케일링을 가능하게 하며, 개별 노드의 장애에 대한 내결함성이 있습니다. 아키텍처 내에서 상태를 유지하는 애플리케이션 구성 요소를 분석하고 이해합니다. 이를 통해 상태 비저장 설계로의 전환이 미칠 수 있는 영향을 평가할 수 있습니다. 상태 비저장 아키텍처는 사용자 데이터를 분리하고 세션 데이터를 오프로드합니다. 이를 통해 각 구성 요소를 독립적으로 규모 조정하여 다양한 워크로드 수요를 충족하고 리소스 활용도를 최적화할 수 있는 유연성을 제공합니다.

구현 단계

- 애플리케이션의 상태 저장 구성 요소를 식별하고 이해합니다.
- 핵심 애플리케이션 논리에서 사용자 데이터를 분리 및 관리하여 데이터를 분리합니다.
  - Amazon Cognito는 자격 증명 풀, 사용자 풀 및 Amazon Cognito Sync 등의 기능을 사용하여 애플리케이션 코드에서 사용자 데이터를 분리할 수 있습니다.
  - AWS Secrets Manager를 사용하면 보안 암호를 안전한 중앙 위치에 저장하여 사용자 데이터를 분리할 수 있습니다. 즉, 애플리케이션 코드에 보안 암호를 저장할 필요가 없으므로 보안이 강화됩니다.
  - 이미지 및 문서와 같은 대용량의 비정형 데이터를 저장하는 데 Amazon S3 사용을 고려합니다. 애플리케이션은 필요할 때 이 데이터를 검색할 수 있으므로 데이터를 메모리에 저장할 필요가 없습니다.
  - Amazon DynamoDB를 사용하여 사용자 프로필과 같은 정보를 저장합니다. 애플리케이션은 이 데이터를 거의 실시간으로 쿼리할 수 있습니다.
- 세션 데이터를 데이터베이스, 캐시 또는 외부 파일로 오프로드합니다.
  - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (Amazon EFS), Amazon MemoryDB는 세션 데이터를 오프로드하는 데 사용할 수 있는 AWS 서비스의 예입니다.
- 선택한 스토리지 솔루션에 어떤 상태 및 사용자 데이터를 유지해야 하는지 파악한 후 상태 비저장 아키텍처를 설계합니다.

리소스

관련 모범 사례:

- REL11-BP03 모든 계층에서 복구 자동화

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
- AWS에서 상태 비저장 웹 티어 모범 사례

REL05-BP07 비상 레버 구현

비상 레버는 워크로드의 가용성에 미치는 영향을 신속하게 완화할 수 있는 프로세스입니다.

비상 레버는 알려진 메커니즘과 테스트를 거친 메커니즘을 사용하여 구성 요소 또는 종속성의 동작을 비활성화, 제한 또는 변경하는 방식으로 작동합니다. 이를 통해 예상치 못한 수요 증가로 인한 리소스 고갈이 유발하는 워크로드 장애를 완화하고 워크로드 내 비핵심 구성 요소에 장애가 미치는 영향을 줄일 수 있습니다.

원하는 성과: 비상 레버를 구현하면 바람직한 사례로 알려진 프로세스를 구축하여 워크로드에서 핵심 구성 요소의 가용성을 유지할 수 있습니다. 비상 레버를 작동시키는 동안에도 워크로드는 단계적으로 줄어들고 비즈니스의 핵심 기능을 계속 수행해야 합니다. 단계적 성능 저하에 대한 자세한 내용은 REL05-BP01 관련 하드 종속성을 소프트 종속성으로 변환하는 단계적 성능 저하 구현을 참조하세요.

일반적인 안티 패턴:

- 비핵심 종속성의 장애가 핵심 워크로드의 가용성에 영향을 미칩니다.
- 비핵심 구성 요소가 손상되었을 때 핵심 구성 요소 동작을 테스트하거나 확인하지 않습니다.
- 비상 레버의 활성화 또는 비활성화에 대한 명확하고 결정적인 기준이 정의되어 있지 않습니다.

이 모범 사례 확립의 이점: 비상 레버를 구현하면 해석기에 확립된 프로세스를 제공하여 예상치 못한 수요 급증 또는 비핵심 종속성 장애에 대응하도록 함으로써 워크로드에서 핵심 구성 요소의 가용성을 높일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

- 워크로드의 핵심 구성 요소를 식별합니다.
- 비핵심 구성 요소의 장애를 견딜 수 있도록 워크로드의 핵심 구성 요소를 설계하고 설계합니다.
- 테스트를 수행하여 비핵심 구성 요소에 장애가 발생한 경우 핵심 구성 요소의 동작을 검증합니다.
- 관련 지표 또는 트리거를 정의하고 모니터링하여 비상 레버 절차를 시작합니다.
- 비상 레버를 구성하는 절차를 수동 또는 자동으로 정의합니다.

구현 단계

- 워크로드에서 비즈니스의 핵심 구성 요소를 식별합니다.
  - 워크로드의 각 기술 구성 요소를 관련 비즈니스 기능에 매핑하고 핵심 또는 비핵심으로 평가해야 합니다. Amazon의 핵심 기능과 비핵심 기능의 예를 보려면 Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second를 참조하세요.
  - 이런 과정은 기술 의사 결정이자 동시에 비즈니스 의사 결정이며 조직과 워크로드에 따라 다릅니다.
- 비핵심 구성 요소의 장애를 견딜 수 있도록 워크로드의 핵심 구성 요소를 설계하고 설계합니다.
  - 종속성 분석 중에 잠재적 장애 모드를 모두 고려하고 비상 레버 메커니즘이 다운스트림 구성 요소에 핵심 기능을 제공하는지 확인합니다.
- 비상 레버를 작동시키는 동안 핵심 구성 요소의 동작을 검증하기 위한 테스트를 실시합니다.
  - 바이모달 동작은 사용하지 마세요. 자세한 내용은 REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지를 참조하세요.
- 관련 지표를 정의 및 모니터링하고 알림을 설정하여 비상 레버 절차를 시작합니다.
  - 모니터링할 올바른 지표를 찾는 방법은 워크로드에 따라 다릅니다. 지표의 예로는 지연 시간 또는 종속성에 대한 요청 실패 횟수가 있습니다.
- 비상 레버를 구성하는 절차를 수동 또는 자동으로 정의합니다.
  - 여기에는 로드 감소, 요청 제한 또는 단계적 성능 저하 구현과 같은 메커니즘이 포함될 수 있습니다.

리소스

관련 모범 사례:

- REL05-BP01 관련 하드 종속성을 소프트 종속성으로 변환하는 단계적 성능 저하 구현
- REL05-BP02 요청 제한
- REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지

관련 문서:

- 안전하고 간편한 배포 자동화
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

관련 비디오:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

변경 관리

Questions

- REL 6. 워크로드 리소스는 어떻게 모니터링하나요?
- REL 7. 수요 변화에 적응하도록 워크로드를 설계하려면 어떻게 해야 하나요?
- REL 8. 변경은 어떻게 구현하나요?

REL 6. 워크로드 리소스는 어떻게 모니터링하나요?

로그와 지표는 워크로드 상태를 파악할 수 있는 강력한 도구입니다. 로그와 지표를 모니터링하고 임계값을 초과하거나 중요한 이벤트가 발생할 경우 알림을 보내도록 워크로드를 구성할 수 있습니다. 모니터링을 수행하면 워크로드가 저성능 임곗값을 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 로그 분석

- REL06-BP06 모니터링 범위 및 지표를 정기적으로 검토
- 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 대시보드에서 AWS 서비스를 모니터링합니다.

프론트엔드, 비즈니스 로직 및 스토리지 계층을 포함하여 워크로드의 모든 구성 요소를 모니터링해야 합니다. 주요 지표를 정의하고, 필요한 경우 로그에서 주요 지표를 추출하는 방법을 정의하며, 해당 경보 이벤트를 간접 호출하는 임곗값을 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 리소스의 기반이 되는 AWS 서비스의 성능 및 가용성에 대한 맞춤형 보기를 보려면 AWS Health Dashboard를 사용합니다.

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 CloudWatch와 같은 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 계절성 및 추세 변화를 설명합니다.

AWS는 사용할 수 있는 모니터링 및 로그 정보를 풍부하게 제공하며, 이를 통해 사용자는 워크로드별 지표를 정의하고, 수요 변화가 있는 프로세스를 정의하며, ML 전문성과 상관없이 기계 학습 기법을 도입하는 데 이 정보를 사용할 수 있습니다.

또한 모든 외부 엔드포인트를 모니터링하여 기본 구현과 독립되어 있는지 확인합니다. 이 능동적 모니터링은 가상 트랜잭션에서도 수행할 수 있습니다(사용자 canary라고도 함, 단, 카나리 배포와 혼동하지 않도록 주의). 그러면 워크로드의 클라이언트가 수행하는 작업에 부합하는 일반적인 여러 작업을 주기적으로 실행합니다. 작업 기간은 짧아야 하며 테스트 중에 워크로드에 과부하가 발생하지 않아야 합니다. Amazon CloudWatch Synthetics를 사용하면 엔드포인트 및 API 모니터링을 위한 가상 canary를 생성할 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 성과:

워크로드의 모든 구성 요소로부터 핵심적인 지표를 수집하고 사용하여 워크로드 신뢰성과 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 모니터링합니다.
- 워크로드별 지표를 생성하지 않고 워크로드가 사용하는 AWS 서비스에서 제공하는 지표에만 의존합니다.
- 워크로드에서 기술적 지표만 사용하고 워크로드가 기여하는 비기술적 KPI와 관련된 지표는 모니터링하지 않습니다.
- 프로덕션 트래픽과 간단한 상태 확인에 의존하여 워크로드 상태를 모니터링하고 평가합니다.

이 모범 사례 확립의 이점: 워크로드의 모든 티어에서 모니터링할 경우 워크로드를 구성하는 구성 요소의 문제를 보다 신속하게 예측하고 해결할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

1. 가능한 경우 로깅을 활성화합니다. 워크로드의 모든 구성 요소에서 모니터링 데이터를 가져와야 합니다. S3 액세스 로그와 같은 추가 로깅을 활성화하고 워크로드에서 워크로드별 데이터를 로깅하도록 허용합니다. Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling, Amazon EMR과 같은 서비스에서 CPU, 네트워크 I/O 및 디스크 I/O 평균에 대한 지표를 수집합니다. CloudWatch에 지표를 게시하는 AWS 서비스 목록은 CloudWatch 지표를 게시하는 AWS 서비스를 참조하세요.
2. 모든 기본 지표를 검토하고 데이터 수집 격차를 살펴봅니다. 모든 서비스에서는 기본 지표를 생성합니다. 기본 지표를 수집하면 워크로드 구성 요소 간 종속성과 구성 요소 신뢰성 및 성능이 워크로드에 미치는 영향을 더 잘 이해할 수 있습니다. 또한 AWS CLI 또는 API를 사용하여 자체 지표를 생성해 CloudWatch에 자체 지표를 게시할 수 있습니다.
3. 모든 지표를 평가하여 워크로드의 각 AWS 서비스에 대해 어떤 지표를 경고할지 결정합니다. 워크로드 신뢰성에 큰 영향을 미치는 지표의 하위 세트를 선택할 수 있습니다. 핵심 지표와 임곗값에 집중하면 알림의 수를 세분화하고 오탐지를 최소화할 수 있습니다.
4. 알림을 정의하고 알림이 간접 호출된 후 워크로드에 대한 복구 프로세스를 정의합니다. 알림을 정의하면 인시던트로부터 복구하는 데 필요한 단계를 빠르게 알리고, 에스컬레이션하며, 단계를 따라 사전에 정해진 Recovery Time Objective(RTO)를 달성할 수 있습니다. Amazon CloudWatch 경보를 사용하여 자동화된 워크플로를 간접 호출하고 정의된 임곗값에 따라 복구 절차를 시작할 수 있습니다.
5. 워크로드 상태에 대한 관련 데이터를 수집하기 위해 가상 트랜잭션 사용 방법을 알아봅니다. 가상 트랜잭션은 동일한 경로를 따라 고객과 동일한 작업을 수행하므로 워크로드에 고객 트래픽이 없는

경우에도 고객 경험을 지속적으로 확인할 수 있습니다. 가상 트랜잭션을 사용하면 고객보다 먼저 문제를 발견할 수 있습니다.

리소스

관련 모범 사례:

- REL11-BP03 모든 계층에서 복구 자동화

관련 문서:

- Getting started with your AWS Health Dashboard – Your account health
- CloudWatch 지표를 게시하는 AWS 서비스
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer
- AWS Lambda에 대한 Amazon CloudWatch logs 액세스
- Amazon S3 서버 액세스 로깅
- Classic Load Balancer 액세스 로그 활성화
- Exporting log data to Amazon S3
- Amazon EC2 인스턴스에 CloudWatch 에이전트 설치
- 사용자 지정 지표 게시
- Amazon CloudWatch 대시보드 사용
- Amazon CloudWatch 지표 사용
- Using Canaries (Amazon CloudWatch Synthetics)
- What are Amazon CloudWatch Logs?

사용 설명서:

- 추적 생성
- Amazon EC2 Linux 인스턴스의 메모리 및 디스크 지표 모니터링
- 컨테이너 인스턴스와 함께 CloudWatch Logs 사용
- VPC 흐름 로그
- What is Amazon DevOps Guru?
- AWS X-Ray란 무엇입니까?

관련 블로그:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

관련 예제:

- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- One Observability 워크숍

REL06-BP02 지표 정의 및 계산(집계)

워크로드 구성 요소에서 지표와 로그를 수집하고 관련 집계 지표를 계산합니다. 이러한 지표를 통해 워크로드를 광범위하고 심층적으로 관찰할 수 있으며 복원 태세를 크게 개선할 수 있습니다.

관찰성은 단순히 워크로드 구성 요소에서 지표를 수집하고 이를 보고 관련 알림을 보내는 데 그치지 않습니다. 워크로드의 동작에 대해 전체적으로 이해하는 것입니다. 이 동작 정보는 워크로드의 모든 구성 요소에서 가져옵니다. 여기에는 워크로드가 의존하는 클라우드 서비스, 잘 작성된 로그 및 지표가 포함됩니다. 이 데이터를 통해 워크로드의 전체 동작을 감독할 수 있을 뿐만 아니라 모든 구성 요소와 모든 작업 단위 간의 상호 작용을 세부적으로 파악할 수 있습니다.

원하는 성과:

- 워크로드 구성 요소 및 AWS 서비스 종속성에서 로그를 수집하여 쉽게 액세스하고 처리할 수 있는 중앙 위치에 게시합니다.
- 로그에는 충실도가 높고 정확한 타임스탬프가 포함되어 있습니다.
- 로그에는 추적 식별자, 사용자 또는 계정 식별자, 원격 IP 주소와 같은 처리 컨텍스트에 대한 관련 정보가 포함되어 있습니다.
- 개괄적인 관점에서 워크로드의 동작을 나타내는 집계 지표를 로그에서 생성합니다.
- 집계된 로그를 쿼리하여 워크로드에 대한 심층적이고 관련 있는 인사이트를 얻고 실제 및 잠재적 문제를 식별할 수 있습니다.

일반적인 안티 패턴:

- 워크로드가 실행되는 컴퓨팅 인스턴스 또는 워크로드가 사용하는 클라우드 서비스에서 관련 로그 또는 지표를 수집하지 않습니다.
- 비즈니스 핵심 성과 지표(KPI)와 관련된 로그로그 및 지표 수집을 간과합니다.
- 집계 및 상관관계 없이 워크로드 관련 원격 측정을 개별적으로 분석합니다.

- 지표와 로그가 너무 빨리 만료되도록 허용하여 추세 분석과 반복되는 문제 식별이 방해됩니다.

이러한 모범 사례 확립의 이점: 더 많은 이상을 감지하고 워크로드의 다양한 구성 요소 간에 이벤트와 지표의 상관관계를 파악할 수 있습니다. 지표만으로는 파악하기 힘든 경우가 많은, 로그에 포함된 정보를 기반으로 워크로드 구성 요소에서 인사이트를 생성할 수 있습니다. 대규모로 로그를 쿼리하여 장애 원인을 더 빠르게 확인할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드 및 해당 구성 요소와 관련된 원격 측정 데이터의 소스를 식별합니다. 이 데이터는 운영 체제 (OS) 및 Java 등의 애플리케이션 런타임과 같은 지표를 게시하는 구성 요소뿐만 아니라 애플리케이션 및 클라우드 서비스 로그에서도 제공됩니다. 예를 들어 웹 서버는 일반적으로 타임스탬프, 처리 지연 시간, 사용자 ID, 원격 IP 주소, 경로 및 쿼리 문자열과 같은 세부 정보와 함께 각 요청을 기록합니다. 이러한 로그의 세부 정보 수준은 세부 쿼리를 수행하고 다른 방법으로는 제공되지 않는 지표를 생성하는 데 도움이 됩니다.

적절한 도구와 프로세스를 사용하여 지표와 로그를 수집합니다. Amazon EC2 인스턴스에서 실행되는 애플리케이션에서 생성된 로그는 Amazon CloudWatch Agent와 같은 에이전트가 수집하여 Amazon CloudWatch Logs와 같은 중앙 스토리지 서비스에 게시될 수 있습니다. AWS Lambda 및 Amazon Elastic Container Service와 같은 AWS 관리형 컴퓨팅 서비스는 자동으로 CloudWatch Logs에 로그를 게시합니다. 워크로드에서 사용하는 Amazon CloudFront, Amazon S3, Elastic Load Balancing 및 Amazon API Gateway와 같은 AWS 스토리지 및 처리 서비스에 대한 로그 수집을 활성화합니다.

동작 패턴을 더 명확하게 보고 관련 구성 요소 그룹에 상관관계가 있는 문제를 격리하는 데 도움이 되는 차원으로 원격 측정 데이터를 강화합니다. 추가한 후에는 구성 요소 동작을 더 세부적으로 관찰하고, 상관관계가 있는 장애를 감지하고, 적절한 수정 조치를 취할 수 있습니다. 유용한 차원의 예로는 가용 영역, EC2 인스턴스 ID, 컨테이너 작업 또는 포드 ID가 있습니다.

지표와 로그를 수집한 후에는 쿼리를 작성하고 정상 동작과 이상 동작 모두에 관해 유용한 인사이트를 제공하는 집계 지표를 생성할 수 있습니다. 예를 들어 Amazon CloudWatch Logs Insights를 사용하여 애플리케이션 로그에서 사용자 지정 지표를 도출하고, Amazon CloudWatch Metrics Insights를 사용하여 대규모로 지표를 쿼리하고, Amazon CloudWatch Container Insights를 사용하여 컨테이너화된 애플리케이션 및 마이크로서비스에서 지표와 로그를 수집, 집계 및 요약할 수 있고, AWS Lambda 함수를 사용하는 경우 Amazon CloudWatch Lambda Insights를 사용할 수 있습니다. 집계 오류율 지표를 생성하려면 구성 요소 로그에서 오류 응답 또는 메시지가 발견될 때마다 카운터를 늘리거나 기존 오류율 지표의 집계 값을 계산할 수 있습니다. 이 데이터를 사용하여 성능이 가장 낮은 요청 또는 프로세스와 같은 말단 행동을 보여주는 히스토그램을 생성할 수 있습니다. CloudWatch Logs 이상 탐지와 같은 솔루

션을 사용하여 이 데이터를 실시간으로 스캔하여 이상 패턴을 확인할 수도 있습니다. 이러한 인사이트는 대시보드에 배치하여 요구 사항과 기본 설정에 따라 정리할 수 있습니다.

로그 쿼리를 통해 워크로드 구성 요소가 특정 요청을 처리하는 방법을 이해하고 워크로드의 복원력에 영향을 미치는 요청 패턴 또는 기타 맥락을 파악할 수 있습니다. 필요에 따라 더 쉽게 실행할 수 있도록 애플리케이션 및 기타 구성 요소의 작동 방식에 대한 지식을 기반으로 쿼리를 미리 조사하고 준비하면 유용할 수 있습니다. 예를 들어, CloudWatch Logs Insights를 사용하면 CloudWatch Logs 내 로그 데이터를 대화식으로 검색하고 분석할 수 있습니다. Amazon Athena를 사용하여 많은 AWS 서비스를 포함하여 여러 소스의 로그를 페타바이트 규모로 쿼리할 수도 있습니다.

로그 보존 정책을 정의할 때 기록 로그의 값을 고려합니다. 기록 로그는 워크로드 성능의 장기 사용 및 동작 패턴, 회귀 및 개선을 식별하는 데 도움이 될 수 있습니다. 영구적으로 삭제된 로그는 나중에 분석할 수 없습니다. 그러나 기록 로그의 가치는 장기간에 걸쳐 감소하는 경향이 있습니다. 적절하게 균형을 맞추고 적용될 수 있는 법적 또는 계약상의 요구 사항을 준수하는 정책을 선택합니다.

구현 단계

1. 관찰성 데이터에 대한 수집, 스토리지, 분석 및 표시 메커니즘을 선택합니다.
2. 워크로드의 적절한 구성 요소(예: Amazon EC2 인스턴스 및 사이드카 컨테이너)에 지표 및 로그 수집기를 설치하고 구성합니다. 예기치 않게 중지되는 경우 자동으로 다시 시작하도록 이러한 수집기를 구성합니다. 임시 게시 실패가 애플리케이션에 영향을 미치거나 데이터가 손실되지 않도록 수집기에 디스크 또는 메모리 버퍼링을 활성화합니다.
3. 워크로드의 일부로 사용하는 AWS 서비스에 대한 로깅을 활성화하고 필요한 경우 선택한 스토리지 서비스에 해당 로그를 전달합니다. 자세한 지침은 해당 서비스의 사용 설명서 또는 개발자 안내서를 참조하세요.
4. 원격 측정 데이터를 기반으로 워크로드와 관련된 운영 지표를 정의합니다. 이는 워크로드 구성 요소에서 방출되는 직접 지표를 기반으로 할 수 있으며, 여기에는 비즈니스 KPI 관련 지표 또는 합계, 비율, 백분위수 또는 히스토그램과 같은 집계된 계산 결과가 포함될 수 있습니다. 로그 분석기를 사용하여 이러한 지표를 계산하고 적절하게 대시보드에 배치합니다.
5. 필요에 따라 워크로드 구성 요소, 요청 또는 트랜잭션 동작을 분석하기 위해 적절한 로그 쿼리를 준비합니다.
6. 구성 요소 로그에 대한 로그 보존 정책을 정의하고 활성화합니다. 정책에서 허용하는 것보다 오래된 로그는 주기적으로 삭제합니다.

리소스

관련 모범 사례:

- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP03 알림 전송(실시간 처리 및 경보)
- REL06-BP04 응답 자동화(실시간 처리 및 경보)
- REL06-BP05 로그 분석
- REL06-BP06 모니터링 범위 및 지표를 정기적으로 검토
- REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링

관련 설명서:

- How Amazon CloudWatch works
- Amazon Managed Prometheus
- Amazon Managed Grafana
- Analyzing log data with CloudWatch Logs Insights
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Query your metrics with CloudWatch Metrics Insights
- AWS Distro for OpenTelemetry
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Searching and Filtering Log Data
- Sending Logs Directly to Amazon S3
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측

관련 워크숍:

- One Observability 워크숍

관련 도구:

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 알림 전송(실시간 처리 및 경보)

조직은 잠재적인 문제를 탐지하면 해당 직원과 시스템에 실시간 알림과 경고를 보내 이러한 문제에 신속하고 효과적으로 대응할 수 있도록 합니다.

원하는 성과: 서비스 및 애플리케이션 지표를 기반으로 관련 경보를 구성하여 운영 이벤트에 신속하게 대응할 수 있습니다. 경보 임곗값이 위반되면 적절한 인력과 시스템에 알림이 전송되어 근본적인 문제를 해결할 수 있습니다.

일반적인 안티 패턴:

- 임곗값이 지나치게 높은 경보를 구성하여 중요한 알림을 보내지 못합니다.
- 임곗값이 너무 낮은 경보를 구성하여 과도한 알림 노이즈로 인해 중요한 알림에 대한 조치를 취하지 않습니다.
- 사용량이 변경될 때 경보 및 임곗값을 업데이트하지 않습니다.
- 자동화된 작업을 통해 가장 잘 해결되는 경보에 대해, 자동화된 작업을 생성하지 않고 담당자에게 알림을 보내 과도한 알림을 전송합니다.

이 모범 사례 확립의 이점: 적절한 인력과 시스템에 실시간 알림과 경고를 전송하면 문제를 조기에 탐지하고 운영 인시던트에 신속하게 대응할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드는 애플리케이션의 가용성에 영향을 미칠 수 있는 문제의 탐지 가능성을 개선하고 자동화된 대응을 위한 트리거 역할을 할 수 있도록 실시간 처리 및 경보 기능을 갖추어야 합니다. 조직은 정의된 지표로 경고를 생성하여 중요한 이벤트가 발생하거나 지표가 임곗값을 초과할 때마다 알림을 수신하여 실시간 처리 및 경보를 수행할 수 있습니다.

Amazon CloudWatch를 사용하면 지표 및 정적 임곗값, 이상 탐지 및 기타 기준에 기반한 CloudWatch 경보를 사용하는 복합 경보를 만들 수 있습니다. CloudWatch를 사용하여 구성할 수 있는 경보 유형에 대한 자세한 내용은 CloudWatch 설명서의 경보 섹션을 참조하세요.

팀의 AWS 리소스 지표 및 알림에 대한 사용자 지정 보기를 구성할 수 있습니다. 이 경우 CloudWatch 대시보드를 사용할 수 있습니다. CloudWatch 콘솔의 사용자 지정 가능한 홈 페이지를 사용하면 여러 리전에 걸쳐 단일 보기에서 리소스를 모니터링할 수 있습니다.

경보는 Amazon SNS 주제로 알림 전송, Amazon EC2 작업이나 Amazon EC2 Auto Scaling 작업 수행, AWS Systems Manager에서 인시던트 또는 OpsItem 생성과 같은 하나 이상의 작업을 수행할 수 있습니다.

Amazon CloudWatch는 경보에서 상태가 변경되면 Amazon SNS를 사용하여 알림을 전송하고, 게시자(생산자)에서 구독자(소비자)에게 메시지를 전달합니다. Amazon SNS 알림 설정에 대한 자세한 내용은 Configuring Amazon SNS를 참조하세요.

CloudWatch 경보가 생성, 업데이트, 삭제되거나 상태가 변경될 때마다 CloudWatch는 EventBridge에 이벤트를 전송합니다. 이러한 이벤트와 함께 EventBridge를 사용하여 경보 상태가 변경될 때마다 사용자에게 알리거나 Systems Manager Automation을 사용하여 계정에서 이벤트를 자동으로 트리거하는 등의 작업을 수행하는 규칙을 만들 수 있습니다.

AWS Health로 최신 정보를 확인하세요. AWS Health는 AWS 클라우드 리소스 상태에 대한 신뢰할 수 있는 정보 소스입니다. AWS Health를 사용하면 확인된 서비스 이벤트에 대한 알림을 받아 영향을 완화하기 위한 조치를 신속하게 취할 수 있습니다. AWS 사용자 알림를 통해 이메일 및 채팅 채널에 적합한 AWS Health 이벤트 알림을 생성하고, Amazon EventBridge를 통해 모니터링 및 알림 도구와 프로그래밍 방식으로 통합할 수 있습니다. AWS Organizations를 사용하는 경우 여러 계정에서 AWS Health 이벤트를 집계합니다.

EventBridge 또는 Amazon SNS는 언제 사용해야 하나요?

이벤트 기반 애플리케이션을 개발하는 데 EventBridge 및 Amazon SNS를 모두 사용할 수 있으며, 특정 요구 사항에 따라 선택이 달라집니다.

Amazon EventBridge는 자체 애플리케이션, SaaS 애플리케이션 및 AWS 서비스의 이벤트에 대응하는 애플리케이션을 구축하려는 경우 권장됩니다. EventBridge는 서드파티 SaaS 파트너와 직접 통합되는 유일한 이벤트 기반 서비스입니다. 또한 EventBridge는 개발자가 계정에 리소스를 생성할 필요 없이 200개가 넘는 AWS 서비스에서 이벤트를 자동으로 수집합니다.

EventBridge는 이벤트에 대해 정의된 JSON 기반 구조를 사용하며, 이벤트 본문 전체에 적용되는 규칙을 생성하여 대상으로 전달한 이벤트를 선택합니다. EventBridge는 현재 20개가 넘는 AWS 서비스(AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams, Amazon Data Firehose 등)를 대상으로 지원합니다.

Amazon SNS는 높은 팬 아웃(수천 또는 수백만 개의 엔드포인트)이 필요한 애플리케이션에 권장됩니다. 일반적으로 고객이 Amazon SNS를 규칙 대상으로 사용하여 필요한 이벤트를 필터링하고 여러 엔드포인트로 팬아웃하는 패턴을 볼 수 있습니다.

메시지는 비정형 양식으며, 어떤 형식이든 가능합니다. Amazon SNS는 Lambda, Amazon SQS, HTTP/S 엔드포인트, SMS, 모바일 푸시, 이메일과 같은 6가지 여러 유형의 대상으로 메시지 전달을

지원합니다. Amazon SNS의 일반적인 지연 시간은 30밀리초 미만입니다. 다양한 AWS 서비스에서 Amazon SNS 메시지를 전송하도록 서비스를 구성하여 이를 수행합니다(Amazon EC2, Amazon S3, Amazon RDS 등 30개가 넘는 서비스 지원).

구현 단계

1. Amazon CloudWatch 경보를 사용하여 경보를 생성합니다.
   a. 지표 경보에서는 단일 CloudWatch 지표 또는 CloudWatch 지표에 종속된 표현식을 모니터링합니다. 경보는 여러 시간 간격에 걸쳐 임곗값과 비교한 지표 또는 표현식의 값에 따라 하나 이상의 작업을 시작합니다. Amazon SNS 주제에 알림 전송, Amazon EC2 작업이나 Amazon EC2 Auto Scaling 작업 수행, AWS Systems Manager에서 인시던트 또는 OpsItem 생성으로 작업이 구성될 수 있습니다.
   b. 복합 경보는 사용자가 만든 다른 경보의 경보 조건을 고려하는 규칙 표현식으로 구성됩니다. 복합 경보는 모든 규칙 조건이 충족되는 경우에만 경보 상태로 전환됩니다. 복합 경보의 규칙 표현식에 지정된 경보에는 지표 경보 및 추가 복합 경보가 포함될 수 있습니다. 복합 경보는 경보 상태가 변경될 때 Amazon SNS 알림을 전송할 수 있고, 경보 상태로 진입할 때 Systems Manager OpsItem 또는 인시던트를 생성할 수 있지만, Amazon EC2 작업 또는 Auto Scaling 작업은 수행할 수 없습니다.
2. Amazon SNS 알림을 설정합니다. CloudWatch 경보 생성 시 경보 상태가 변경될 때 알림을 보내도록 Amazon SNS 주제를 포함할 수 있습니다.
3. 지정된 CloudWatch 경보와 일치하는 규칙을 EventBridge에서 생성합니다. 각 규칙은 Lambda 함수를 비롯한 여러 대상을 지원합니다. 예를 들어, 사용 가능한 디스크 공간이 부족할 때 시작되는 경보를 정의하여 EventBridge 규칙을 통해 Lambda 함수를 트리거하여 공간을 정리할 수 있습니다. EventBridge 대상에 대한 자세한 내용은 EventBridge targets를 참조하세요.

리소스

관련 Well-Architected 모범 사례:

- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP02 지표 정의 및 계산(집계)
- REL12-BP01 플레이북을 사용하여 장애 조사

관련 문서:

- Amazon CloudWatch

- CloudWatch Logs insights
- Amazon CloudWatch 경보 사용
- Amazon CloudWatch 대시보드 사용
- Amazon CloudWatch 지표 사용
- Setting up Amazon SNS notifications
- CloudWatch 이상 탐지
- CloudWatch Logs data protection
- Amazon EventBridge
- Amazon Simple Notification Service

관련 비디오:

- reinvent 2022 observability 동영상
- AWS re:Invent 2022 - Observability best practices at Amazon

관련 예제:

- One Observability 워크숍
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 응답 자동화(실시간 처리 및 경보)

이벤트가 감지되면 자동화를 사용하여 실패한 구성 요소를 대체하는 등의 조치를 취합니다.

경보의 자동화된 실시간 처리가 구현되어 경보가 트리거될 때 시스템이 신속한 시정 조치를 취하고 장애 또는 서비스 저하를 방지할 수 있습니다. 경보에 대한 자동 대응에는 장애가 발생한 구성 요소 교체, 컴퓨팅 용량 조정, 정상적인 호스트, 가용 영역 또는 기타 리전으로 트래픽 리디렉션, 운영자에게 알림 등이 포함될 수 있습니다.

원하는 성과: 실시간 경보를 식별하고 서비스 수준 목표 및 서비스 수준에 관한 계약(SLA)을 유지하기 위해 적절한 조치를 취하도록 경보 자동 처리를 설정합니다. 자동화는 단일 구성 요소의 자가 복구 작업부터 전체 사이트 장애 조치에 이르기까지 다양합니다.

일반적인 안티 패턴:

- 주요 실시간 경보의 명확한 인벤토리 또는 카탈로그가 없습니다.

- 핵심 경보에 대한 자동 응답이 없습니다(예: 컴퓨팅이 거의 고갈될 때 자동 규모 조정 시행).
- 경보의 대응 조치가 상충합니다.
- 운영자가 경고 알림을 받을 때 따라야 하는 표준 운영 절차(SOP)가 없습니다.
- 구성 변경을 모니터링하지 않습니다(감지되지 않은 구성 변경으로 인해 워크로드에 다운타임이 발생할 수 있음).
- 의도하지 않은 구성 변경을 취소할 전략이 없습니다.

이 모범 사례 확립의 이점: 경보 처리를 자동화하면 시스템 복원력을 개선할 수 있습니다. 시스템이 자동으로 시정 조치를 취하므로 사람이 개입하여 오류가 발생하기 쉬운 수동 작업이 줄어듭니다. 워크로드 운영이 가용성 목표를 충족하고 서비스 중단을 줄입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

알림을 효과적으로 관리하고 대응을 자동화하려면 중요도와 영향을 기준으로 알림을 분류하고, 대응 절차를 문서화하며, 작업에 순위를 매기기 전에 대응을 계획합니다.

구체적인 조치가 필요한 작업을 식별하고(런북에 자세히 설명되어 있는 경우가 많음), 모든 런북과 플레이북을 검토하여 자동화할 수 있는 작업을 결정합니다. 작업을 정의할 수 있는 경우 대개 자동화할 수 있습니다. 작업을 자동화할 수 없는 경우 SOP에 수동 단계를 문서화하고 운영자에게 이 단계를 교육합니다. 알림 대응을 자동화하기 위한 계획을 수립하고 유지할 수 있는 자동화 기회를 위해 수동 프로세스에 지속적으로 검토합니다.

구현 단계

1. 경보 인벤토리 생성: 모든 경보 목록을 가져오려면 AWS CLI에서 Amazon CloudWatch `describe-alarms` 명령을 사용할 수 있습니다. 설정한 경보 수에 따라 페이지 매김을 사용하여 각 직접 호출에 대한 경보의 일부를 검색해야 할 수도 있고, AWS SDK를 사용하여 API 직접 호출을 통해 경보를 가져올 수도 있습니다.
2. 모든 경보 동작 문서화: 수동이든 자동이든 관계없이 모든 경보와 해당 작업으로 런북을 업데이트합니다. AWS Systems Manager에서는 사전 정의된 런북을 제공합니다. 실행서에 대한 자세한 내용은 실행서 작업을 참조하세요. 런북 콘텐츠를 보는 방법에 대한 자세한 내용은 View runbook content를 참조하세요.
3. 경보 작업 설정 및 관리: 작업이 필요한 모든 경보의 경우 CloudWatch SDK를 사용하여 자동화된 작업을 지정합니다. 예를 들어, 경보에 대한 작업을 생성 및 활성화하거나 비활성화하여 CloudWatch 경보를 기반으로 Amazon EC2 인스턴스 상태를 자동으로 변경할 수 있습니다.

Amazon EventBridge를 사용하여 애플리케이션 가용성 문제나 리소스 변경 등의 시스템 이벤트에 자동으로 대응할 수 있습니다. 관심 있는 이벤트와 이벤트가 규칙과 일치할 때 수행할 작업을 표시하는 규칙을 생성할 수 있습니다. 자동으로 시작할 수 있는 작업으로, AWS Lambda 함수 간접 호출, Amazon EC2 `Run Command` 간접 호출, Amazon Kinesis Data Streams에 이벤트 중계, EventBridge를 사용하여 Amazon EC2 자동화 참조 등이 포함될 수 있습니다.

4. 표준 운영 절차(SOP): 애플리케이션 구성 요소를 기반으로 AWS Resilience Hub에서 여러 개의 SOP 템플릿을 권장합니다. 이러한 SOP를 사용하여 경보가 발생한 경우 운영자가 따라야 하는 모든 프로세스를 문서화할 수 있습니다. Resilience Hub의 권장 사항에 따라 SOP를 구성할 수도 있습니다. 이 경우 관련 복원력 정책이 포함된 Resilience Hub 애플리케이션과 해당 애플리케이션에 대한 복원력 평가 내역이 필요합니다. SOP에 대한 추천은 복원력 평가를 통해 작성됩니다.

   Resilience Hub는 Systems Manager와 연동하여 SOP의 기반으로 사용할 수 있는 다양한 SSM 문서를 제공함으로써 SOP의 단계를 자동화합니다. 예를 들어, 은 기존 SSM Automation 문서를 기반으로 디스크 스페이스를 추가하기 위한 SOP를 권장할 수 있습니다.

5. Amazon DevOps Guru를 사용하여 자동화된 작업 수행: Amazon DevOps Guru는 애플리케이션 리소스가 비정상적으로 작동하는지를 자동으로 모니터링하고 표적화된 권장 사항을 제공하여 문제 파악 및 해결 시간을 단축합니다. DevOps Guru를 사용하면 Amazon CloudWatch 지표, AWS Config, AWS CloudFormation, AWS X-Ray를 비롯한 여러 소스에서 운영 데이터 스트림을 거의 실시간으로 모니터링할 수 있습니다. 또한 DevOps Guru를 사용하여 OpsCenter에서 OpsItems를 자동으로 생성하고 추가 자동화를 위해 EventBridge에 이벤트를 전송할 수 있습니다.

리소스

관련 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)
- REL08-BP01 배포와 같은 표준 활동에 런북 사용

관련 문서:

- AWS Systems Manager 자동화
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- One Observability 워크숍

- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- What is Amazon DevOps Guru?
- 자동화 문서(플레이북) 작업

관련 비디오:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

관련 예제:

- Amazon CloudWatch 및 Systems Manager 워크숍

REL06-BP05 로그 분석

로그 파일 및 지표 기록을 수집하고 이를 분석하여 더 광범위한 추세 및 워크로드 인사이트를 확보합니다.

Amazon CloudWatch 로그 인사이트는 로그 데이터를 분석하는 데 사용할 수 있는 단순하지만 강력한 쿼리 언어를 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. 자세한 내용은 Amazon Athena 사용 설명서의 지원되는 SerDes 및 데이터 형식을 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 여러 경로에 이 수명 주기 관리 기능

을 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

- CloudWatch 로그 인사이트를 사용하면 Amazon CloudWatch Logs 내 로그 데이터를 대화식으로 검색해 분석할 수 있습니다.
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights Sample Queries
- Amazon CloudWatch Logs를 사용하여 Amazon S3으로 로그를 전송한 후, 로그 데이터를 사용하거나 Amazon Athena로 데이터를 쿼리할 수 있습니다.
  - Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 하나요?
    - 서버 액세스 로그 버킷에 대한 S3 수명 주기 정책을 생성합니다. 주기적으로 로그 파일을 제거하도록 수명 주기 정책을 구성하십시오. 그러면 Athena에서 쿼리마다 분석하는 데이터의 양이 줄어듭니다.
      - S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 하나요?

리소스

관련 문서:

- Amazon CloudWatch Logs Insights Sample Queries
- Analyzing Log Data with CloudWatch Logs Insights
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- S3 버킷에 대한 수명 주기 정책을 생성하려면 어떻게 해야 하나요?
- Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 하나요?
- One Observability 워크숍
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측

REL06-BP06 모니터링 범위 및 지표를 정기적으로 검토

워크로드 모니터링이 구현되는 방법을 자주 검토하고 워크로드와 아키텍처가 변화함에 따라 업데이트합니다. 모니터링을 정기적으로 감사하면 문제 지표를 놓치거나 간과할 위험을 줄이고 워크로드가 가용성 목표를 달성하는 데 도움이 됩니다.

효과적인 모니터링은 주요 비즈니스 지표에 기반을 두고 있으며, 이는 비즈니스 우선순위가 변경될 때 변화합니다. 모니터링 검토 프로세스는 서비스 수준 지표(SLI)를 강조하고 인프라, 애플리케이션, 클라이언트 및 사용자의 인사이트를 통합해야 합니다.

원하는 성과: 정기적으로, 그리고 중요한 이벤트 또는 변경 사항이 발생한 후에 검토 및 업데이트되는 효과적인 모니터링 전략이 있습니다. 워크로드 및 비즈니스 요구 사항이 진화함에 따라 주요 애플리케이션 상태 지표가 여전히 관련이 있는지 확인합니다.

일반적인 안티 패턴:

- 기본 지표만 수집합니다.
- 모니터링 전략을 설정하지만 검토하지는 않습니다.
- 주요 변경 사항이 배포될 때 모니터링에 대해 논의하지 않습니다.
- 오래된 지표를 신뢰하여 워크로드 상태를 판단합니다.
- 운영 팀은 오래된 지표와 임계값으로 인해 발생하는 오탐지로 업무 부담이 큽니다.
- 모니터링되지 않는 애플리케이션 구성 요소를 관찰할 수 없습니다.
- 모니터링에서 비즈니스 지표를 제외하고 하위 수준의 기술 지표에만 중점을 둡니다.

이 모범 사례 확립의 이점: 모니터링을 정기적으로 검토하면 잠재적 문제를 예측하고 감지할 수 있는지 확인할 수 있습니다. 또한 이전 검토 중에 놓쳤을 수 있는 사각지대를 발견할 수 있으므로 문제를 감지하는 능력이 더욱 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

운영 준비도 검토(ORR) 프로세스 중에 모니터링 지표와 범위를 검토합니다. 일관된 일정에 따라 정기적인 운영 준비도 상태 검토를 수행하여 현재 워크로드와 구성한 모니터링 사이에 격차가 있는지 평가합니다. 운영 성능 검토 및 지식 공유를 위한 정기 케이던스를 설정하여 운영 팀의 성과를 개선하는 역량을 발전시킵니다. 기존 알림 임계값이 여전히 적절한지 확인하고 운영 팀이 오탐지 알림을 수신하거나 모니터링해야 하는 애플리케이션의 측면을 모니터링하지 않는 상황을 확인합니다.

Resilience Analysis Framework는 프로세스를 탐색하는 데 도움이 되는 유용한 지침을 제공합니다. 프레임워크의 초점은 잠재적 장애 모드와 그 영향을 완화하는 데 사용할 수 있는 예방 및 수정을 위한 제어 조치를 식별하는 것입니다. 이 지식은 모니터링 및 알림에 적합한 지표와 이벤트를 식별하는 데 도움이 될 수 있습니다.

구현 단계

1. 워크로드 대시보드에 대한 정기적인 검토를 예약하고 수행합니다. 검사하는 세부 수준을 나타내는 다양한 케이던스를 구성할 수 있습니다.
2. 지표에서 추세가 나타나는지 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.
3. 평균 또는 중앙값으로 마스킹할 수 있는 지표의 이상치 및 이상을 검사합니다. 기간 중 가장 높은 값과 가장 낮은 값을 살펴보고 정상 범위를 훨씬 벗어난 관찰의 원인을 조사합니다. 이러한 원인을 계속 제거하면 워크로드 성능의 일관성 향상에 따라 예상 지표 범위를 좁힐 수 있습니다.
4. 동작에 급격한 변화가 있는지 알아봅니다. 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.
5. 현재 모니터링 전략이 애플리케이션과 관련이 있는지 검토합니다. 이전 인시던트 분석(또는 Resilience Analysis Framework)을 기반으로 모니터링 범위에 추가로 포함해야 하는 측면이 있는지 평가합니다.
6. 실제 사용자 모니터링(RUM) 지표를 검토하여 애플리케이션 기능 적용 범위에 격차가 있는지 확인합니다.
7. 변경 관리 프로세스를 검토합니다. 필요한 경우 절차를 업데이트하여 변경을 승인하기 전에 수행해야 하는 모니터링 분석 단계를 포함합니다.
8. 운영 준비도 검토 및 오류 프로세스 수정의 일환으로 모니터링 검토를 구현합니다.

리소스

관련 모범 사례

- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP02 지표 정의 및 계산(집계)
- REL06-BP07 시스템을 통한 요청의 엔드 투 엔드 추적 모니터링
- REL12-BP02 인시던트 사후 분석 수행
- REL12-BP06 정기적으로 게임 데이 진행

관련 문서:

- Why you should develop a correction of error (COE)
- Amazon CloudWatch 대시보드 사용
- 운영 가시성을 위한 대시보드 구축
- Advanced Multi-AZ Resilience Patterns - Gray failures
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability 워크숍
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- Amazon CloudWatch 대시보드 사용
- AWS Observability Best Practices
- Resilience analysis framework
- Resilience Analysis Framework - Observability
- Operational Readiness Review - ORR

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 성과: 모든 구성 요소를 포괄적으로 추적할 수 있는 워크로드는 디버깅하기 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.
- 가상 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 서드파티 추적 도구가 모두 포함되지만 단일 추적 솔루션을 선택하고 완전히 통합하는 단계는 아직 수행하지 않았습니다. 선택한 추적 솔루션에 따라 클라우드 네이티브 추적 SDK를 사용하여 클라우드 네이티브가 아닌 구성 요소를 측정하거나 서드파티 도구를 클라우드 네이티브 추적 원격 측정을 수집하도록 구성해야 합니다.

이 모범 사례 확립의 이점: 개발 팀은 문제에 대한 경고를 받으면 구성 요소별 로깅, 성능, 장애와의 상관관계를 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 않을 경우 노출되는 위험 수준: 중간

구현 가이드

분산된 애플리케이션을 운영하는 팀은 추적 도구를 사용하여 상관관계 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 수준에 관한 계약과 목표에 대해 시스템 성능을 정확하게 평가할 수 있도록 엔드 투 엔드 추적 구성에 가상 canary와 실제 사용자 모니터링을 포함하여 원격 클라이언트 상호 작용과 지연 시간을 측정합니다.

AWS X-Ray 및 Amazon CloudWatch 애플리케이션 모니터링 계측 서비스를 사용하여 애플리케이션에서 요청이 이동할 때 해당 요청을 전체적으로 파악할 수 있습니다. X-Ray는 애플리케이션 원격 측정을 수집합니다. 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 및 경보와 통합하는 Servicelens가 포함되어 있습니다. 또한 CloudWatch 애플리케이션 모니터링에는 엔드포인트와 API를 모니터링하기 위한 가상 및 웹 애플리케이션 클라이언트를 측정하기 위한 실제 사용자 모니터링도 포함됩니다.

구현 단계

- Amazon S3, AWS Lambda, Amazon API Gateway와 같은 지원되는 모든 기본 서비스에서 AWS X-Ray를 사용합니다. 이러한 AWS 서비스에서는 코드형 인프라, AWS SDK 또는 AWS Management Console을 사용하여 구성 토글을 통해 X-Ray가 활성화됩니다.
- 애플리케이션 AWS Distro for Open Telemetry 및 X-Ray 또는 서드파티 수집 에이전트를 계측합니다.

- 프로그래밍 언어별 구현에 대한 자세한 내용은 AWS X-Ray 개발자 안내서를 검토하세요. 이러한 설명서 섹션에서는 애플리케이션 프로그래밍 언어와 관련된 HTTP 요청, SQL 쿼리 및 기타 프로세스의 계측 방법을 자세히 설명합니다.
- Amazon CloudWatch Synthetic Canary 및 Amazon CloudWatch RUM에 X-Ray 추적을 사용하여 최종 사용자 클라이언트에서 다운스트림 AWS 인프라를 통과하는 요청 경로를 분석합니다.
- 팀이 문제에 대해 빠르게 경고를 받은 후 ServiceLens를 사용하여 추적 및 서비스 맵을 심층적으로 분석할 수 있도록 리소스 상태와 canary 원격 측정을 기반으로 CloudWatch 지표 및 경보를 구성합니다.
- 기본 추적 솔루션에 서드파티 도구를 사용하는 경우 Datadog, New Relic 또는 Dynatrace와 같은 서드파티 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 무엇입니까?
- Amazon CloudWatch: 애플리케이션 모니터링
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Amazon Builders' Library: 운영 가시성을 위한 분산 시스템 계측
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry 및 AWS X-Ray
- Amazon CloudWatch: 가상 모니터링 사용
- Amazon CloudWatch: CloudWatch RUM 사용
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS

관련 예제:

- One Observability 워크숍

관련 비디오:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

관련 도구:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. 수요 변화에 적응하도록 워크로드를 설계하려면 어떻게 해야 하나요?

확장 가능한 워크로드는 리소스를 자동으로 추가 또는 제거할 수 있는 탄력성을 제공하여 리소스 공급이 특정 시점의 수요와 거의 일치하도록 합니다.

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

클라우드에서 신뢰성의 초석은 인프라 및 리소스의 프로그래밍 방식 정의, 프로비저닝 및 관리입니다. 자동화를 통해 리소스 프로비저닝을 간소화하고, 일관되고 안전한 배포를 촉진하며, 전체 인프라에서 리소스를 확장할 수 있습니다.

원하는 성과: 코드형 인프라(IaC)를 관리합니다. 버전 제어 시스템(VCS)에서 인프라 코드를 정의하고 유지 관리합니다. AWS 리소스 프로비저닝을 자동화된 메커니즘에 위임하고 Application Load Balancer(ALB), Network Load Balancer(NLB) 및 Auto Scaling 그룹과 같은 관리형 서비스를 활용합니다. 지속적 통합/지속적 전송(CI/CD) 파이프라인을 사용하여 리소스를 프로비저닝하면 코드 변경이 Auto Scaling 구성에 대한 업데이트를 포함하여 리소스 업데이트를 자동으로 시작합니다.

일반적인 안티 패턴:

- 명령줄을 사용하거나 AWS Management Console(클릭 작업이라고도 함)에서 리소스를 수동으로 배포합니다.
- 애플리케이션 구성 요소 또는 리소스를 긴밀하게 결합하고 결과적으로 유연하지 않은 아키텍처를 생성합니다.
- 변화하는 비즈니스 요구 사항, 트래픽 패턴 또는 새로운 리소스 유형에 맞춰 조정되지 않는, 유연하지 않은 크기 조정 정책을 구현합니다.
- 예상 수요를 충족하기 위해 용량을 수동으로 추정합니다.

이 모범 사례 확립의 이점: 코드형 인프라(IaC)를 사용하면 인프라를 프로그래밍 방식으로 정의할 수 있습니다. 이렇게 하면 동일한 소프트웨어 개발 수명 주기를 통해 애플리케이션 변경과 동일하게 인프라 변경을 관리할 수 있으므로 일관성과 반복성을 높이고 오류가 발생하기 쉬운 수동 작업의 위험을 줄일 수 있습니다. 자동화된 전송 파이프라인으로 IaC를 구현하여 리소스를 프로비저닝하고 업데이트하는 프로세스를 더욱 간소화할 수 있습니다. 인프라 업데이트를 수동 개입 없이 안정적이고 효율적으로 배포할 수 있습니다. 이러한 민첩성은 변동하는 수요에 맞게 리소스 크기를 조정할 때 특히 중요합니다.

IaC 및 전송 파이프라인과 함께 동적이고 자동화된 리소스 크기 조정을 달성할 수 있습니다. Auto Scaling은 주요 지표를 모니터링하고 사전 정의된 크기 조정 정책을 적용하여 필요에 따라 리소스를 자동으로 프로비저닝하거나 프로비저닝을 해제할 수 있으므로 성능과 비용 효율성이 향상됩니다. 이렇게 하면 애플리케이션 또는 워크로드 요구 사항의 변경에 대한 대응으로 수동 오류 또는 지연의 가능성이 줄어듭니다.

IaC, 자동 전송 파이프라인 및 Auto Scaling의 조합을 통해 조직은 환경을 자신 있게 프로비저닝, 업데이트 및 확장할 수 있습니다. 이 자동화는 응답성과 복원력이 뛰어나며 효율적으로 관리되는 클라우드 인프라를 유지 관리하는 데 필수적입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS 아키텍처의 CI/CD 파이프라인 및 코드형 인프라(IaC)를 사용하여 자동화를 설정하려면 Git과 같은 버전 관리 시스템을 선택하여 IaC 템플릿 및 구성을 저장합니다. 이러한 템플릿은 AWS CloudFormation과 같은 도구를 사용하여 작성할 수 있습니다. 시작하려면 이러한 템플릿 내에서 인프라 구성 요소(예: AWS VPC, Amazon EC2 Auto Scaling 그룹 및 Amazon RDS 데이터베이스)를 정의합니다.

다음으로 이러한 IaC 템플릿을 CI/CD 파이프라인과 통합하여 배포 프로세스를 자동화합니다. AWS CodePipeline은 원활한 AWS 네이티브 솔루션을 제공하며, 다른 서드파티 CI/CD 솔루션을 사용할 수도 있습니다. 버전 관리 리포지토리가 변경될 때 활성화되는 파이프라인을 생성합니다. IaC 템플릿을 린팅하고 검증하는 단계를 포함하도록 파이프라인을 구성하고, 인프라를 스테이징 환경에 배포하고, 자동 테스트를 실행하고, 마지막으로 프로덕션에 배포합니다. 필요한 경우 승인 단계를 포함하여 변경 사항에 대한 관리를 유지합니다. 이 자동화된 파이프라인은 배포 속도를 높일 뿐만 아니라 환경 전반에서 일관성과 신뢰성을 높입니다.

필요에 따라 자동 스테일 아웃 및 스케일 인을 제공하도록 Amazon EC2 인스턴스, Amazon ECS 작업, IaC의 데이터베이스 복제본과 같은 리소스의 오토 스케일링을 구성합니다. 이 접근 방식은 수요에 따라 리소스 크기를 동적으로 조정하여 애플리케이션 가용성과 성능을 개선하고 비용을 최적화합니다. 지원되는 리소스 목록은 Amazon EC2 Auto Scaling 및 AWS Auto Scaling 섹션을 참조하세요.

구현 단계

1. 소스 코드 리포지토리를 생성하고 사용하여 인프라 구성을 제어하는 코드를 저장합니다. 이 리포지토리에 변경 사항을 커밋하여 원하는 지속적인 변경 사항을 반영합니다.
2. AWS CloudFormation과 같은 코드형 인프라 솔루션을 선택하여 인프라를 최신 상태로 유지하고 의도한 상태에서 불일치(드리프트)를 감지합니다.
3. IaC 플랫폼을 CI/CD 파이프라인과 통합하여 배포를 자동화합니다.
4. 리소스의 자동 크기 조정에 적합한 지표를 결정하고 수집합니다.
5. 워크로드 구성 요소에 적합한 스케일 아웃 및 스케일 인 정책을 사용하여 리소스의 자동 크기 조정을 구성합니다. 예측 가능한 사용 패턴을 위해 예약된 크기 조정을 사용하는 것을 고려해 보세요.
6. 배포를 모니터링하여 장애 및 회귀를 감지합니다. CI/CD 플랫폼 내에서 롤백 메커니즘을 구현하여 필요한 경우 변경 사항을 되돌립니다.

리소스

관련 문서:

- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: Auto Scaling과 함께 사용할 수 있는 제품
- DynamoDB Auto Scaling을 사용하여 자동으로 처리량 용량 관리
- Using a load balancer with an Auto Scaling group
- What Is AWS Global Accelerator?
- What Is Amazon EC2 Auto Scaling?

- AWS Auto Scaling란 무엇입니까?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- Elastic Load Balancing이란 무엇인가요?
- Network Load Balancer란 무엇인가요?
- Application Load Balancer란 무엇인가요?
- Integrating Jenkins with AWS CodeBuild and AWS CodeDeploy
- Creating a four stage pipeline with AWS CodePipeline

관련 비디오:

- Back to Basics: Deploy Your Code to Amazon EC2
- AWS Supports You | Starting Your Infrastructure as Code Solution Using AWS CloudFormation Templates
- Streamline Your Software Release Process Using AWS CodePipeline
- Monitor AWS Resources Using Amazon CloudWatch Dashboards
- Create Cross Account & Cross Region CloudWatch Dashboards | Amazon Web Services

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 리소스 규모를 조정하도록 알리거나 자동화를 시작하여 자동으로 리소스 규모를 조정합니다.

워크로드에 맞게 수동으로 규모를 조정할 수 있습니다. 예를 들어 AWS Management Console 또는 AWS CLI를 통해 DynamoDB 테이블의 처리량을 수정하거나 Auto Scaling 그룹의 EC2 인스턴스 수를 변경합니다. 하지만 가능하면 자동화를 사용해야 합니다(리소스를 확보하거나 조정할 때 자동화 사용 참조).

원하는 성과: 장애 또는 고객 경험 저하가 감지되면 가용성을 복원하기 위해 규모 조정 활동(자동 또는 수동)이 시작됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

워크로드의 모든 구성 요소에 대한 관찰성 및 모니터링을 구현하여 고객 경험을 모니터링하고 장애를 감지합니다. 필요한 리소스의 규모를 조정하는 절차를 수동 또는 자동으로 정의합니다. 자세한 내용은 REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지를 참조하세요.

구현 단계

- 필요한 리소스 규모를 조정하는 절차를 수동 또는 자동으로 정의합니다.
  - 규모 조정 절차는 워크로드 내의 다양한 구성 요소가 어떻게 설계되었는지에 따라 달라집니다.
  - 규모 조정 절차는 사용되는 기본 기술에 따라서도 달라집니다.
    - AWS Auto Scaling을 사용하는 구성 요소는 규모 조정 계획을 사용하여 리소스 규모 조정을 위한 일련의 지침을 구성할 수 있습니다. AWS CloudFormation을 사용하거나 AWS 리소스에 태그를 추가하는 경우 애플리케이션별로 다양한 리소스 세트에 대한 크기 조정 계획을 설정할 수 있습니다. Auto Scaling은 각 리소스별로 맞춤화된 조정 전략에 대한 권장 사항을 제공합니다. 규모 조정 계획을 생성하면 Auto Scaling이 동적 규모 조정과 예측 규모 조정 방식을 결합하여 규모 조정 전략을 지원합니다. 자세한 내용은 How scaling plans work를 참조하세요.
    - Amazon EC2 Auto Scaling을 사용하면 애플리케이션의 로드를 처리할 수 있는 정확한 수의 Amazon EC2 인스턴스를 유지할 수 있습니다. Auto Scaling 그룹이라는 EC2 인스턴스 모음을 생성합니다. 각 Auto Scaling 그룹의 최소 및 최대 인스턴스 수를 지정할 수 있으며 Amazon EC2 Auto Scaling은 그룹이 이러한 한도에 미달하거나 한도를 초과하지 않도록 합니다. 자세한 내용은 What is Amazon EC2 Auto Scaling?을 참조하세요.
    - Amazon DynamoDB Auto Scaling은 Application Auto Scaling 서비스를 사용하여 프로비저닝된 처리 능력을 실제 트래픽 패턴에 따라 사용자 대신 동적으로 조정합니다. 따라서 테이블 또는 글로벌 보조 인덱스에 따라 할당된 읽기 및 쓰기 용량을 늘려 병목 현상 없이 갑작스러운 트래픽 증가를 처리할 수 있습니다. 자세한 내용은 DynamoDB Auto Scaling을 사용하여 자동으로 처리량 용량 관리를 참조하세요.

리소스

관련 모범 사례:

- REL07-BP01 리소스를 확보하거나 조정할 때 자동화 사용
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

관련 문서:

- AWS Auto Scaling: How Scaling Plans Work
- DynamoDB Auto Scaling을 사용하여 자동으로 처리량 용량 관리
- What Is Amazon EC2 Auto Scaling?

REL07-BP03 워크로드에 더 많은 리소스가 필요한 것으로 감지되면 리소스 확보

클라우드 컴퓨팅의 가장 중요한 기능 중 하나는 리소스를 동적으로 프로비저닝하는 기능입니다.

기존 온프레미스 컴퓨팅 환경에서는 피크 수요를 충족하기에 충분한 용량을 미리 식별하고 프로비저닝해야 합니다. 이는 비용이 많이 들고 워크로드의 피크 용량 요구 사항을 과소평가할 경우 가용성에 위험을 초래하기 때문에 문제가 됩니다.

클라우드에서는 이렇게 할 필요가 없습니다. 필요에 따라 컴퓨팅, 데이터베이스 및 기타 리소스 용량을 프로비저닝하여 현재 및 예상 수요를 충족할 수 있습니다. Amazon EC2 Auto Scaling 및 Application Auto Scaling과 같은 자동화된 솔루션은 지정한 지표를 기반으로 리소스를 온라인 상태로 만들 수 있습니다. 이렇게 하면 규모 조정 프로세스가 더 쉽고 예측 가능하며, 항상 충분한 리소스를 사용할 수 있도록 하여 워크로드의 신뢰성을 크게 높일 수 있습니다.

원하는 성과: 수요에 맞게 컴퓨팅 및 기타 리소스의 자동 크기 조정을 구성합니다. 크기 조정 정책에 충분한 여유를 제공하여 추가 리소스를 온라인 상태로 가져오는 동안 트래픽 버스트를 허용합니다.

일반적인 안티 패턴:

- 확장 가능한 리소스를 일정 개수로 프로비저닝합니다.
- 실제 수요와 상관관계가 없는 크기 조정 지표를 선택합니다.
- 크기 조정 계획에 수요 버스트를 수용할 수 있는 충분한 여유를 제공하지 못합니다.
- 크기 조정 정책은 용량을 너무 늦게 추가하여 추가 리소스를 온라인 상태로 전환하는 동안 용량 소진 및 서비스 저하로 이어집니다.
- 최소 및 최대 리소스 수를 올바르게 구성하지 못하여 조정에 실패합니다.

이 모범 사례 확립의 이점: 워크로드의 고가용성을 제공하고 정의된 서비스 수준 목표(SLO)를 준수하려면 현재 수요를 충족할 수 있는 충분한 리소스를 확보하는 것이 중요합니다. 자동 크기 조정을 사용하면 현재 및 예측된 수요를 제공하기 위해 워크로드에 필요한 적절한 양의 컴퓨팅, 데이터베이스 및 기타 리소스를 제공할 수 있습니다. 피크 용량 요구 사항을 결정하고 리소스를 정적으로 할당하여 제공할 필요가 없습니다. 대신 수요가 증가함에 따라 더 많은 리소스를 할당하여 수용하고 수요가 감소한 후에는 리소스를 비활성화하여 비용을 절감할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

먼저 워크로드 구성 요소가 자동 크기 조정에 적합한지 확인합니다. 이러한 구성 요소는 동일한 리소스를 제공하고 동일하게 작동하기 때문에 수평 크기 조정이 가능하다고 합니다. 수평 크기 조정이 가능한 구성 요소의 예로는 동일하게 구성된 EC2 인스턴스, Amazon Elastic Container Service(Amazon ECS) 작업, Amazon Elastic Kubernetes Service(Amazon EKS)에서 실행되는 포드 등이 있습니다. 이러한 컴퓨팅 리소스는 일반적으로 로드 밸런서 뒤에 위치하며 복제본 이라고 합니다.

복제된 다른 리소스에는 데이터베이스 읽기 전용 복제본, Amazon DynamoDB 테이블 및 Amazon ElastiCache(Redis OSS) 클러스터가 포함될 수 있습니다. 지원되는 리소스의 전체 목록은 AWS services that you can use with Application Auto Scaling을 참조하세요.

컨테이너 기반 아키텍처의 경우 두 가지 방법으로 크기를 조정해야 할 수 있습니다. 먼저 수평적으로 크기 조정이 가능한 서비스를 제공하는 컨테이너의 크기를 조정해야 할 수 있습니다. 둘째, 컴퓨팅 리소스를 확장하여 새 컨테이너를 위한 공간을 확보해야 할 수 있습니다. 각 계층마다 다양한 자동 크기 조정 메커니즘이 있습니다. ECS 작업의 크기를 조정하려면 Application Auto Scaling을 사용할 수 있습니다. Kubernetes 포드의 크기를 조정하려면 Horizontal Pod Autoscaler(HPA) 또는 Kubernetes Event-driven Autoscaling(KEDA)을 사용할 수 있습니다. 컴퓨팅 리소스를 확장하려면 ECS용 또는 Kubernetes용 용량 공급자를 사용하거나 Karpenter 또는 Cluster Autoscaler를 사용할 수 있습니다.

그런 다음 자동 크기 조정을 수행하는 방법을 선택합니다. 지표 기반 크기 조정, 예약된 크기 조정, 예측 크기 조정의 세 가지 주요 옵션이 있습니다.

지표 기반 크기 조정

지표 기반 크기 조정은 하나 이상의 크기 조정 지표 값을 기반으로 리소스를 프로비저닝합니다. 크기 조정 지표는 워크로드의 수요에 해당하는 지표입니다. 적절한 크기 조정 지표를 결정하는 좋은 방법은 비프로덕션 환경에서 로드 테스트를 수행하는 것입니다. 로드 테스트 중에 크기 조정 가능한 리소스 수를 고정하고 수요(예: 처리량, 동시성 또는 시뮬레이션된 사용자)를 천천히 증가시킵니다. 그런 다음 수요가 증가함에 따라 증가(또는 감소)하고 수요가 감소하면 반대로 감소(또는 증가)하는 지표를 찾습니다. 일반적인 크기 조정 지표에는 CPU 사용률, 작업 대기열 깊이(예: Amazon SQS 대기열), 활성 사용자 수 및 네트워크 처리량이 포함됩니다.

> **Note**
>
> AWS는 대부분의 애플리케이션에서 애플리케이션이 워밍업된 다음 일정한 값에 도달하면 메모리 사용률이 증가한다는 것을 확인했습니다. 수요가 감소하면 일반적으로 메모리 사용률이 병렬로 감소하지 않고 상승 상태로 유지됩니다. 메모리 사용률은 두 방향의 수요에 상응하지

않기 때문에, 즉 수요에 따라 증가하거나 감소하지 않기 때문에 자동 크기 조정을 위해 이 지표를 선택하기 전에 신중하게 고려하세요.

지표 기반 크기 조정은 잠정적인 작업입니다. 사용률 지표가 오토 스케일링 메커니즘으로 전파되는 데 몇 분 정도 걸릴 수 있으며, 이러한 메커니즘은 일반적으로 수요 증가의 명확한 신호를 기다렸다가 반응합니다. 그런 다음 오토 스케일러가 새 리소스를 생성하므로 완전히 서비스를 제공하는 데 시간이 더 걸릴 수 있습니다. 따라서 크기 조정 지표 목표를 전체 사용률에 너무 가깝게(예: CPU 사용률의 90%) 설정하지 않는 것이 중요합니다. 전체 사용률에 가깝게 설정하면 추가 용량이 온라인 상태가 되기 전에 기존 리소스 용량이 소진될 위험이 있습니다. 일반적으로 최적의 가용성을 위한 리소스 사용률 목표는 수요 패턴 및 추가 리소스를 프로비저닝하는 데 필요한 시간에 따라 50~70% 범위일 수 있습니다.

예약된 크기 조정

예약된 크기 조정은 달력 또는 시간대에 따라 리소스를 프로비저닝하거나 제거합니다. 주중 영업 시간 또는 세일 이벤트 중 피크 사용률과 같이 예측 가능한 수요가 있는 워크로드에 자주 사용됩니다. Amazon EC2 Auto Scaling과 Application Auto Scaling 모두 예약된 크기 조정을 지원합니다. KEDA의 cron scaler는 Kubernetes 포드의 예약된 크기 조정을 지원합니다.

예측 크기 조정

예측 크기 조정은 기계 학습을 사용하여 예상 수요에 따라 리소스 크기를 자동으로 조정합니다. 예측 크기 조정은 제공하는 사용률 지표의 과거 값을 분석하고 미래 값을 지속적으로 예측합니다. 그런 다음 예측 값을 사용하여 리소스를 확장하거나 축소합니다. Amazon EC2 Auto Scaling은 예측 크기 조정을 수행할 수 있습니다.

구현 단계

1. 워크로드 구성 요소가 자동 크기 조정에 적합한지 확인합니다.
2. 지표 기반 크기 조정, 예약된 크기 조정 또는 예측 크기 조정 중에서 워크로드에 가장 적합한 크기 조정 메커니즘을 결정합니다.
3. 구성 요소에 적합한 자동 크기 조정 메커니즘을 선택합니다. Amazon EC2 인스턴스의 경우 Amazon EC2 Auto Scaling을 사용합니다. 다른 AWS 서비스의 경우 Application Auto Scaling을 사용합니다. Kubernetes 포드(예: Amazon EKS 클러스터에서 실행되는 포드)의 경우 Horizontal Pod Autoscaler(HPA) 또는 Kubernetes Event-driven Autoscaling(KEDA)을 고려합니다. Kubernetes 또는 EKS 노드의 경우 Karpenter 및 Cluster Auto Scaler(CAS)를 고려합니다.
4. 지표 또는 예약된 크기 조정의 경우 로드 테스트를 수행하여 워크로드에 적합한 크기 조정 지표와 목표 값을 결정합니다. 예약된 크기 조정의 경우 선택한 날짜 및 시간에 필요한 리소스 수를 결정합니다. 예상 피크 트래픽을 처리하는 데 필요한 최대 리소스 수를 결정합니다.

5. 위에서 수집한 정보를 기반으로 오토 스케일러를 구성합니다. 자세한 내용은 오토 스케일링 서비스의 설명서를 참조하세요. 최대 및 최소 크기 조정 제한이 올바르게 구성되었는지 확인합니다.
6. 크기 조정 구성이 예상대로 작동하는지 확인합니다. 비프로덕션 환경에서 로드 테스트를 수행하고 시스템이 어떻게 반응하는지 관찰하고 필요에 따라 조정합니다. 프로덕션에서 오토 스케일링을 활성화할 때는 예기치 않은 동작이 발생할 경우 알리도록 적절한 경보를 구성합니다.

리소스

관련 문서:

- What Is Amazon EC2 Auto Scaling?
- AWS 규범적 지침: Load testing applications
- AWS Marketplace: Auto Scaling과 함께 사용할 수 있는 제품
- DynamoDB Auto Scaling을 사용하여 자동으로 처리량 용량 관리
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Telling Stories About Little's Law

REL07-BP04 워크로드 로드 테스트

확장 작업이 워크로드의 필요 사항을 충족하는지 측정하기 위한 로드 테스트 방식을 채택하세요.

지속적인 로드 테스트를 수행하는 것이 중요합니다. 로드 테스트를 통해 한계점을 찾고 워크로드의 성능을 테스트할 수 있습니다. AWS를 사용하면 프로덕션 워크로드의 규모를 모델링하는 임시 테스트 환경을 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 몇분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.

또한 프로덕션에서의 로드 테스트는 프로덕션 시스템에 스트레스가 가해지는 게임 데이의 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 테스트를 수행합니다.
- 전체 워크로드가 아니라 워크로드의 개별 부분에 대해서만 로드 테스트를 수행합니다.

- 대표적인 실제 요청 세트가 아니라 요청의 하위 세트를 사용하여 로드 테스트를 수행합니다.
- 예상 로드보다 작은 안전 계수로 로드 테스트를 수행합니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 중간

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.
  - Distributed Load Testing on AWS: 수천 명의 사용자가 접속한 상황을 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 로드에서는 효과가 다르게 나타나므로 가능한 한 제품 환경에 가깝게 테스트해야 합니다.

리소스

관련 문서:

- Distributed Load Testing on AWS: 수천 명의 사용자가 접속한 상황을 시뮬레이션
- 부하 테스트 애플리케이션

관련 비디오:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

REL 8. 변경은 어떻게 구현하나요?

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 해결하는 것이 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 실행하도록 간접 호출합니다.

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 변경을 수행합니다.
- 더 빠르게 배포하기 위해 계획의 단계를 건너뛰고, 그 결과 배포에 실패합니다.
- 변경 사항 되돌리기를 테스트하지 않고 변경을 수행합니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

- 절차를 런북으로 문서화하면 적절하게 파악한 이벤트에 일관된 방식으로 신속하게 대응할 수 있습니다.
  - AWS Well-Architected Framework: Concepts: Runbook
- 코드형 인프라의 원칙을 사용해 인프라를 정의합니다. AWS CloudFormation 또는 신뢰할 수 있는 서드파티를 사용하여 인프라를 명확히 하면 버전 관리 소프트웨어를 통한 변경 사항의 각 버전을 차례대로 확인할 수 있습니다.
  - AWS CloudFormation(또는 신뢰할 수 있는 서드파티 제품)을 사용하여 인프라를 정의합니다.
    - AWS CloudFormation란 무엇입니까?
  - 우수한 소프트웨어 설계 원칙으로 분리된 단일 템플릿을 생성합니다.
    - 구현 시 권한, 템플릿 및 책임자를 결정합니다.
      - AWS Identity and Access Management를 통한 액세스 제어
    - Git과 같은 인기 있는 기술을 기반으로 하는 호스팅 소스 코드 관리 시스템을 사용하여 소스 코드 및 코드형 인프라(IaC) 구성을 저장합니다.

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: Concepts: Runbook
- AWS CloudFormation란 무엇입니까?

관련 예제:

- Automating operations with Playbooks and Runbooks

REL08-BP02 배포의 일부로 기능 테스트 통합

단위 테스트 및 필수 기능을 검증하는 통합 테스트와 같은 기술을 사용합니다.

유닛 테스트는 코드의 가장 작은 기능 유닛을 테스트하여 동작을 검증하는 프로세스입니다. 통합 테스트에서는 각 애플리케이션 기능이 소프트웨어 요구 사항에 따라 작동하는지 검증합니다. 유닛 테스트는 애플리케이션의 일부를 개별적으로 테스트하는 데 중점을 두지만 통합 테스트는 부작용(예: 변경 작업을 통해 데이터가 변경될 때의 영향)을 고려합니다. 어느 경우든 테스트를 배포 파이프라인에 포함해야 하며, 성공 기준이 충족되지 않으면 파이프라인이 중단되거나 롤백됩니다. 이러한 테스트는 파이프라인에서 프로덕션 전에 준비되는 사전 프로덕션 환경에서 실행됩니다.

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. 애플리케이션이 구축되고 유닛 테스트가 실행됩니다. 유닛 테스트를 통과한 후 구축된 코드를 테스트를 위해 스테이징 서버로 배포합니다. 스테이징 서버에서 CodePipeline은 통합이나 로드 테스트 같은 추가 테스트를 실행합니다. 이러한 테스트가 성공적으로 완료되면 CodePipeline은 테스트되고 승인된 코드를 프로덕션 인스턴스에 배포합니다.

원하는 성과: 자동화를 사용하여 유닛 테스트 및 통합 테스트를 수행하여 코드가 예상대로 작동하는지 검증합니다. 이러한 테스트가 배포 프로세스에 포함되며 테스트 실패 시 배포를 중단합니다.

일반적인 안티 패턴:

- 배포 타임라인을 가속화하기 위해 배포 프로세스 중에 테스트 실패 및 계획을 무시하거나 우회합니다.
- 배포 파이프라인 외부에서 수동으로 테스트를 수행합니다.
- 수동 긴급 워크플로를 통해 자동화의 테스트 단계를 건너뜁니다.
- 프로덕션 환경과 별로 유사하지 않은 환경에서 자동 테스트를 실행합니다.
- 유연성이 충분하지 않고 애플리케이션이 발전함에 따라 유지 관리, 업데이트 또는 크기 조정이 어려운 테스트 세트를 구축합니다.

이 모범 사례 확립의 이점: 배포 프로세스 중 자동 테스트는 문제를 조기에 포착하여 버그 또는 예상치 못한 동작으로 프로덕션으로 릴리스될 위험을 줄입니다. 유닛 테스트에서는 코드 동작이 원하는 대로 작동하는지, API 계약을 준수하는지 확인합니다. 통합 테스트에서는 시스템이 지정된 요구 사항에 따라 작동하는지 확인합니다. 이러한 테스트 유형은 사용자 인터페이스, API, 데이터베이스 및 소스 코드와 같은 구성 요소의 의도된 작동 순서를 확인하는 데 사용됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

테스트 사례를 개발하여 코드를 지정하고 검증하는 테스트 기반 개발(TDD) 접근 방식을 채택합니다. 시작하려면 각 기능에 대한 테스트 사례를 생성합니다. 테스트가 실패하면 새 코드를 작성하여 테스트를 통과합니다. 이 접근 방식은 각 기능의 예상 결과를 검증하는 데 도움이 됩니다. 유닛 테스트를 실행하고 소스 코드 리포지토리에 코드를 커밋하기 전에 전달되는지 확인합니다.

CI/CD 파이프라인의 구축, 테스트 및 배포 단계의 일부로 유닛 및 통합 테스트를 모두 구현합니다. 테스트를 자동화하고 새 버전의 애플리케이션을 배포할 준비가 될 때마다 테스트를 자동으로 시작합니다. 성공 기준이 충족되지 않으면 파이프라인이 중지되거나 롤백됩니다.

애플리케이션이 웹 또는 모바일 앱인 경우 여러 데스크톱 브라우저 또는 실제 디바이스에서 자동 통합 테스트를 수행합니다. 이 접근 방식은 다양한 디바이스에서 모바일 앱의 호환성과 기능을 검증하는 데 특히 유용합니다.

구현 단계

1. 기능 코드(테스트 기반 개발, 즉 TDD)를 작성하기 전에 유닛 테스트를 작성합니다. 유닛 테스트 작성 및 실행이 비기능적 코딩 요구 사항이 되도록 코드 지침을 만듭니다.
2. 식별된 테스트 가능한 기능을 포함하는 자동화된 통합 테스트 세트를 생성합니다. 이러한 테스트는 사용자 상호 작용을 시뮬레이션하고 예상 결과를 검증해야 합니다.
3. 통합 테스트를 실행하는 데 필요한 테스트 환경을 생성합니다. 여기에는 프로덕션 환경을 유사하게 모방한 스테이징 또는 프로덕션 전 환경이 포함될 수 있습니다.
4. AWS CodePipeline 콘솔 또는 AWS Command Line Interface(CLI)를 사용하여 소스, 구축, 테스트 및 배포 단계를 설정합니다.
5. 코드를 구축하고 테스트한 후 애플리케이션을 배포합니다. AWS CodeDeploy는 스테이징(테스트) 및 프로덕션 환경에 배포할 수 있습니다. 이러한 환경에는 Amazon EC2 인스턴스, AWS Lambda 함수 또는 온프레미스 서버가 포함될 수 있습니다. 애플리케이션을 모든 환경에 배포하려면 동일한 배포 메커니즘을 사용해야 합니다.
6. 파이프라인의 진행 상황과 각 단계의 상태를 모니터링합니다. 품질 검사를 사용하여 테스트 상태에 따라 파이프라인을 차단합니다. 파이프라인 단계 장애 또는 파이프라인 완료에 대한 알림을 받을 수도 있습니다.
7. 테스트 결과를 지속적으로 모니터링하고 더 많은 주의가 필요한 패턴, 회귀 또는 영역을 찾습니다. 이 정보를 사용하여 테스트 세트를 개선하고, 더 강력한 테스트가 필요한 애플리케이션 영역을 식별하고, 배포 프로세스를 최적화합니다.

리소스

관련 모범 사례:

- REL07-BP04 워크로드 로드 테스트
- REL08-BP03 배포의 일부로 복원력 테스트 통합
- REL12-BP04 카오스 엔지니어링을 이용한 복원력 테스트

관련 문서:

- AWS Prescriptive Guidance: Test automation
- Continuous Delivery and Continuous Integration
- Indicators for functional testing
- Monitoring pipelines
- Use AWS CodePipeline with AWS CodeBuild to test code and run builds
- AWS Device Farm

REL08-BP03 배포의 일부로 복원력 테스트 통합

장애 시나리오가 발생할 경우에 대비해 시스템에 장애를 의도적으로 도입하여 성능을 측정함으로써 복원력 테스트를 통합합니다. 복원력 테스트는 시스템에서 예상치 못한 장애를 식별하는 데 중점을 두기 때문에 일반적으로 배포 주기에 통합되는 단위 및 기능 테스트와는 다릅니다. 프로덕션 전 단계에서 복원력 테스트 통합을 시작하는 것이 안전하지만, 게임 데이의 일부로 프로덕션 환경에서 이러한 테스트를 구현한다는 목표를 설정하세요.

원하는 성과: 복원력 테스트는 프로덕션에서의 성능 저하를 견디는 시스템의 능력에 대해 확신을 얻는 데 도움이 됩니다. 실험을 통해 장애로 이어질 수 있는 약점을 식별하면 시스템을 개선하여 장애 및 성능 저하를 자동으로 효율적으로 완화할 수 있습니다.

일반적인 안티 패턴:

- 배포 프로세스의 관찰성과 모니터링이 부족합니다.
- 시스템 장애 해결을 위해 사람에게 의존합니다.
- 품질 분석 메커니즘이 열악합니다.
- 시스템의 알려진 문제에 초점을 맞추고, 알려지지 않은 문제를 식별하기 위한 실험은 부족합니다.
- 장애를 식별하지만 해결책은 없습니다.

- 조사 결과 및 런북에 대한 문서화 과정이 없습니다.

이 모범 사례 확립의 이점: 배포에 통합된 복원력 테스트는 시스템 내에서 미처 알아채지 못하다가 프로덕션에서 가동 중단으로 이어질 수 있는 알려지지 않은 문제를 식별하는 데 도움이 됩니다. 시스템에서 알려지지 않은 문제를 식별하면 조사 결과를 문서화하고, 테스트를 CI/CD 프로세스에 통합하며, 효율적이고 반복 가능한 메커니즘을 통해 완화를 간소화하는 런북을 빌드할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

시스템의 배포에 통합할 수 있는 가장 일반적인 복원력 테스트 형식은 재해 복구와 카오스 엔지니어링입니다.

- 중요한 배포 시 재해 복구 계획 및 표준 운영 절차(SOP)에 대한 업데이트를 포함하세요.
- 신뢰성 테스트를 자동화된 배포 파이프라인에 통합하세요. AWS Resilience Hub와 같은 서비스를 CI/CD 파이프라인에 통합하여 배포할 때마다 배포의 일부로 자동으로 평가되는 지속적인 복원력 평가를 설정할 수 있습니다.
- AWS Resilience Hub에서 애플리케이션을 정의하세요. 복원력 평가는 복구 절차를 애플리케이션에 대한 AWS Systems Manager 문서로 생성하고 권장되는 Amazon CloudWatch 모니터 및 경보의 목록을 제공할 수 있는 코드 스니펫을 생성합니다.
- DR 계획과 SOP가 업데이트되면 재해 복구 테스트를 완료하여 효과가 있는지 확인하세요. 재해 복구 테스트는 이벤트 발생 후 시스템을 복원하고 정상 운영 상태로 돌아갈 수 있는지를 결정하는 데 도움이 됩니다. 다양한 재해 복구 전략을 시뮬레이션하고 계획이 가동 시간 요구 사항을 충족하기에 충분한지 확인할 수 있습니다. 일반적인 재해 복구 전략에는 백업 및 복원, 파일럿 라이트, 수동 대기 방식, 예열 대기 방식, 상시 대기 방식, 액티브-액티브가 포함되며 비용과 복잡성이 각기 다릅니다. 재해 복구 테스트 전에 Recovery Time Objective(RTO) 및 Recovery Point Objective(RPO)를 정의하여 시뮬레이션할 전략의 선택지를 간소화하는 것이 좋습니다. AWS는 AWS Elastic Disaster Recovery와 같은 재해 복구 도구를 제공하여 계획과 테스트를 시작하는 데 도움을 줍니다.
- 카오스 엔지니어링 실험은 네트워크 중단 및 서비스 장애와 같은 시스템 장애를 초래합니다. 통제된 장애로 시뮬레이션하면 주입된 장애의 영향을 억제하면서 시스템의 취약성을 발견할 수 있습니다. 다른 전략과 마찬가지로 AWS Fault Injection Service와 같은 서비스를 사용하여 비프로덕션 환경에서 통제된 장애 시뮬레이션을 실행하면 프로덕션에 배포하기 전에 확신을 얻을 수 있습니다.

리소스

관련 문서:

- Experiment with failure using resilience testing to build recovery preparedness
- Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline
- Disaster recovery (DR) architecture on AWS, part 1: Strategies for recovery in the cloud
- Verify the resilience of your workloads using Chaos Engineering
- Principles of Chaos Engineering
- Chaos Engineering 워크숍

관련 비디오:

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Improve Application Resilience with AWS Fault Injection Service
- Prepare & Protect Your Applications From Disruption With AWS Resilience Hub

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 배포 전략을 따라 워크로드 배포의 신뢰성, 일관성 및 재현성을 높입니다.

원하는 성과: 변경 불가능한 인프라를 사용했을 때 워크로드 내에서 인프라 리소스를 실행하기 위한 인플레이스 수정이 허용되지 않습니다. 대신 변경이 필요한 경우 필요한 변경 사항이 모두 포함하도록 업데이트된 신규 인프라 리소스 세트를 기존 리소스와 병렬로 배포합니다. 이 배포는 자동으로 검증되며, 성공하면 트래픽이 점차 새로운 리소스 세트로 이동합니다.

이 배포 전략은 소프트웨어 업데이트, 보안 패치, 인프라 변경, 구성 업데이트, 애플리케이션 업데이트 등에 적용됩니다.

일반적인 안티 패턴:

- 실행 중인 인프라 리소스에 인플레이스 변경을 구현합니다.

이 모범 사례 확립의 이점:

- 환경 간 일관성 향상: 환경 간에 인프라 리소스의 차이가 없으므로 일관성이 향상되고 테스트가 간소화됩니다.

- 구성 드리프트 감소: 인프라 리소스를 버전이 관리되는 알려진 구성으로 대체하면 인프라가 테스트를 거쳐 신뢰할 수 있는 알려진 상태로 설정되므로 구성 드리프트가 발생하지 않습니다.
- 신뢰할 수 있는 원자 단위 배포: 배포가 성공적으로 완료되거나 변경 사항이 없으므로 배포 프로세스의 일관성과 신뢰성이 향상됩니다.
- 간소화된 배포: 업그레이드를 지원할 필요가 없으므로 배포가 간소화됩니다. 업그레이드는 단지 새로운 배포일 뿐입니다.
- 빠른 롤백 및 복구 프로세스를 통해 배포 안전성 개선: 이전 작업 버전이 변경되지 않으므로 배포가 더 안전합니다. 오류가 감지되면 롤백할 수 있습니다.
- 보안 태세 강화: 인프라 변경을 허용하지 않음으로써 원격 액세스 메커니즘(예: SSH)을 비활성화할 수 있습니다. 이렇게 하면 공격 벡터가 줄어들어 조직의 보안 태세를 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

자동화

변경 불가능한 인프라 배포 전략을 정의할 때는 재현성을 높이고 인적 오류 가능성을 최소화하기 위해 최대한 자동화를 사용하는 것이 좋습니다. 자세한 내용은 REL08-BP05 자동화를 통한 변경 사항 배포 및 안전하고 간편한 배포 자동화를 참조하세요.

코드형 인프라(IaC)를 사용하면 인프라 프로비저닝, 오케스트레이션 및 배포 단계가 프로그래밍, 설명 및 선언적 방식으로 정의되고 소스 제어 시스템에 저장됩니다. 코드형 인프라를 활용하면 인프라 배포를 더 간단하게 자동화할 수 있고 인프라 불변성을 달성하는 데 도움이 됩니다.

배포 패턴

워크로드 변경이 필요한 경우 변경 불가능한 인프라 배포 전략에서는 필요한 모든 변경을 포함하여 새로운 인프라 리소스 세트를 배포해야 합니다. 이 새로운 리소스 세트는 사용자에게 미치는 영향을 최소화하는 롤아웃 패턴을 따르는 것이 중요합니다. 이 배포에는 두 가지 주요 전략이 있습니다.

카나리 배포: 일반적으로 단일 서비스 인스턴스(canary)에서 실행되는 새 버전으로 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. 카나리 배포를 허용하는 배포 구성을 사용하여 AWS CodeDeploy를 구성할 수 있습니다.

블루/그린 배포: 카나리 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 CodeDeploy 및 AWS Elastic Beanstalk를 구성할 수 있습니다.

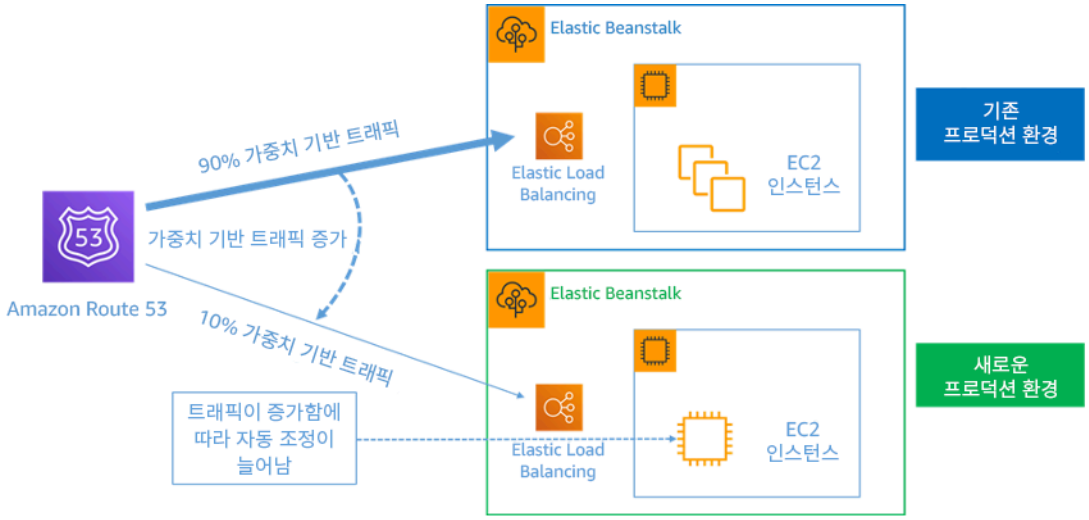


그림 8: AWS Elastic Beanstalk 및 Amazon Route 53을 사용한 블루/그린 배포

드리프트 감지

드리프트는 인프라 리소스의 상태 또는 구성이 예상과 달라지는 모든 변경으로 정의됩니다. 모든 유형의 관리되지 않는 구성 변경은 변경 불가능한 인프라의 개념에 어긋나므로 변경 불가능한 인프라를 성공적으로 구현하려면 이를 감지하고 수정해야 합니다.

구현 단계

- 실행 중인 인프라 리소스의 인플레이스 수정을 허용하지 않습니다.
  - AWS Identity and Access Management(IAM)를 사용하여 AWS에서 서비스 및 리소스에 액세스할 수 있는 사용자 또는 대상을 지정하고, 세분화된 권한을 중앙에서 관리하며, 액세스를 분석하여 AWS에서 권한을 세분화할 수 있습니다.
- 인프라 리소스 배포를 자동화하여 재현성을 높이고 인적 오류 가능성을 최소화합니다.
  - AWS의 DevOps 소개 백서에 설명되어 있는 것처럼 자동화는 AWS 서비스의 초석이며 모든 서비스, 기능 및 오퍼링에서 내부적으로 지원됩니다.
  - Amazon Machine Image(AMI)를 프리베이킹하면 시작 시간을 단축할 수 있습니다. EC2 Image Builder는 사용자 지정되고 안전한 최신 Linux 또는 Windows 사용자 지정 AMI의 생성, 유지 관리, 검증, 공유 및 배포를 자동화하는 데 도움이 되는 완전관리형 AWS 서비스입니다.

- 자동화를 지원하는 서비스의 예를 몇 가지 들자면 다음과 같습니다.
  - AWS Elastic Beanstalk는 Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker를 사용하여 개발된 웹 애플리케이션 및 서비스를 Apache, NGINX, Passenger, IIS와 같은 친숙한 서버에 빠르게 배포하고 규모를 조정하기 위한 서비스입니다.
  - AWS Proton은 인프라 프로비저닝, 코드 배포, 모니터링 및 업데이트를 위해 개발 팀에 필요한 모든 도구를 플랫폼 팀이 연결하고 조정할 수 있도록 지원합니다. AWS Proton은 서버리스 및 컨테이너 기반 애플리케이션의 코드형 인프라 자동 프로비저닝 및 배포를 지원합니다.
- 코드형 인프라를 활용하면 인프라 배포를 쉽게 자동화할 수 있고 인프라 불변성을 달성하는 데 도움이 됩니다. AWS는 프로그래밍, 설명 및 선언적 방식으로 인프라를 생성, 배포 및 유지 관리할 수 있는 서비스를 제공합니다.
  - AWS CloudFormation은 개발자가 질서 있고 예측 가능한 방식으로 AWS 리소스를 만들 수 있도록 도와줍니다 리소스는 JSON 또는 YAML 형식을 사용하여 텍스트 파일로 작성됩니다. 템플릿에는 생성 및 관리되는 리소스 유형에 따라 특정 구문과 구조가 필요합니다. 코드 편집기를 사용하여 JSON 또는 YAML로 리소스를 작성하고 이를 버전 관리 시스템에 체크인하면 AWS CloudFormation이 명시된 서비스를 안전하고 반복 가능한 방식으로 구축합니다.
  - AWS Serverless Application Model(AWS SAM)은 AWS에서 서버리스 애플리케이션을 구축하는 데 사용할 수 있는 오픈 소스 프레임워크입니다. AWS SAM은 다른 AWS 서비스와 통합되며 AWS CloudFormation의 확장 기능입니다.
  - AWS 클라우드 개발 키트 (AWS CDK)는 익숙한 프로그래밍 언어를 사용하여 클라우드 애플리케이션 리소스를 모델링하고 프로비저닝하기 위한 오픈 소스 소프트웨어 개발 프레임워크입니다. TypeScript, Python, Java 및 .NET을 사용하여 애플리케이션 인프라를 모델링하는 데 AWS CDK를 사용할 수 있습니다. AWS CDK는 백그라운드에서 AWS CloudFormation을 사용하여 안전하고 반복 가능한 방식으로 리소스를 프로비저닝합니다.
  - AWS Cloud Control API는 개발자들이 쉽고 일관된 방식으로 클라우드 인프라를 관리할 수 있도록 CRUDL(Create, Read, Update, Delete, List) API로 구성된 일반적인 세트를 제공합니다. 개발자는 Cloud Control API를 사용하여 AWS 및 서드파티 서비스의 수명 주기를 일관적으로 관리할 수 있습니다.
- 사용자에게 미치는 영향을 최소화하는 배포 패턴을 구현합니다.
  - 카나리 배포:
    - API Gateway Canary 릴리스 배포 설정
    - Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh
  - 블루/그린 배포: Blue/Green Deployments on AWS whitepaper에서는 블루/그린 배포 전략을 구현하기 위한 기술 예제를 설명합니다.

- 구성 또는 상태 드리프트를 감지합니다. 자세한 내용은 스택 및 리소스에 대한 비관리형 구성 변경 감지를 참조하세요.

리소스

관련 모범 사례:

- REL08-BP05 자동화를 통한 변경 사항 배포

관련 문서:

- 안전하고 간편한 배포 자동화
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- 코드형 인프라
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

관련 비디오:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

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 및 배포, 용량 추가 또는 제거, 데이터 마이그레이션 포함)에 자동화를 사용해야 합니다.

원하는 성과: 광범위한 프로덕션 전 테스트, 자동 롤백, 시차를 둔 프로덕션 배포를 통해 릴리스 프로세스에 자동화된 배포 안전을 구축합니다. 이러한 자동화를 통해 배포 실패로 인한 프로덕션 환경의 잠재적 영향을 최소화할 수 있으며, 개발자는 더 이상 프로덕션으로의 배포를 적극적으로 관찰할 필요가 없습니다.

일반적인 안티 패턴:

- 수동 변경을 수행합니다.
- 수동 비상 워크플로를 통해 자동화의 단계를 건너뜁니다.

- 일정을 앞당기기 위해 정해진 계획과 프로세스를 따르지 않습니다.
- 베이크 소요 시간을 허용하지 않고 신속한 후속 배포를 수행합니다.

이 모범 사례 확립의 이점: 자동화를 사용하여 모든 변경 사항을 배포하면 인적 오류가 발생할 가능성을 없애고 프로덕션을 변경하기 전에 테스트하는 기능을 제공합니다. 프로덕션 푸시 전에 이 프로세스를 수행하면 계획이 완전한지 확인할 수 있습니다. 또한 릴리스 프로세스로의 자동 롤백을 통해 프로덕션 문제를 식별하고 워크로드를 이전의 작동 상태로 되돌릴 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

배포 파이프라인을 자동화하세요. 배포 파이프라인을 사용하면 이상 징후의 자동 테스트 및 탐지를 간접 호출하여 프로덕션 배포 전 특정 단계에서 파이프라인을 중지하거나 변경 사항을 자동으로 롤백할 수 있습니다. 여기에 핵심은 통합 및 지속적 전달/배포(CI/CD) 문화를 채택하는 것입니다. 이 방식을 사용하면 커밋 또는 코드 변경이 구축 및 테스트 단계부터 프로덕션 환경에서의 배포에 이르기까지 다양한 자동화된 단계 게이트를 통과합니다.

일반적인 통념으로는 가장 어려운 작업 절차에 사람을 투입하는 것이 좋다고 하지만 바로 그런 이유로 가장 어려운 절차를 자동화하는 것이 좋습니다.

구현 단계

다음 단계에 따라 배포를 자동화하여 수동 작업을 제거할 수 있습니다.

- 코드를 안전하게 저장하도록 코드 리포지토리 설정: Git과 같은 인기 있는 기술을 기반으로 하는 호스팅 소스 코드 관리 시스템을 사용하여 소스 코드 및 코드형 인프라(IaC) 구성을 저장합니다.
- 소스 코드를 컴파일하고, 테스트를 실행하며, 배포 아티팩트를 생성하도록 지속적 통합 서비스 구성: 이를 위한 빌드 프로젝트를 설정하려면 Getting started with AWS CodeBuild using the console을 참조하세요.
- 오류가 발생하기 쉬운 수동 배포에 의존하지 않고 애플리케이션 배포를 자동화하고 애플리케이션 업데이트의 복잡성을 처리하는 배포 서비스 설정: AWS CodeDeploy는 Amazon EC2, AWS Fargate, AWS Lambda, 온프레미스 서버와 같은 다양한 컴퓨팅 서비스에 대한 소프트웨어 배포를 자동화합니다. 이러한 단계를 구성하려면 Getting started with CodeDeploy를 참조하세요.
- 더 빠르고 안정적인 애플리케이션 및 인프라 업데이트를 위해 릴리스 파이프라인을 자동화하는 지속적 전달 서비스 설정: 릴리스 파이프라인을 자동화하는 데 도움이 되도록 AWS CodePipeline 사용을 고려해 보세요. 자세한 내용은 CodePipeline 자습서를 참조하세요.

리소스

관련 모범 사례:

- OPS05-BP04 구축 및 배포 관리 시스템 사용
- OPS05-BP10 통합 및 배포 완전 자동화
- OPS06-BP02 테스트 배포
- OPS06-BP04 테스트 및 롤백 자동화

관련 문서:

- Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline
- APN 파트너: 자동화된 배포 솔루션의 생성을 지원할 수 있는 파트너
- AWS Marketplace: 배포 자동화에 사용할 수 있는 제품
- Automate chat messages with webhooks.
- Amazon Builders' Library: 배포 중 롤백 안전 보장
- Amazon Builders' Library: 지속적 전달을 통한 신속한 배포
- AWS CodePipeline란 무엇입니까?
- What Is CodeDeploy?
- AWS Systems Manager Patch Manager
- What is Amazon SES?
- What is Amazon Simple Notification Service?

관련 비디오:

- AWS Summit 2019: CI/CD on AWS

장애 관리

Questions

- REL 9. 데이터는 어떻게 백업하나요?
- REL 10. 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 하나요?
- REL 11. 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 하나요?

- REL 12. 신뢰성은 어떻게 테스트하나요?
- REL 13. 재해 복구(DR)를 어떻게 계획하나요?

REL 9. 데이터는 어떻게 백업하나요?

복구 시간 목표(RTO) 및 복구 지점 목표(RPO)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 서비스 및 리소스의 백업 기능을 숙지하고 사용합니다. 대부분의 서비스는 워크로드 데이터를 백업할 수 있는 기능을 제공합니다.

원하는 성과: 중요도에 따라 데이터 소스를 식별하고 분류합니다. 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

모든 AWS 데이터 스토어는 백업 기능을 제공합니다. Amazon RDS 및 Amazon DynamoDB와 같은 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 Backup은 AWS 서비스 전체에서 데이터 보호를 중앙 집중화하고 자동화하는 기능을 제공하는 도구입니다. 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 DynamoDB와 같은 AWS 서비스에는 기본적으로 백업을 생성하는 기능이 내장되어 있습니다. 서드파티 백업 소프트웨어를 사용할 수도 있습니다.

온프레미스 데이터는 AWS Storage Gateway 또는 AWS DataSync를 사용하여 AWS 클라우드에 백업할 수 있습니다. Amazon S3 버킷은 AWS에 이 데이터를 저장하는 데 사용할 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 항목이 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 백업하지 않아도 됩니다.

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 식별합니다. 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. 데이터세트마다 워크로드의 중요도가 다르므로 복원력에 대한 요구 사항도 달라집니다. 예를 들어, 어떤 데이터는 매우 중요하며 0에 가까운 RPO가 필요하

지만 어떤 데이터는 덜 중요하며 더 높은 RPO와 일부 데이터 손실을 용인할 수 있습니다. 마찬가지로, 데이터세트마다 RTO 요구 사항이 다릅니다.

3. AWS 또는 서드파티 서비스를 사용하여 데이터 백업을 생성합니다. AWS Backup은 AWS에서 다양한 데이터 소스의 백업을 생성할 수 있는 관리형 서비스입니다. AWS Elastic Disaster Recovery에서는 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 자세한 내용은 아래 나열된 리소스를 참조하세요.
4. 백업되지 않은 데이터에 대해서는 데이터 재현 메커니즘을 설정합니다. 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 들어 Amazon S3의 데이터를 데이터 웨어하우스(예: Amazon Redshift) 또는 MapReduce 클러스터(예: Amazon EMR)로 로드하여 해당 데이터를 분석하는 경우 이 사례는 다른 소스에서 복제할 수 있는 데이터 예제에 해당할 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?
- What is AWS DataSync?
- What is Volume Gateway?

- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Amazon EBS 스냅샷
- Backing Up Amazon EFS
- Backing up Amazon FSx for Windows File Server
- ElastiCache for Redis 백업 및 복원
- Creating a DB Cluster Snapshot in Neptune
- DB 스냅샷 생성
- Creating an EventBridge Rule That Triggers on a Schedule
- Amazon S3에서 크로스 리전 복제
- EFS-to-EFS AWS Backup
- Exporting Log Data to Amazon S3
- 객체 수명 주기 관리
- DynamoDB에 대한 온디맨드 백업 및 복원
- DynamoDB의 특정 시점으로 복구
- Working with Amazon OpenSearch Service Index Snapshots
- What is AWS Elastic Disaster Recovery?

관련 비디오:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

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 설정합니다.
- 백업을 암호화하지 않습니다.

이 모범 사례 확립의 이점: 백업의 보안을 유지하면 데이터 변조가 방지되며, 데이터 암호화는 데이터가 실수로 노출될 경우 해당 데이터에 대한 액세스를 방지합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 저장 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. 두 방법 모두 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 스토어에서 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.
   - 필요한 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.
   - 소스 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 S3: 암호화로 데이터 보호
- CRR 추가 구성: AWS KMS에 저장된 암호화 키를 사용하는 서버 측 암호화(SSE)로 생성된 객체 복제
- DynamoDB 저장 데이터 암호화
- Amazon RDS 리소스 암호화
- Encrypting Data and Metadata in Amazon EFS
- Encryption for Backups in AWS
- 암호화된 테이블 관리
- 보안 원칙 - AWS Well-Architected Framework
- What is AWS Elastic Disaster Recovery?

REL09-BP03 자동으로 데이터 백업 수행

Recovery Point Objective(RPO)에 정의된 정기적인 일정 또는 데이터세트의 변경에 따라 백업이 자동으로 생성되도록 구성합니다. 적은 데이터 손실을 요구하는 중요한 데이터세트는 자주 자동으로 백업되어야 합니다. 반면 일부 손실은 용인하는 중요도가 상대적으로 낮은 데이터의 경우 더 낮은 빈도로 백업할 수 있습니다.

원하는 성과: 정해진 주기로 데이터 소스의 백업을 생성하는 자동화된 프로세스.

일반적인 안티 패턴:

- 수동으로 백업을 수행합니다.
- 백업을 지원하는 리소스를 사용하지만 자동화 대상에 백업을 포함하지 않습니다.

이 모범 사례 확립의 이점: 백업을 자동화하면 RPO를 기준으로 주기적으로 백업이 생성되며, 생성되지 않은 경우 알림을 보냅니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

AWS Backup은 다양한 AWS 데이터 소스의 자동 데이터 백업을 생성하는 데 사용할 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 지속적 백업을 지원합니다. Amazon S3 버전 관리를 활성화한 경우 자동으로 수행됩니다. Amazon Data Lifecycle Manager를 사용하여 Amazon EBS 스냅샷의 생성, 복사 및 삭제를 자동화할 수 있습니다. 또한 Amazon EBS 지원 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 백업되고 있는 데이터 소스를 식별합니다. 자세한 내용은 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제 섹션을 참조하세요.
2. 워크로드의 RPO를 결정합니다. 자세한 내용은 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의 섹션을 참조하세요.

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 Testing Backup and Restore of Data를 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. 온프레미스 데이터 소스 또는 메시지 대기열과 같이 자동화된 백업 솔루션 또는 관리형 서비스에서 지원되지 않는 데이터 소스의 경우 신뢰할 수 있는 서드파티 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 생성에 포함된 로직을 정의하고, Amazon EventBridge를 사용하여 RPO를 기반으로 정해진 빈도에 간접 호출할 수 있습니다.

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
- AWS Backup란 무엇입니까?
- AWS Step Functions란 무엇입니까?
- What is AWS Elastic Disaster Recovery?

관련 비디오:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

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 성과: 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 누락되거나 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 시점 복구(PITR)를 지원합니다. 연속 백업을 사용하면 데이터세트를 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 Recovery에서는 Amazon EBS 볼륨의 지속적인 시점 복구 스냅샷을 제공합니다. 소스 서버가 복제되면 구성된 정책을 기반으로 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 백업되고 있는 데이터 소스와 이러한 백업이 저장되는 위치를 식별합니다. 구현 지침은 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제 섹션을 참조하세요.
2. 각 데이터 소스에 대해 데이터 검증 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 데이터를 복원하기 위한 RTO 및 RPO를 설정합니다. 구현 지침은 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의 섹션을 참조하세요.
4. 복구 역량을 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. 프로덕션에서 데이터 복원을 위해 현재 확립된 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 리소스에 간단하게 복원할 수 있습니다. AWS Elastic Disaster Recovery를 사용한 경우 복구 드릴을 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 있나요? 이 데이터가 워크로드의 RPO에 부합하나요?
7. 복원 및 복구에 필요한 시간을 측정하고 이를 설정된 RTO와 비교합니다. 프로세스가 워크로드의 RTO에 부합하나요? 예를 들어, 복원 프로세스가 시작됐을 때의 타임스탬프와 복구 검증이 완료됐을 때의 타임스탬프를 비교하여 프로세스에 걸린 시간을 계산합니다. 모든 AWS API 직접 호출에는 타임스탬프가 지정되며 이 정보는 AWS CloudTrail에서 확인할 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 사

용하여 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 타임스탬프 정보가 메타데이터의 일부로 포함됩니다.

8. 데이터 검증이 실패하거나 복원 및 복구에 필요한 시간이 워크로드에 확립된 RTO를 초과하는 경우 이해관계자에게 알립니다. 이 실습과 같이 자동화를 구현하는 경우 Amazon Simple Notification Service(SNS)와 같은 서비스를 사용하여 이해관계자에게 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. 이 프로세스를 주기적으로 실행하도록 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 간접 호출할 수 있습니다. AWS Backup에서 데이터 복구 유효성 검사 자동화 방법에 대해 알아보세요. 또한 이 Well-Architected 실습은 여기에 있는 여러 단계에 대해 자동화를 수행하는 한 가지 방법에 대한 실습 경험을 제공합니다.

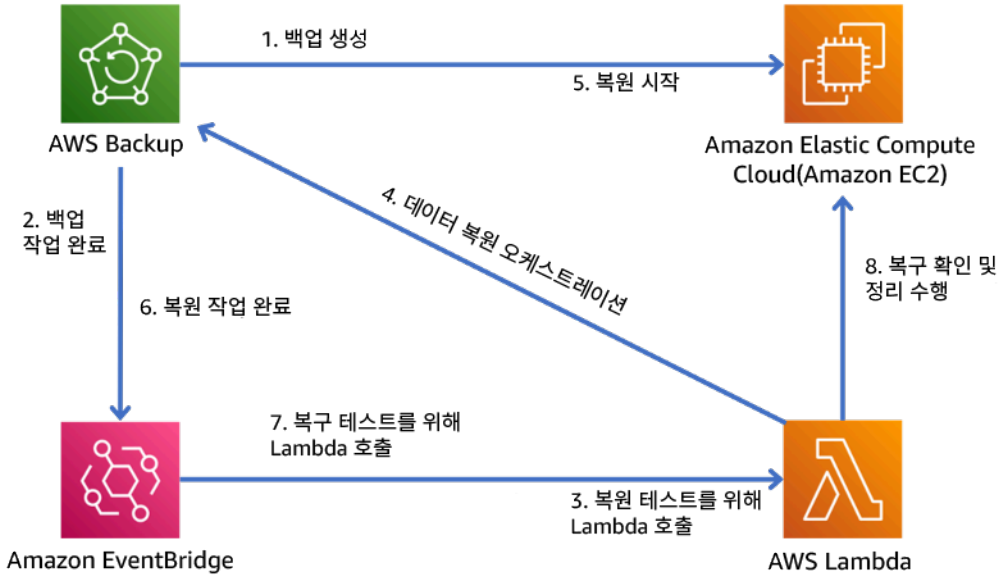


그림 9. 자동화된 백업 및 복원 프로세스

구현 계획의 작업 수준: 보통~높음(유효성 검사 기준의 복잡성에 따라 달라짐).

리소스

관련 문서:

- Automate data recovery validation with AWS Backup
- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Creating an EventBridge Rule That Triggers on a Schedule
- DynamoDB에 대한 온디맨드 백업 및 복원
- AWS Backup란 무엇입니까?
- AWS Step Functions란 무엇입니까?
- AWS Elastic Disaster Recovery이란 무엇입니까?
- AWS Elastic Disaster Recovery

REL 10. 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 하나요?

장애 격리는 구성 요소 또는 시스템 장애의 영향을 정의된 경계로 제한합니다. 올바르게 격리하면 경계 외부의 구성 요소는 장애의 영향을 받지 않습니다. 여러 장애 격리 경계에서 워크로드를 실행하면 장애에 대한 복원력이 향상될 수 있습니다.

모범 사례

- REL10-BP01 워크로드를 여러 위치에 배포
- REL10-BP02 단일 위치로 제약된 구성 요소의 복구 자동화
- REL10-BP03 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.

AWS에서 서비스 설계의 기본 원칙은 기본 물리적 인프라를 포함하여 단일 장애 지점을 방지하는 것입니다. AWS는 리전이라는 여러 지리적 위치에서 전 세계적으로 클라우드 컴퓨팅 리소스 및 서비스를 제공합니다. 각 리전은 물리적 및 논리적으로 독립적이며 3개 이상의 가용 영역(AZ)으로 구성됩니다. 가용 영역은 지리적으로 서로 가깝지만 물리적으로 분리되고 격리되어 있습니다. 가용 영역과 리전 간에 워크로드를 분산하면 화재, 홍수, 날씨 관련 재해, 지진, 인적 오류와 같은 위협의 위험을 완화할 수 있습니다.

워크로드에 적합한 고가용성을 제공하는 위치 전략을 수립하세요.

원하는 성과: 프로덕션 워크로드는 내결함성과 고가용성을 달성하기 위해 여러 가용 영역(AZ) 또는 리전에 분산됩니다.

일반적인 안티 패턴:

- 프로덕션 워크로드는 단일 가용 영역에만 존재합니다.
- 다중 AZ 아키텍처로 비즈니스 요구 사항을 충족할 수 있는 경우 다중 리전 아키텍처를 구현합니다.
- 배포 또는 데이터가 비동기화되어 구성이 드리프트되거나 복제가 저조해지지 않습니다.
- 복원력과 다중 위치 요구 사항이 구성 요소 간에 다를 경우 애플리케이션 구성 요소 간의 종속성을 고려하지 않습니다.

이 모범 사례 확립의 이점:

- 워크로드는 전력 또는 환경 제어 장애, 자연 재해, 업스트림 서비스 장애 또는 AZ나 전체 리전에 영향을 미치는 네트워크 문제와 같은 인시던트에 대해 복원력이 더 높습니다.
- 더 광범위한 Amazon EC2 인스턴스 인벤토리에 액세스하고 특정 EC2 인스턴스 유형을 시작할 때 InsufficientCapacityExceptions(ICE)의 가능성을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

한 리전에서 최소 2개 이상의 가용 영역(AZ)에 모든 프로덕션 워크로드를 배포하고 운영합니다.

다중 가용 영역 사용

가용 영역은 리소스를 호스팅하는 위치로, 화재, 홍수 및 토네이도와 같은 위험으로 인한 상관관계가 있는 장애를 방지하기 위해 물리적으로 서로 분리되어 있습니다. 각 가용 영역에는 유틸리티 전원 연결, 백업 전원, 기계 서비스 및 네트워크 연결을 포함한 독립적인 물리적 인프라가 있습니다. 이로 인해 이러한 구성 요소에 장애가 발생하면 영향을 받는 가용 영역으로만 장애가 제한됩니다. 예를 들어 AZ 전체에 발생한 인시던트로 인해 영향을 받는 가용 영역에서 EC2 인스턴스를 사용할 수 없는 경우 다른 가용 영역의 인스턴스는 여전히 사용할 수 있습니다.

물리적으로 분리되어 있음에도 불구하고 동일한 AWS 리전의 가용 영역은 처리량이 높고 지연 시간이 짧은(10밀리초 미만) 네트워킹을 제공할 수 있을 만큼 충분히 가깝습니다. 사용자 경험에 큰 영향을 주지 않고 대부분의 워크로드에 대해 가용 영역 간에 데이터를 동기식으로 복제할 수 있습니다. 즉, 액티브/액티브 또는 액티브/대기 구성에서 가용 영역을 사용할 수 있습니다.

워크로드와 연결된 모든 컴퓨팅은 여러 가용 영역에 분산되어야 합니다. 여기에는 Amazon EC2 인스턴스, AWS Fargate 작업 및 VPC 연결 AWS Lambda 함수가 포함됩니다. EC2 Auto Scaling, Amazon

Elastic Container Service(Amazon ECS) 및 Amazon Elastic Kubernetes Service(Amazon EKS)를 포함한 AWS 컴퓨팅 서비스는 가용 영역에서 컴퓨팅을 시작하고 관리할 수 있는 방법을 제공합니다. 가용성을 유지하기 위해 필요에 따라 다른 가용 영역에서 컴퓨팅을 자동으로 대체하도록 구성합니다. 트래픽을 사용 가능한 가용 영역으로 전달하려면 컴퓨팅 앞에 Application Load Balancer 또는 Network Load Balancer와 같은 로드 밸런서를 배치합니다. AWS 로드 밸런서는 가용 영역 장애가 발생할 경우 사용 가능한 인스턴스로 트래픽을 다시 라우팅할 수 있습니다.

또한 워크로드에 대한 데이터를 복제하여 여러 가용 영역에서 사용할 수 있도록 해야 합니다. Amazon S3, Amazon Elastic File Service(Amazon EFS), Amazon Aurora, Amazon DynamoDB, Amazon Simple Queue Service(Amazon SQS) 및 Amazon Kinesis Data Streams와 같은 일부 AWS 관리형 데이터 서비스는 기본적으로 여러 가용 영역에 데이터를 복제하며 가용 영역 장애에 대해 견고합니다. Amazon Relational Database Service(Amazon RDS), Amazon Redshift 및 Amazon ElastiCache 등 다른 AWS 관리형 데이터 서비스를 사용하려면 다중 AZ 복제를 활성화해야 합니다. 활성화하면 이러한 서비스는 자동으로 가용 영역 장애를 감지하고, 요청을 사용할 수 있는 가용 영역으로 리디렉션하고, 고객 개입 없이 복구 후 필요에 따라 데이터를 다시 복제합니다. 다중 AZ 기능, 동작 및 작업을 이해하기 위해 사용하는 각 AWS 관리형 데이터 서비스에 대한 사용 설명서를 숙지합니다.

Amazon Elastic Block Store(Amazon EBS) 볼륨 또는 Amazon EC2 인스턴스 스토리지와 같은 자체 관리형 스토리지를 사용하는 경우 다중 AZ 복제를 직접 관리해야 합니다.

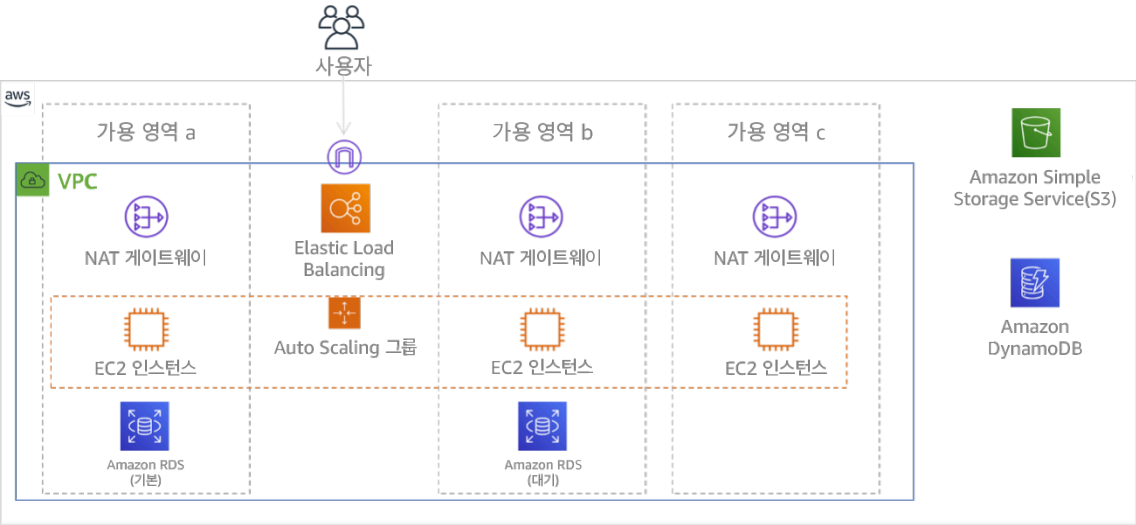


그림 9: 3개의 가용 영역에 배포된 다중 계층 아키텍처. Amazon S3 및 Amazon DynamoDB는 항상 자동으로 다중 AZ에 유지됩니다. 또한 ELB는 3개 영역 모두에 배포됩니다.

다중 AWS 리전 사용

극도의 복원력이 필요한 워크로드(예: 중요한 인프라, 건강 관련 애플리케이션 또는 엄격한 고객 요구 사항 또는 필수 가용성 요구 사항이 있는 서비스)가 있는 경우 단일 AWS 리전이 제공할 수 있는 것 이

상의 추가 가용성이 필요할 수 있습니다. 이 경우 최소 2개의 AWS 리전에 워크로드를 배포하고 운영해야 합니다(데이터 레지던시 요구 사항에서 허용한다고 가정).

AWS 리전은 전 세계의 여러 지리적 지역과 여러 대륙에 위치해 있습니다. AWS 리전은 가용 영역만 있는 것보다 물리적 분리 및 격리의 정도가 훨씬 더 큽니다. AWS 서비스는 거의 예외 없이 이 설계를 활용하여 서로 다른 리전 간에 완전히 독립적으로 운영됩니다(리전 서비스라고도 함). 한 AWS 리전 서비스의 장애는 다른 리전의 서비스에 영향을 주지 않도록 설계되었습니다.

여러 리전에서 워크로드를 운영할 때는 추가 요구 사항을 고려해야 합니다. 서로 다른 리전의 리소스는 서로 별개이고 독립적이므로 각 리전에서 워크로드의 구성 요소를 복제해야 합니다. 여기에는 컴퓨팅 및 데이터 서비스 외에도 VPC와 같은 기본 인프라가 포함됩니다.

참고: 다중 리전 설계를 고려할 때는 워크로드가 단일 리전에서 실행될 수 있는지 확인합니다. 한 리전의 구성 요소가 다른 리전의 서비스 또는 구성 요소에 의존하는 리전 간 종속성을 생성하는 경우 장애 위험을 높이고 신뢰성 태세를 크게 약화시킬 수 있습니다.

다중 리전 배포를 용이하게 하고 일관성을 유지 관리하기 위해 AWS CloudFormation StackSets는 여러 리전에 전체 AWS 인프라를 복제할 수 있습니다. 또한 AWS CloudFormation은 구성 드리프트를 감지하고 리전의 AWS 리소스가 동기화되지 않을 때 알려줍니다. 많은 AWS 서비스가 중요한 워크로드 자산에 대해 다중 리전 복제를 제공합니다. 예를 들어 EC2 Image Builder는 사용하는 각 리전에 구축할 때마다 EC2 머신 이미지(AMI)를 게시할 수 있습니다. Amazon Elastic Container Registry(Amazon ECR)는 컨테이너 이미지를 선택한 리전에 복제할 수 있습니다.

또한 선택한 각 리전에 데이터를 복제해야 합니다. Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Elasticache 및 Amazon EFS를 비롯한 많은 AWS 관리형 데이터 서비스는 리전 간 복제 기능을 제공합니다. Amazon DynamoDB 글로벌 테이블은 지원되는 모든 리전의 쓰기를 수락하고 구성된 다른 모든 리전에 데이터를 복제합니다. 다른 리전에는 읽기 전용 복제본이 포함되어 있으므로 다른 서비스에서는 쓰기를 위한 기본 리전을 지정해야 합니다. 워크로드가 사용하는 각 AWS 관리형 데이터 서비스에 관해 사용 설명서 및 개발자 안내서를 참조하여 다중 리전 기능과 제한 사항을 이해하세요. 쓰기를 지시해야 하는 위치, 트랜잭션 기능 및 제한 사항, 복제 수행 방식, 리전 간 동기화를 모니터링하는 방법에 특히 주의를 기울이세요.

AWS는 요청 트래픽을 뛰어난 유연성으로 리전 배포로 라우팅할 수 있는 기능도 제공합니다. 예를 들어, Amazon Route 53를 사용하여 트래픽을 사용자에게 가장 가까운 가용 리전으로 전달하도록 DNS 레코드를 구성할 수 있습니다. 또는 한 리전을 기본 리전으로 지정하고 기본 리전이 비정상이 되는 경우에만 리전 복제본으로 돌아가는 액티브/대기 구성으로 DNS 레코드를 구성할 수 있습니다. 비정상 엔드포인트를 감지하고 자동 장애 조치를 수행하고 Amazon Application Recovery Controller(ARC)를 추가로 사용하여 필요에 따라 수동으로 다시 트래픽 라우팅을 할 수 있는 고가용성 라우팅 제어를 제공하도록 Route 53 상태 확인을 구성할 수 있습니다.

고가용성을 위해 여러 리전에서 운영하지 않기로 선택하더라도 재해 복구(DR) 전략의 일환으로 여러 리전을 고려해 보세요. 가능하면 워크로드의 인프라 구성 요소와 데이터를 보조 리전의 웜 대기 또는 파일럿 라이트 구성으로 복제합니다. 이 설계에서는 기본 리전에서 VPC, Auto Scaling 그룹, 컨테이너 오케스트레이터 및 기타 구성 요소와 같은 기준 인프라를 복제하지만 대기 리전의 가변 크기 구성 요소(예: EC2 인스턴스 및 데이터베이스 복제본 수)는 작동 가능한 최소 크기로 구성합니다. 또한 기본 리전에서 대기 리전으로의 지속적인 데이터 복제를 준비합니다. 인시던트가 발생하면 대기 리전의 리소스를 스케일 아웃, 즉 증가시킨 다음 기본 리전으로 승격할 수 있습니다.

구현 단계

1. 비즈니스 이해관계자 및 데이터 레지던시 전문가와 협력하여 리소스 및 데이터를 호스팅하는 데 사용할 수 있는 AWS 리전을 결정합니다.
2. 비즈니스 및 기술 이해관계자와 협력하여 워크로드를 평가하고 다중 AZ 접근 방식(단일 AWS 리전)으로 복원력 요구 사항을 충족할 수 있는지, 아니면 다중 리전 접근 방식(여러 리전이 허용되는 경우)이 필요한지 확인합니다. 여러 리전을 사용하면 가용성이 향상될 수 있지만 복잡성이 늘어나고 비용이 추가로 발생할 수 있습니다. 평가에서 다음 요소를 고려하세요.
   a. 비즈니스 목표 및 고객 요구 사항: 가용 영역 또는 리전에서 워크로드에 영향을 미치는 인시던트가 발생할 경우 가동 중지 시간이 얼마나 허용되나요? REL13-BP01 가동 중지 시간 및 데이터 손실에 대한 복구 목표 정의에 설명된 대로 복구 시점 목표를 평가합니다.
   b. 재해 복구(DR) 요구 사항: 어떤 종류의 잠재적 재해에 대비하고 싶은가요? 단일 가용 영역에서 전체 리전에 이르기까지 다양한 영향 범위에서 데이터 손실 또는 장기적인 사용 불가의 가능성을 고려합니다. 여러 가용 영역에 데이터와 리소스를 복제하고 단일 가용 영역에 지속적인 장애가 발생하는 경우 다른 가용 영역에서 서비스를 복구할 수 있습니다. 리전 간에 데이터 및 리소스를 복제하는 경우 다른 리전에서 서비스를 복구할 수 있습니다.
3. 컴퓨팅 리소스를 여러 가용 영역에 배포합니다.
   a. VPC에서 서로 다른 가용 영역에 여러 서브넷을 생성합니다. 인시던트 발생 중에도 워크로드를 처리하는 데 필요한 리소스를 수용할 수 있도록 각 서브넷을 충분히 크게 구성합니다. 자세한 내용은 REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인을 참조하세요.
   b. Amazon EC2 인스턴스를 사용하는 경우 EC2 Auto Scaling을 사용하여 인스턴스를 관리합니다. Auto Scaling 그룹을 생성할 때 이전 단계에서 선택한 서브넷을 지정합니다.
   c. Amazon ECS 또는 Amazon EKS용 AWS Fargate 컴퓨팅을 사용하는 경우 ECS 서비스를 생성하거나 ECS 작업을 시작하거나 EKS용 Fargate 프로필을 생성할 때 첫 번째 단계에서 선택한 서브넷을 선택합니다.
   d. VPC에서 실행해야 하는 AWS Lambda 함수를 사용하는 경우 Lambda 함수를 생성할 때 첫 번째 단계에서 선택한 서브넷을 선택합니다. VPC 구성이 없는 함수의 경우 AWS Lambda는 자동으로 가용성을 관리합니다.

   e. 로드 밸런서와 같은 트래픽 디렉터를 컴퓨팅 리소스 앞에 배치합니다. 교차 영역 로드 밸런싱이 활성화된 경우 AWS Application Load Balancer 및 Network Load Balancer는 가용 영역 장애로 인해 EC2 인스턴스 및 컨테이너와 같은 대상에 연결할 수 없을 때 이를 감지하고 정상 가용 영역의 대상으로 트래픽을 다시 라우팅합니다. 교차 영역 로드 밸런싱을 비활성화하는 경우 Amazon Application Recovery Controller(ARC)를 사용하여 영역 전환 기능을 제공합니다. 서드파티 로드 밸런서를 사용하거나 자체 로드 밸런서를 구현한 경우 여러 가용 영역에서 여러 프런트엔드로 구성합니다.

4. 여러 가용 영역에서 워크로드의 데이터를 복제합니다.

   a. Amazon RDS, Amazon ElastiCache 또는 Amazon FSx와 같은 AWS 관리형 데이터 서비스를 사용하는 경우 사용 설명서를 살펴보고 데이터 복제 및 복원력 기능을 이해합니다. 필요한 경우 교차 AZ 복제 및 장애 조치를 활성화합니다.

   b. Amazon S3, Amazon EFS 및 Amazon FSx와 같은 AWS 관리형 스토리지 서비스를 사용하는 경우 높은 내구성을 필요로 하는 데이터에 단일 AZ 또는 One Zone 구성을 사용하지 마세요. 이러한 서비스에는 다중 AZ 구성을 사용합니다. 각 서비스의 사용 설명서를 확인하여 다중 AZ 복제가 기본적으로 활성화되어 있는지, 아니면 직접 활성화해야 하는지 확인합니다.

   c. 자체 관리형 데이터베이스, 대기열 또는 기타 스토리지 서비스를 실행하는 경우 애플리케이션의 지침 또는 모범 사례에 따라 다중 AZ 복제를 준비합니다. 애플리케이션의 장애 조치 절차를 숙지합니다.

5. AZ 장애를 감지하고 트래픽을 정상 가용 영역으로 다시 라우팅하도록 DNS 서비스를 구성합니다. Amazon Route 53를 Elastic Load Balancer와 함께 사용하면 이 작업을 자동으로 수행할 수 있습니다. Route 53는 상태 확인을 사용하여 쿼리에 정상 IP 주소로만 응답하는 장애 조치 레코드로 구성할 수도 있습니다. 장애 조치에 사용되는 DNS 레코드의 경우 레코드 캐싱이 복구를 방해하지 않도록 짧은 Time to Live(TTL) 값(예: 60초 이하)을 지정합니다(Route 53 별칭 레코드가 적합한 TTL을 제공함).

여러 AWS 리전을 사용할 때의 추가 단계

1. 워크로드에서 사용하는 모든 운영 체제(OS) 및 애플리케이션 코드를 선택한 리전에 복제합니다. 필요한 경우 Amazon EC2 Image Builder와 같은 솔루션을 사용하여 Amazon EC2 Machine Images(AMI)를 복제합니다. Amazon ECR 리전 간 복제와 같은 솔루션을 사용하여 레지스트리에 저장된 컨테이너 이미지를 복제합니다. 애플리케이션 리소스를 저장하는 데 사용되는 모든 Amazon S3 버킷에 대해 리전 복제를 활성화합니다.

2. 컴퓨팅 리소스 및 구성 메타데이터(예: AWS Systems Manager Parameter Store에 저장된 파라미터)를 여러 리전에 배포합니다. 이전 단계에서 설명한 것과 동일한 절차를 사용하되, 워크로드에 사용 중인 각 리전의 구성을 복제합니다. AWS CloudFormation과 같은 코드형 인프라 솔루션을 사용

하여 리전 간에 구성을 동일하게 복제합니다. 재해 복구를 위해 파일럿 라이트 구성에서 보조 리전을 사용하는 경우 컴퓨팅 리소스 수를 최솟값으로 줄여 비용을 절감할 수 있습니다. 이로 인해 복구 시간이 늘어날 수 있습니다.

3. 기본 리전의 데이터를 보조 리전으로 복제합니다.
   a. Amazon DynamoDB 글로벌 테이블은 지원되는 모든 리전에서 작성할 수 있는 데이터의 글로벌 복제본을 제공합니다. Amazon RDS, Amazon Aurora, Amazon Elasticache와 같은 다른 AWS 관리형 데이터 서비스를 사용하면 기본(읽기/쓰기) 리전과 복제본(읽기 전용) 리전을 지정합니다. 리전 복제에 대한 자세한 내용은 각 서비스의 사용자 및 개발자 안내서를 참조하세요.
   b. 자체 관리형 데이터베이스를 실행하는 경우 애플리케이션의 지침 또는 모범 사례에 따라 다중 리전 복제를 준비합니다. 애플리케이션의 장애 조치 절차를 숙지합니다.
   c. 워크로드가 AWS EventBridge를 사용하는 경우 선택한 이벤트를 기본 리전에서 보조 리전으로 전달해야 할 수 있습니다. 이렇게 하려면 보조 리전의 이벤트 버스를 기본 리전의 일치하는 이벤트의 대상으로 지정합니다.
4. 리전 간에 동일한 암호화 키를 사용할지, 어느 정도의 범위로 사용할지를 고려합니다. 보안과 사용 편의성의 균형을 맞추는 일반적인 접근 방식은 리전-로컬 데이터 및 인증에 리전 범위 키를 사용하고, 다른 리전 간에 복제되는 데이터의 암호화에는 전역 범위 키를 사용하는 것입니다. AWS Key Management Service(KMS)는 다중 리전 키를 지원하여 리전 간에 공유되는 키를 안전하게 배포하고 보호합니다.
5. 트래픽을 정상 엔드포인트가 포함된 리전으로 전달하여 애플리케이션의 가용성을 개선하려면 AWS Global Accelerator를 고려하세요.

리소스

관련 모범 사례:

- REL02-BP03 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인
- REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지
- REL13-BP01 가동 중지 시간 및 데이터 손실에 대한 복구 목표 정의

관련 문서:

- AWS 글로벌 인프라
- 백서: AWS Fault Isolation Boundaries
- Resilience in Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: Example: Distribute instances across Availability Zones

- How EC2 Image Builder works
- How Amazon ECS places tasks on container instances (includes Fargate)
- AWS Lambda의 복원성
- Amazon S3: Replicating objects overview
- Private image replication in Amazon ECR
- 글로벌 테이블: DynamoDB를 사용한 다중 리전 복제
- Amazon Elasticache for Redis OSS: Replication across AWS 리전 using global datastores
- Resilience in Amazon RDS
- Amazon Aurora Global Database 사용
- AWS Global Accelerator 개발자 안내서
- Multi-Region keys in AWS KMS
- Amazon Route 53: Configuring DNS failover
- Amazon Application Recovery Controller (ARC) Developer Guide
- Sending and receiving Amazon EventBridge events between AWS 리전
- Creating a Multi-Region Application with AWS Services blog series
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby

관련 비디오:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure

REL10-BP02 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 배포하며, 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. 여러 노드를 프로비저닝할 수 있습니다. EMR 파일 시스템(EMRFS)을 사용하면 EMR의 데이터를 Amazon S3에 저장할 수 있으며, 이는 다시 여러 가용 영역 또는 AWS 리전에 걸쳐 복제할 수 있습니다.

마찬가지로 Amazon Redshift의 경우 기본적으로 사용자가 선택한 AWS 리전 내에서 임의로 선택된 가용 영역에 클러스터를 프로비저닝합니다. 클러스터 노드는 모두 동일한 영역에 프로비저닝됩니다.

온프레미스 데이터 센터에 배포된 상태 저장 서버 기반 워크로드의 경우 AWS Elastic Disaster Recovery를 사용하여 AWS에서 워크로드를 보호할 수 있습니다. AWS에 이미 호스팅된 경우 Elastic Disaster Recovery를 사용하여 워크로드를 대체 가용 영역 또는 리전으로 보호할 수 있습니다. Elastic Disaster Recovery는 가벼운 스테이징 영역으로의 지속적 블록 수준 복제를 사용하여 온프레미스 및 클라우드 기반 애플리케이션에 대한 빠르고 신뢰할 수 있는 복구를 지원합니다.

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 규모 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
- 인스턴스를 복구합니다.
- 서비스 자동 규모 조정
- What Is Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP03 격벽 아키텍처를 사용하여 영향 범위 제한

격벽 아키텍처(셀 기반 아키텍처라고도 함)를 구현하여 워크로드 내 장애의 영향을 제한된 수의 구성 요소로 제한합니다.

원하는 성과: 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 세트를 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 간접 호출하는 요청과 같이 억제하기 어려운 실패 유형에 대한 복원력을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 장애의 영향을 받지 않습니다. 장애 격리 경계를 여러 개 사용하면 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 직접 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.

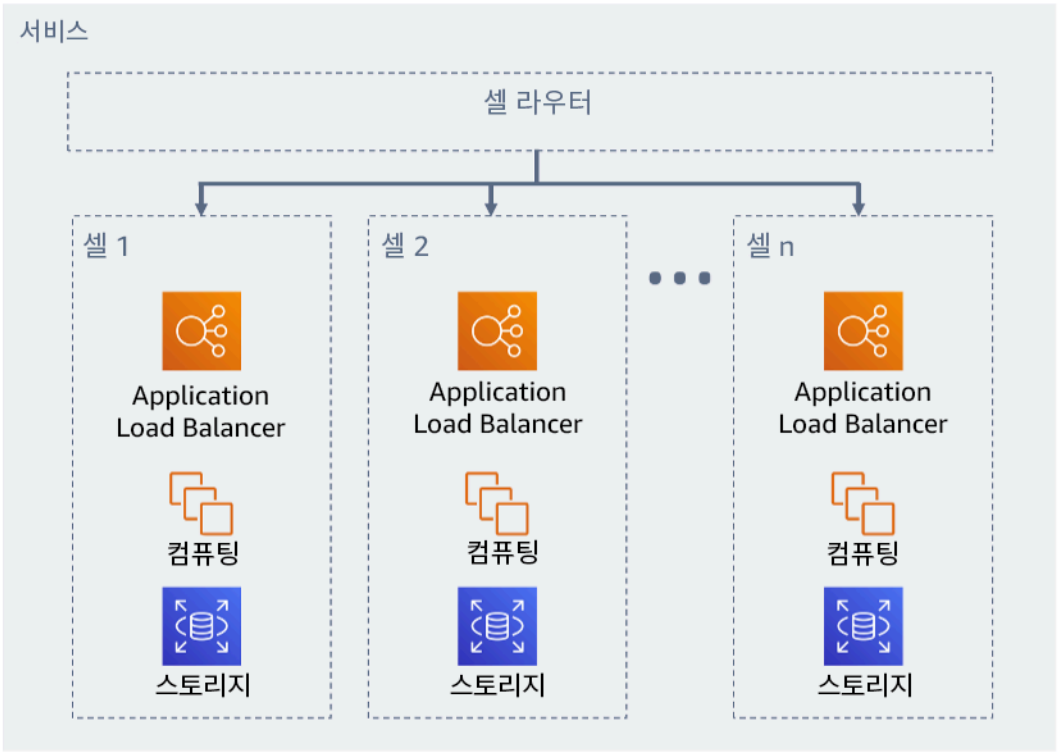


그림 11:; 셀 기반 아키텍처

구현 단계

셀 기반 아키텍처를 설계할 때 고려해야 할 몇 가지 설계 고려 사항이 있습니다.

1. 파티션 키: 파티션 키를 선택할 때는 특별한 고려 사항에 주의해야 합니다.

- 이 키는 서비스의 세부 수준 또는 최소한의 크로스 셀 상호 작용으로 서비스의 워크로드를 세분화할 수 있는 자연스러운 방식에 맞춰 조정되어야 합니다. `customer ID` 또는 `resource ID`를 예로 들 수 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 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 구획 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 MacCárthaigh의 Reliability, constant work, and a good cup of coffee에서 설명한 대로, 단순한 설계와 일정한 작업 패턴은 신뢰할 수 있는 시스템을 만들고 취약성을 줄여줍니다.
4. 셀 크기: 셀은 최대 크기를 가져야 하며 이 수치를 초과하여 성장하지 않도록 해야 합니다.
   - 중단점에 도달하고 안전한 작동 마진이 설정될 때까지 철저한 테스트를 수행하여 최대 크기를 식별해야 합니다. 테스트 사례를 구현하는 방법에 대한 자세한 내용은 REL07-BP04 워크로드 로드 테스트 섹션을 참조하세요.
   - 전체 워크로드는 셀 추가를 통해 증가하므로, 수요 증가에 따라 워크로드를 확장할 수 있습니다.
5. 다중 AZ 또는 다중 리전 전략: 다양한 장애 도메인으로부터 보호하려면 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 예방합니다. 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-BP01 워크로드를 여러 위치에 배포 섹션을 참조하세요.

6. 코드 배포: 코드 변경 사항을 모든 셀에 동시에 배포하는 것보다 시차를 두고 코드를 배포하는 전략을 사용하는 것이 좋습니다.
   - 이렇게 하면 잘못된 배포 또는 사람의 실수로 인해 여러 셀에 미치는 잠재적인 장애를 최소화하는 데 도움이 됩니다. 자세한 내용은 안전하고 간편한 배포 자동화를 참조하세요.

리소스

관련 모범 사례:

- REL07-BP04 워크로드 로드 테스트
- REL10-BP01 워크로드를 여러 위치에 배포

관련 문서:

- Reliability, constant work, and a good cup of coffee
- AWS and Compartmentalization
- 셔플 샤딩을 사용한 워크로드 격리
- 안전하고 간편한 배포 자동화

관련 비디오:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

REL 11. 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 하나요?

고가용성 및 낮은 평균 복구 시간(MTTR)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 플레인 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 장애 또는 성능 저하가 발생하는 즉시 사용자 및 자동화된 시스템이 이를 인식할 수 있도록 합니다. 비즈니스 가치를 기반으로 핵심 성과 지표(KPI)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.

원하는 성과: 워크로드의 필수 구성 요소를 독립적으로 모니터링하여 언제 어디서 장애가 발생하는지 감지하고 이에 대해 경고합니다.

일반적인 안티 패턴:

- 경보가 구성되지 않았기 때문에 알림 없이 중단이 발생합니다.
- 경보가 존재하지만 대응 시간이 충분하지 않은 임계치에 있습니다.
- 지표는 Recovery Time Objective(RTO)를 충족하기에 충분한 지표가 수집되지 않는 경우가 많습니다.
- 워크로드의 고객 대상 인터페이스만 능동적으로 모니터링됩니다.
- 기술 지표만 수집하며 비즈니스 기능 지표는 수집하지 않습니다.
- 워크로드의 사용자 경험을 측정하는 지표가 없습니다.
- 너무 많은 모니터가 생성되었습니다.

이 모범 사례 확립의 이점: 모든 계층에서 적절한 모니터링을 사용하면 감지 시간을 단축하여 복구 시간을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

모니터링을 위해 검토할 모든 워크로드를 식별합니다. 모니터링해야 할 워크로드의 모든 구성 요소를 식별했으면 이제 모니터링 간격을 결정해야 합니다. 모니터링 간격은 장애 감지에 걸리는 시간을 기준으로 복구를 시작할 수 있는 속도에 직접적인 영향을 미칩니다. 평균 감지 시간(MTTD)은 장애가 발생한 시점부터 수리 작업이 시작되는 시점까지의 시간입니다. 서비스 목록은 광범위하고 완전해야 합니다.

모니터링은 애플리케이션, 플랫폼, 인프라 및 네트워크를 포함한 애플리케이션 스택의 모든 계층을 포괄해야 합니다.

모니터링 전략에서는 회색 장애의 영향을 고려해야 합니다. 회색 장애에 대한 자세한 내용은 Advanced Multi-AZ Resilience Patterns 백서의 Gray failures를 참조하세요.

구현 단계

- 모니터링 간격은 필요한 복구 속도에 따라 달라집니다. 복구 시간은 복구에 걸리는 시간에 따라 결정되므로 이 시간과 Recovery Time Objective(RTO)를 고려하여 수집 빈도를 결정해야 합니다.
- 구성 요소 및 관리형 서비스에 대한 세부 모니터링을 구성합니다.
  - Auto Scaling 및 EC2 인스턴스에 대한 세부 모니터링이 필요한지 결정합니다. 세부 모니터링은 1분 간격 지표를 제공하고, 기본 모니터링은 5분 간격 지표를 제공합니다.
  - RDS에 대한 향상된 모니터링이 필요한지 결정합니다. 향상된 모니터링은 RDS 인스턴스의 에이전트를 사용하여 여러 프로세스 또는 스레드에 대한 유용한 정보를 얻습니다.
  - Lambda, API Gateway, Amazon EKS, Amazon ECS, 모든 유형의 로드 밸런서에 대한 주요 서버리스 구성 요소의 모니터링 요구 사항을 결정합니다.
  - Amazon S3, Amazon FSx, Amazon EFS, Amazon EBS에 대한 스토리지 구성 요소의 모니터링 요구 사항을 결정합니다.
- 비즈니스 핵심 성과 지표(KPI)를 측정하는 사용자 지정 지표를 생성합니다. 워크로드는 주요 비즈니스 기능을 구현하며, 이는 간접적인 문제 발생 시점을 파악하는 데 도움이 되는 KPI로 사용되어야 합니다.
- 사용자 Canary를 사용하여 사용자 경험에서 장애가 발생했는지 모니터링합니다. 가장 중요한 테스트 프로세스 중 하나는 고객 행동을 실행하고 시뮬레이션할 수 있는 가상 트랜잭션 테스트(canary 테스트라고도 하지만 카나리 배포와는 다름)입니다. 다양한 원격 위치에서 워크로드 엔드포인트에 대해 이러한 테스트를 지속적으로 실행합니다.
- 사용자 경험을 추적하는 사용자 지정 지표를 생성합니다. 고객의 경험을 계측할 수 있으면 소비자 경험이 저하되는 시기를 결정할 수 있습니다.

- 워크로드의 일부가 제대로 작동하지 않는 시기를 감지하고 리소스 규모를 자동 조정해야 하는 시점을 알려주도록 경보를 설정합니다. 경보를 사용하면 대시보드에 경보를 시각적으로 표시하고, Amazon SNS 또는 이메일을 통해 알림을 전송하며, Auto Scaling을 통해 워크로드의 리소스를 스케일 업 또는 스케일 다운할 수 있습니다.
- 지표를 시각화하는 대시보드를 생성합니다. 대시보드를 사용하면 추세, 이상값 및 기타 잠재적 문제의 지표를 시각적으로 표시하거나, 조사가 필요할 수 있는 문제를 표시할 수 있습니다.
- 서비스에 대한 분산 추적 모니터링을 생성합니다. 분산 모니터링을 사용하면 애플리케이션과 해당하는 기본 서비스의 성능을 파악하여 성능 문제 및 오류의 근본 원인을 식별하고 해결할 수 있습니다.
- 별도의 리전 및 계정에서 모니터링 시스템(CloudWatch 또는 X-Ray 사용) 대시보드 및 데이터 수집을 생성합니다.
- AWS Health를 사용하여 서비스 성능 저하에 대한 최신 정보를 확인하세요. AWS 사용자 알림를 통해 이메일 및 채팅 채널에 적합한 AWS Health 이벤트 알림을 생성하고, Amazon EventBridge를 통해 모니터링 및 알림 도구와 프로그래밍 방식으로 통합할 수 있습니다.

리소스

관련 모범 사례:

- 가용성 정의
- REL11-BP06 이벤트가 가용성에 영향을 미치는 경우 알림 전송

관련 문서:

- Amazon CloudWatch Synthetics로 사용자 Canary 생성 지원
- 인스턴스에 대한 세부 모니터링 활성화 또는 비활성화
- 확장 모니터링
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- 사용자 지정 지표 게시
- Amazon CloudWatch 경보 사용
- CloudWatch 대시보드 사용
- Account CloudWatch의 크로스 리전 및 크로스 계정 대시보드 사용
- X-Ray의 크로스 리전 및 크로스 계정 추적 사용

- Understanding availability

관련 비디오:

- Mitigating gray failures

관련 예제:

- One Observability 워크숍: Explore X-Ray

관련 도구:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 정상 리소스로 장애 조치

리소스 장애가 발생하는 경우 정상 리소스가 계속해서 요청을 처리해야 합니다. 위치 장애(예: 가용 영역 또는 AWS 리전)의 경우, 손상되지 않은 위치의 정상 리소스로 장애 조치할 수 있는 시스템을 갖추고 있어야 합니다.

서비스를 설계할 때는 리소스, 가용 영역 또는 리전에 부하를 분산하세요. 따라서 트래픽을 정상적인 리소스로 전환하여 개별 리소스의 장애 또는 중단을 완화할 수 있습니다. 장애 발생 시 서비스를 검색하고 라우팅하는 방법을 고려해 보세요.

장애 복구를 염두에 두고 서비스를 설계하세요. AWS에서는 장애에서 복구하는 시간과 데이터에 대한 영향을 최소화하는 방식으로 서비스가 설계됩니다. 자사 서비스는 기본적으로 한 리전 내의 여러 복제본에 저장된 요청만 승인하는 데이터 스토어를 사용합니다. 이들은 셀 기반 격리와 가용 영역에서 제공되는 장애 격리 기능을 사용하도록 구성됩니다. 자사 운영 절차에서는 자동화가 광범위하게 사용됩니다. 또한 서비스 중단 시 빠르게 복구할 수 있도록 교체 및 다시 시작 기능도 최적화됩니다.

장애 조치를 허용하는 패턴과 디자인은 AWS 플랫폼 서비스마다 다릅니다. 대부분의 AWS 기본 관리형 서비스는 기본적으로 다중 가용 영역(예: Lambda 또는 API Gateway)입니다. 다른 AWS 서비스(예: EC2 및 EKS)에서는 AZ에서 리소스 또는 데이터 스토리지의 장애 조치를 지원하기 위한 구체적인 모범 사례 설계가 필요합니다.

장애 조치 리소스가 정상인지 확인하고, 장애 조치 중인 리소스의 진행 상황을 추적하며, 비즈니스 프로세스 복구를 모니터링하도록 설정해야 합니다.

원하는 성과: 시스템은 새 리소스를 자동 또는 수동으로 사용하여 성능 저하를 복구할 수 있습니다.

일반적인 안티 패턴:

- 장애에 대한 계획은 계획 및 설계 단계의 일부가 아닙니다.
- RTO와 RPO가 설정되지 않았습니다.
- 모니터링이 충분하지 않아 장애가 발생한 리소스를 감지할 수 없습니다.
- 장애 도메인의 적절한 격리가 되지 않습니다.
- 다중 리전 장애 조치는 고려되지 않습니다.
- 장애 탐지는 장애 조치를 결정할 때 너무 민감하거나 공격적입니다.
- 장애 조치 설계를 테스트하거나 검증하지 않습니다.
- 자동 복구 자동화를 수행하지만 복구가 필요한 것을 알리지 않습니다.
- 너무 빨리 페일백하지 않도록 하는 완충 기간이 부족합니다.

이 모범 사례 확립의 이점: 점진적으로 성능이 저하되고 빠르게 복구되므로 장애 발생 시 신뢰성을 유지하는 복원력이 뛰어난 시스템을 구축할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS 서비스(예: Elastic Load Balancing 및 Amazon EC2 Auto Scaling)는 리소스 및 가용 영역 전체에 부하를 분산하는 데 도움이 됩니다. 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 설계가 더 복잡합니다. 예를 들어, 리전 간 읽기 전용 복제본을 사용하면 데이터를 여러 AWS 리전에 배포할 수 있습니다. 하지만 읽기 전용 복제본을 기본 복제본으로 승격한 다음 트래픽이 새 엔드포인트로 향하도록 하려면 여전히 장애 조치가 필요합니다. Amazon Route 53, Amazon Application Recovery Controller(ARC), Amazon CloudFront 및 AWS Global Accelerator는 AWS 리전에서 트래픽을 라우팅하는 데 도움이 될 수 있습니다.

AWS 서비스(예: Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge 또는 Amazon DynamoDB)는 AWS에 의해 다중 가용 영역에 자동으로 배포됩니다. 장애가 발생할 경우 이러한 AWS 서비스는 자동으로 트래픽을 정상 위치로 라우팅합니다. 데이터가 여러 가용 영역에 중복으로 저장되고 가용성이 유지됩니다.

Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS 또는 Amazon ECS의 경우 다중 AZ는 구성 옵션으로 제공됩니다. AWS에서는 장애 조치가 시작된 경우 정상 인스턴스로 트래픽을 전달할 수 있습니다. 이 장애 조치는 AWS에 의해 또는 고객의 요구에 따라 수행될 수 있습니다.

Amazon EC2 인스턴스, Amazon Redshift, Amazon ECS 작업 또는 Amazon EKS 포드의 경우 배포할 가용 영역을 선택할 수 있습니다. 일부 설계에서는 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 트래픽 장애 조치의 경우 재라우팅은 Amazon Route 53, Amazon Application Recovery Controller, AWS Global Accelerator, Route 53 Private DNS for VPC 또는 CloudFront를 통해 인터넷 도메인을 정의하고 상태 확인을 포함한 라우팅 정책을 할당하여 트래픽을 정상 리전으로 라우팅하는 방법을 제공할 수 있습니다. AWS Global Accelerator에서는 애플리케이션에 대한 고정 진입점 역할을 하는 고정 IP 주소를 제공한 다음, 성능 및 신뢰성 향상을 위해 인터넷 대신 AWS 글로벌 네트워크를 사용하여 선택한 AWS 리전 엔드포인트로 라우팅합니다.

구현 단계

- 모든 적절한 애플리케이션과 서비스를 위한 장애 조치 설계를 생성합니다. 각 아키텍처 구성 요소를 분리하고 각 구성 요소에 대한 RTO 및 RPO를 충족하는 장애 조치 설계를 생성합니다.
- 장애 조치 계획을 수립하는 데 필요한 모든 서비스를 갖춘 하위 환경(예: 개발 또는 테스트)을 구성합니다. 코드형 인프라(IaC)를 사용하여 솔루션을 배포해 반복성을 보장합니다.
- 복구 사이트(예: 보조 리전)를 구성하여 장애 조치 설계를 구현하고 테스트합니다. 필요한 경우 테스트 리소스를 임시로 구성하여 추가 비용을 제한할 수 있습니다.
- AWS를 통해 어떤 장애 조치 계획을 자동화할지, DevOps 프로세스로 어떤 장애 조치 계획을 자동화할 수 있는지, 어떤 장애 조치 계획을 수동으로 할지 결정하세요. 각 서비스의 RTO 및 RPO를 문서화하고 측정합니다.
- 장애 조치 플레이북을 만들고 각 리소스, 애플리케이션, 서비스를 장애 조치하기 위한 모든 단계를 포함하세요.
- 페일백 플레이북을 만들고 각 리소스, 애플리케이션, 서비스를 페일백하기 위한 모든 단계(타이밍 포함)를 포함합니다.
- 계획을 세워 플레이북을 시작하고 연습하세요. 시뮬레이션과 카오스 테스트를 사용하여 플레이북 단계 및 자동화를 테스트하세요.
- 위치 장애(예: 가용 영역 또는 AWS 리전)의 경우, 중단되지 않은 위치의 정상 리소스로 장애 조치할 수 있는 시스템을 갖추고 있어야 합니다. 장애 조치 테스트 전에 할당량, 자동 규모 조정 수준, 실행 중인 리소스를 확인하세요.

리소스

관련 Well-Architected 모범 사례:

- REL13 - 재해 복구 계획
- REL10 - 장애 격리를 사용하여 워크로드 보호

관련 문서:

- Setting RTO and RPO Targets
- Failover using Route 53 Weighted routing
- Amazon Application Recovery Controller를 사용하여 재해 복구
- EC2 with autoscaling
- EC2 Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Amazon Application Recovery Controller를 사용하여 트래픽 전환
- Lambda with an Application Load Balancer and Failover
- ACM Replication and Failover
- Parameter Store Replication and Failover
- ECR cross region replication and Failover
- Secrets manager cross region replication configuration
- Enable cross region replication for EFS and Failover
- EFS Cross Region Replication and Failover
- Networking Failover
- S3 Endpoint failover using MRAP
- S3에 대한 크로스 리전 복제 생성
- AWS에서 리전 간 장애 조치 및 Graceful 장애 복구에 대한 지침
- Failover using multi-region global accelerator
- Failover with DRS

관련 예제:

- Disaster Recovery on AWS

- Elastic Disaster Recovery on AWS

REL11-BP03 모든 계층에서 복구 자동화

장애가 감지되면 자동화된 기능을 사용하여 수정 작업을 수행합니다. 성능 저하는 내부 서비스 메커니즘을 통해 자동으로 해결되거나 수정 조치를 통해 리소스를 재시작하거나 제거해야 할 수 있습니다.

자체 관리 애플리케이션 및 크로스 리전 간 복구를 위해 기존 모범 사례에서 복구 설계 및 자동화된 복구 프로세스를 가져올 수 있습니다.

리소스를 재시작하거나 제거하는 기능은 장애를 해결하는 데 중요한 도구입니다. 가장 좋은 방법은 가능한 경우 서비스를 상태 비저장으로 만드는 것입니다. 이렇게 하면 리소스 재시작 시 데이터 손실 또는 가용성이 손실되는 것을 방지할 수 있습니다. 클라우드에서는 재시작의 일부로 전체 리소스(예: 컴퓨팅 인스턴스 또는 서버리스 함수)를 대체할 수 있으며 이러한 대체는 일반적으로 필수적입니다. 재시작은 그 자체로 장애를 복구할 수 있는 단순하면서도 신뢰할 수 있는 방법입니다. 워크로드에는 다양한 유형의 장애가 발생합니다. 장애는 하드웨어, 소프트웨어, 통신 및 작업 과정에서 발생할 수 있습니다.

재시작 또는 재시도는 네트워크 요청에도 적용됩니다. 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 사례로 처리하는 대신 지수 백오프 및 지터를 통해 제한적으로 재시도하는 비슷한 전략이 적용됩니다. 재시작 기능은 복구 중심 컴퓨팅 및 고가용성 클러스터 아키텍처에 포함된 복구 메커니즘입니다.

원하는 성과: 장애 감지를 해결하기 위해 자동화된 조치가 수행됩니다.

일반적인 안티 패턴:

- 자동 규모 조정 없이 리소스를 프로비저닝합니다.
- 인스턴스 또는 컨테이너에 개별적으로 애플리케이션을 배포합니다.
- 자동 복구를 사용하지 않고 여러 위치에 배포할 수 없는 애플리케이션을 배포합니다.
- 자동 규모 조정 및 자동 복구로 복구하지 못한 애플리케이션을 수동으로 복구합니다.
- 데이터베이스 장애 조치를 자동화할 수 없습니다.
- 트래픽을 새 엔드포인트로 재라우팅하는 자동화된 방법이 부족합니다.
- 스토리지 복제가 없습니다.

이 모범 사례 확립의 이점: 자동 복구를 통해 평균 복구 시간을 줄이고 가용성을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon EKS 또는 기타 Kubernetes 서비스를 위한 설계에는 최소 및 최대 복제본 또는 상태 저장 세트와 최소 클러스터 및 노드 그룹 크기가 모두 포함되어야 합니다. 이러한 메커니즘은 Kubernetes 컨트롤 플레인을 사용하여 장애를 자동으로 해결하면서 지속적으로 사용할 수 있는 최소한의 처리 리소스를 제공합니다.

컴퓨팅 클러스터를 사용하여 로드 밸런서를 통해 액세스하는 설계 패턴은 Auto Scaling 그룹을 활용해야 합니다. Elastic Load Balancing(ELB)은 애플리케이션 인바운드 트래픽을 하나 이상의 가용 영역에 있는 여러 대상 및 가상 어플라이언스에 자동으로 분산합니다.

부하 분산을 사용하지 않는 클러스터링된 컴퓨팅 기반 설계는 최소 하나 이상의 노드 손실을 고려하여 크기가 설계되어야 합니다. 이렇게 하면 새 노드를 복구하는 동안 서비스가 잠재적으로 줄어든 용량으로 계속 운영될 수 있습니다. 서비스 예로는 Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK, Amazon OpenSearch Service가 있습니다. 이러한 서비스 중 다수는 추가 자동 복구 기능을 사용하여 설계할 수 있습니다. 일부 클러스터 기술은 노드가 손실되면 자동 또는 수동 워크플로를 트리거하여 새 노드를 다시 생성하는 경고를 생성해야 합니다. AWS Systems Manager를 사용하여 이 워크플로를 자동화하여 문제를 신속하게 해결할 수 있습니다.

Amazon EventBridge를 사용하면 CloudWatch 경보 또는 다른 AWS 서비스의 상태 변경과 같은 이벤트를 모니터링하고 필터링할 수 있습니다. 그런 다음 이벤트 정보를 기반으로 AWS Lambda, Systems Manager Automation 또는 다른 대상을 간접 호출하여 워크로드에 대한 사용자 지정 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. 대규모 교체(예: 전체 가용 영역이 손실됨)의 경우 정적 안정성을 통해 고가용성을 유지하는 것이 좋습니다.

구현 단계

- Auto Scaling 그룹을 사용하여 워크로드에 티어를 배포합니다. Auto Scaling은 상태 비저장 애플리케이션에서 자가 복구를 수행하고 용량을 추가 및 제거할 수 있습니다.
- 앞서 언급한 컴퓨팅 인스턴스의 경우 로드 밸런싱을 사용하고 적절한 유형의 로드 밸런서를 선택합니다.
- Amazon RDS에서 복구를 고려합니다. 대기 인스턴스를 사용하여 대기 인스턴스로의 자동 장애 조치를 구성합니다. Amazon RDS 읽기 전용 복제본의 경우 읽기 전용 복제본을 기본 복제본으로 만들려면 자동화된 워크플로가 필요합니다.
- 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 복구

를 사용하여 장애가 발생한 하드웨어를 교체하고 인스턴스를 다시 시작할 수 있습니다. 인스턴스 메타데이터 및 관련 IP 주소가 유지되며, EBS 볼륨과 Amazon Elastic File System 및 File Systems for Lustre 및 File Systems for Windows에 대한 탑재 지점도 유지됩니다. AWS OpsWorks를 사용하면 계층 수준에서 EC2 인스턴스의 자동 복구를 구성할 수 있습니다.

- 자동 규모 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.
- 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 간접 호출하여 워크로드에 대한 사용자 지정 수정 로직을 실행할 수 있습니다.

리소스

관련 모범 사례:

- 가용성 정의
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

관련 문서:

- AWS Auto Scaling 작동 방식
- Amazon EC2 자동 복구
- Amazon Elastic Block Store(Amazon EBS)
- Amazon Elastic File System(Amazon EFS)
- What is Amazon FSx for Lustre?
- What is Amazon FSx for Windows File Server?
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- AWS Step Functions란 무엇입니까?
- AWS Lambda란 무엇입니까?
- Amazon EventBridge란 무엇인가요?
- Amazon CloudWatch 경보 사용
- Amazon RDS Failover
- SSM - Systems Manager Automation

- Resilient Architecture Best Practices

관련 비디오:

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

관련 예제:

- Amazon RDS Failover 워크숍

관련 도구:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 플레인 사용

컨트롤 플레인은 리소스를 생성, 읽기 및 설명, 업데이트, 삭제, 나열(CRUDL)하는 데 사용되는 관리 API를 제공하는 반면, 데이터 플레인은 일상적인 서비스 트래픽을 처리합니다. 복원력에 영향을 미칠 수 있는 이벤트에 대한 복구 또는 완화 대응을 구현할 때는 최소한의 컨트롤 플레인 작업을 사용하여 서비스를 복구, 재조정, 복원, 복구 또는 장애 조치하는 데 집중합니다. 데이터 플레인 작업은 이러한 성능 저하 이벤트가 발생하는 동안의 모든 활동을 대체해야 합니다.

예를 들어, 새 컴퓨팅 인스턴스 시작, 블록 스토리지 생성, 대기열 서비스 설명 등은 모두 컨트롤 플레인 작업입니다. 컴퓨팅 인스턴스를 시작할 때 컨트롤 플레인은 용량이 있는 물리적 호스트 찾기, 네트워크 인터페이스 할당, 로컬 블록 스토리지 볼륨 준비, 자격 증명 생성, 보안 규칙 추가와 같은 여러 작업을 수행해야 합니다. 컨트롤 플레인은 복잡한 오케스트레이션이 필요한 경우가 많습니다.

원하는 성과: 리소스가 손상된 상태가 되면 시스템은 트래픽을 손상된 리소스에서 정상 리소스로 전환하여 자동 또는 수동으로 복구할 수 있습니다.

일반적인 안티 패턴:

- 트래픽을 재라우팅하기 위해 DNS 레코드 변경에 의존합니다.
- 불충분하게 프로비저닝된 리소스로 인해 손상된 구성 요소를 대체하기 위한 컨트롤 플레인 규모 조정 작업에 의존합니다.
- 광범위한 다중 서비스, 다중 API 컨트롤 플레인 조치를 활용하여 모든 범주의 장애를 해결합니다.

이 모범 사례 확립의 이점: 자동화된 문제 해결의 성공률이 높아지면 평균 시간이 단축되고 워크로드의 가용성이 향상됩니다.

이 모범 사례가 확립되지 않은 경우 노출되는 위험 수준: 중간: 특정 유형의 서비스 성능 저하에서는 컨트롤 플레인이 영향을 받습니다. 개선을 위해 컨트롤 플레인을 광범위하게 사용하는 것에 의존하면 복구 시간(RTO)과 평균 복구 시간(MTTR)이 증가할 수 있습니다.

구현 지침

데이터 플레인 작업을 제한하려면 서비스를 복원하는 데 필요한 조치가 무엇인지 각 서비스를 평가하세요.

Amazon Application Recovery Controller를 활용하여 DNS 트래픽을 전환합니다. 이러한 기능은 애플리케이션의 장애 복구 성능을 지속적으로 모니터링하며, 이를 통해 여러 AWS 리전, 가용 영역 및 온프레미스에서 애플리케이션 복구를 제어할 수 있습니다.

Route 53 라우팅 정책은 컨트롤 플레인을 사용하므로 복구 시 컨트롤 플레인을 사용하지 마세요. Route 53 데이터 플레인은 DNS 쿼리에 응답하고 상태 확인을 수행 및 평가합니다. 전 세계에 분산되어 있으며, 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(미국 동부, 버지니아 북부)에 위치합니다. 두 시스템 모두 신뢰성이 높게 구축되었지만 컨트롤 플레인은 SLA에 포함되지 않습니다. 데이터 플레인의 복원력 높은 설계가 컨트롤 플레인과 달리 가용성을 유지하는 상황이 드물게 있을 수 있습니다. 재해 복구 및 장애 조치 메커니즘에는 데이터 플레인 기능을 사용하여 가능한 한 최고 수준의 신뢰성을 제공합니다.

인시던트 중에 컨트롤 플레인을 사용하지 않도록 컴퓨팅 인프라를 정적으로 안정적으로 설계합니다. 예를 들어 Amazon EC2 인스턴스를 사용하는 경우 새 인스턴스를 수동으로 프로비저닝하거나 Auto Scaling 그룹에 응답으로 인스턴스를 추가하도록 지시하지 마십시오. 최고 수준의 복원력을 얻으려면 장애 조치에 사용되는 클러스터에 충분한 용량을 프로비저닝하세요. 이 용량 임곗값을 제한해야 하는 경우 전체 엔드 투 엔드 시스템에 제한을 설정하여 총 트래픽이 제한된 리소스 세트에 도달하도록 안전하게 제한합니다.

Amazon DynamoDB, Amazon API Gateway, 로드 밸런서 및 AWS Lambda 서버리스와 같은 서비스의 경우 이러한 서비스를 사용하면 데이터 플레인을 활용할 수 있습니다. 하지만 새 함수, 로드 밸런서, API 게이트웨이 또는 DynamoDB 테이블을 생성하는 것은 컨트롤 플레인 작업이므로 이벤트 준비 및 장애 조치 리허설로 성능 저하 전에 완료해야 합니다. Amazon RDS의 경우, 데이터 플레인 작업을 통해 데이터에 액세스할 수 있습니다.

데이터 플레인, 컨트롤 플레인 및 AWS가 고가용성 목표를 충족하기 위해 서비스를 구축하는 방법에 대한 자세한 내용은 가용 영역을 사용한 정적 안정성DMF 참조하세요.

데이터 플레인을 사용할 작업과 컨트롤 플레인을 사용할 작업을 파악합니다.

구현 단계

성능 저하 이벤트 후 복원해야 하는 각 워크로드에 대해 장애 조치 런북, 고가용성 설계, 자동 복구 설계 또는 HA 리소스 복원 계획을 평가하세요. 컨트롤 플레인 작업으로 간주될 수 있는 각 작업을 식별하세요.

컨트롤 작업을 데이터 플레인 작업으로 변경하는 것을 고려해 보세요.

- 오토 스케일링(컨트롤 플레인)과 사전 규모 조정된 Amazon EC2 리소스(데이터 플레인) 비교
- Amazon EC2 인스턴스 조정(컨트롤 플레인)과 AWS Lambda 조정(데이터 플레인) 비교
- Kubernetes를 사용하는 모든 설계와 컨트롤 플레인 작업의 특성을 평가하세요. 포드를 추가하는 것은 Kubernetes의 데이터 플레인 작업입니다. 작업은 포드를 추가하고 노드는 추가하지 않는 것으로 제한해야 합니다. 과다 프로비저닝된 노드 사용은 컨트롤 플레인 작업을 제한하는 데 선호되는 방법입니다.

데이터 플레인 작업이 동일한 문제 해결에 영향을 줄 수 있는 다른 접근 방식을 고려해 보세요.

- Route 53 레코드 변경(컨트롤 플레인) 또는 Amazon Application Recovery Controller(데이터 플레인)
- 보다 자동화된 업데이트를 위한 Route 53 상태 확인

서비스가 업무상 중요한 경우 영향을 받지 않는 리전에서 더 많은 컨트롤 플레인 및 데이터 플레인 작업을 수행할 수 있도록 보조 리전의 일부 서비스를 고려해 보세요.

- 기본 리전의 Amazon EC2 Auto Scaling 또는 Amazon EKS와 보조 리전의 Amazon EC2 Auto Scaling 또는 Amazon EKS 비교 및 보조 리전으로의 트래픽 라우팅(컨트롤 플레인 작업)
- 보조 기본 리전에서 읽기 전용 복제본을 만들거나 기본 리전에서 동일한 작업 시도(컨트롤 플레인 작업)

리소스

관련 모범 사례:

- 가용성 정의

- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지

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
- Amazon DynamoDB API(컨트롤 플레인 및 데이터 플레인)
- AWS Lambda 실행(컨트롤 플레인 및 데이터 플레인으로 분할)
- AWS Elemental MediaStore 데이터 플레인
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- Amazon Application Recovery Controller란 무엇입니까?
- Kubernetes 컨트롤 플레인 및 데이터 플레인

관련 비디오:

- Back to Basics - Using Static Stability
- Building resilient multi-site workloads using AWS global services

관련 예제:

- Amazon Application Recovery Controller 소개
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack

- 가용 영역을 사용한 정적 안정성

관련 도구:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 정적 안정성을 사용하여 바이모달 동작 방지

워크로드는 정적으로 안정적인 상태를 유지하고 단일 정상 모드에서만 작동해야 합니다. 바이모달 동작은 정상 모드와 장애 모드에서 워크로드가 서로 다른 동작을 보일 때를 말합니다.

예를 들어 서로 다른 가용 영역에서 새 인스턴스를 시작하여 가용 영역 장애 복구를 시도할 수 있습니다. 이로 인해 장애 모드 중에 바이모달 응답이 발생할 수 있습니다. 그러나 이 방법 대신 정적으로 안정적이며 한 모드에서만 작동하는 워크로드를 구축해야 합니다. 이 예제에서 이러한 인스턴스는 장애가 발생하기 전에 두 번째 가용 영역에 프로비저닝되었어야 합니다. 이 정적 안정성 설계는 워크로드가 단일 모드에서만 작동하는지 확인합니다.

원하는 성과: 정상 모드와 장애 모드에서 워크로드가 바이모달 동작을 보이지 않습니다.

일반적인 안티 패턴:

- 장애 범위에 관계없이 항상 리소스를 프로비저닝할 수 있다고 가정합니다.
- 장애 발생 시 동적으로 리소스를 확보하려고 시도합니다.
- 장애가 발생할 때까지 여러 영역 또는 리전에 걸쳐 적절한 리소스를 프로비저닝하지 않습니다.
- 컴퓨팅 리소스에 대해서만 정적이고 안정적인 설계를 고려합니다.

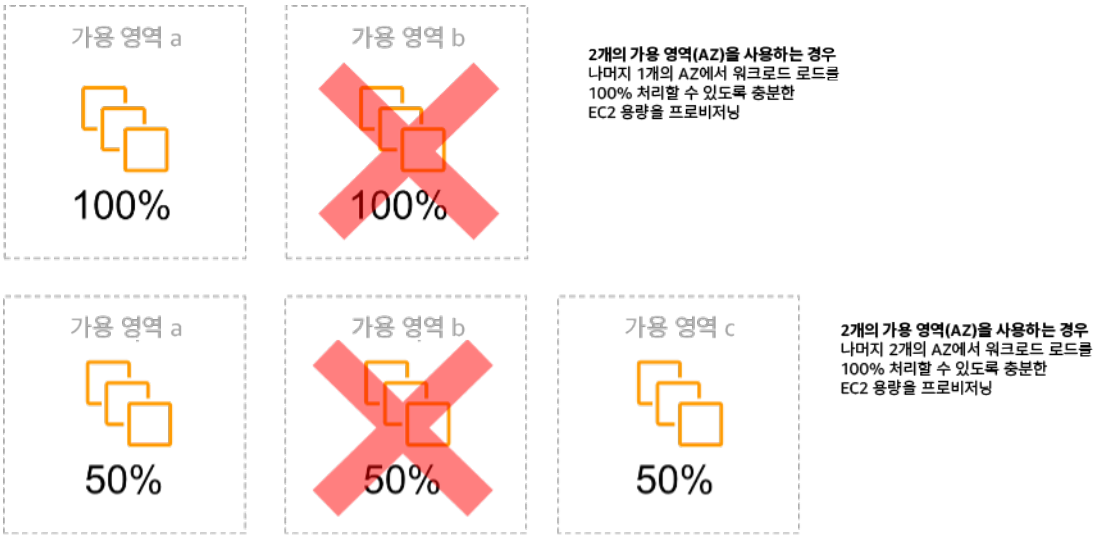
이 모범 사례 확립의 이점: 정적이고 안정적인 설계로 실행되는 워크로드는 정상 및 장애 이벤트 발생 시 예측 가능한 결과를 얻을 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

바이모달 동작은 워크로드가 정상 모드와 장애 모드에서 다른 동작을 보일 때 발생합니다. 예를 들어 가용 영역에 장애가 발생할 경우 새 인스턴스를 시작하는 방법을 사용할 수 있습니다. 바이모달 동작의 예로는 한 AZ가 제거된 경우 안정적인 Amazon EC2 설계가 워크로드 로드를 처리하기에 충분한 인스턴스를 각 가용 영역에 프로비저닝하는 경우를 들 수 있습니다. 그런 다음 Elastic Load Balancing 또는

Amazon Route 53 상태를 확인하여 손상된 인스턴스로부터 로드를 이동합니다. 트래픽이 전환된 후에는 AWS Auto Scaling을 사용하여 장애가 발생한 영역의 인스턴스를 비동기식으로 교체하고 정상 영역에서 인스턴스를 시작합니다. 컴퓨팅 배포(예: EC2 인스턴스 또는 컨테이너)에서 정적 안정성은 최고의 신뢰성을 제공합니다.



가용 영역에 걸친 EC2 인스턴스의 정적 안정성

모든 복원력 사례에서 워크로드를 유지 관리하는 데 따르는 비즈니스 가치 및 이 모델의 비용을 이 정적 안정성과 비교해야 합니다. 컴퓨팅 용량을 적게 프로비저닝하고 장애 발생 시 새 인스턴스를 시작하는 방법이 비용은 더 저렴하지만, 대규모 장애(예: 가용 영역 또는 리전 장애)의 경우 이 접근 방식은 영향을 받지 않는 영역 또는 리전에서 제공되는 충분한 리소스와 운영 플레인을 활용하기 때문에 덜 효과적입니다.

따라서 워크로드의 신뢰성 요구 사항과 비용 요구 사항도 비교해야 합니다. 정적 안정성 아키텍처는 가용 영역에 분산된 컴퓨팅 인스턴스, 데이터베이스 읽기 전용 복제본 설계, Kubernetes(Amazon EKS) 클러스터 설계, 다중 리전 장애 조치 아키텍처 등 다양한 아키텍처에 적용됩니다.

또한 각 영역에서 더 많은 리소스를 사용하여 더 정적으로 안정적인 설계를 구현할 수도 있습니다. 영역을 더 추가할수록 정적 안정성을 유지하는 데 필요한 추가 컴퓨팅 용량은 줄어듭니다.

바이모달 동작의 한 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 이어질 수 있습니다. 이 부정적인 피드백 루프는 워크로드의 가용성에 영향을 미칩니다. 대신 정적으로 안정적이며 한 모드에서만 작동하는 시스템을 구축할 수 있습니다. 일정한 작업을 수행하고 항상 고정된 케이던스에서 구성 상태를 새로 고치는 것이 정적으로 안정적인 설계의 하나일 것입니다. 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 변경될 수 있고 장애를 초래할 가능성이 높습니다.

중요 워크로드를 평가하여 이러한 유형의 복원력 설계가 필요한 워크로드를 결정합니다. 중요하다고 판단되는 워크로드에 대해서는 각 애플리케이션 구성 요소를 검토해야 합니다. 정적 안정성 평가가 필요한 서비스 유형의 예는 다음과 같습니다.

- 컴퓨팅: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- 데이터베이스: Amazon Redshift, Amazon RDS, Amazon Aurora
- 스토리지: Amazon S3(단일 영역), Amazon EFS(탑재), Amazon FSx(탑재)
- 로드 밸런서: 특정 설계 시

구현 단계

- 정적으로 안정적이고 한 모드에서만 작동하는 시스템을 구축합니다. 이 경우 가용 영역 또는 리전 하나가 제거된 경우 각 가용 영역 또는 리전에서 워크로드 용량을 처리할 만큼 충분한 인스턴스를 프로비저닝합니다. 다음과 같은 다양한 서비스를 사용하여 정상 리소스로 라우팅할 수 있습니다.
  - 크로스 리전 DNS 라우팅
  - MRAP Amazon S3 다중 리전 라우팅
  - AWS Global Accelerator
  - Amazon Application Recovery Controller
- 단일 기본 인스턴스 또는 읽기 전용 복제본의 손실을 고려하도록 데이터베이스 읽기 복제본을 구성합니다. 읽기 전용 복제본으로 트래픽을 처리하는 경우 각 가용 영역 및 각 리전의 용량은 영역 또는 리전 장애 발생 시 필요한 전체 용량과 동일해야 합니다.
- 가용 영역 장애 발생 시 저장된 데이터에 대해 정적으로 안정적으로 설계된 Amazon S3 스토리지에서 중요한 데이터를 구성합니다. Amazon S3 One Zone-IA 스토리지 클래스를 사용하는 경우, 해당 영역이 손실되면 저장된 데이터에 대한 액세스가 최소화되므로 이 스토리지 클래스를 정적으로 안정적인 것으로 간주해서는 안 됩니다.
- 로드 밸런서가 잘못 구성되거나 특정 가용 영역을 서비스하도록 설계되는 경우가 간혹 있습니다. 이 경우 보다 복잡한 설계에서 여러 AZ에 워크로드를 분산하는 것이 정적으로 안정적인 설계일 수 있습니다. 원래 설계는 보안, 지연 시간 또는 비용상의 이유로 영역 간 트래픽을 줄이는 데 사용될 수 있습니다.

리소스

관련 Well-Architected 모범 사례:

- 가용성 정의
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 플레인 사용

관련 문서:

- Minimizing Dependencies in a Disaster Recovery Plan
- Amazon Builders' Library: 가용 영역을 사용한 정적 안정성
- Fault Isolation Boundaries
- 가용 영역을 사용한 정적 안정성
- 다중 영역 RDS
- Minimizing Dependencies in a Disaster Recovery Plan
- 크로스 리전 DNS 라우팅
- MRAP Amazon S3 다중 리전 라우팅
- AWS Global Accelerator
- Amazon Application Recovery Controller
- 단일 영역 Amazon S3
- Cross Zone Load Balancing

관련 비디오:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 이벤트가 가용성에 영향을 미치는 경우 알림 전송

임곗값 위반이 감지되면 문제를 일으킨 이벤트가 자동으로 해결된 경우에도 알림이 전송됩니다.

자동 복구를 사용하면 워크로드의 신뢰성을 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.

복원력이 뛰어난 시스템은 성능 저하 이벤트가 해당 팀에 즉시 전달되도록 설계되었습니다. 이러한 알림은 하나 이상의 통신 채널을 통해 전송되어야 합니다.

원하는 성과: 오류율, 지연 시간 또는 기타 중요한 핵심 성과 지표(KPI)와 같은 임곗값을 위반하면 운영 팀에 즉시 알림이 전송되므로 이러한 문제를 최대한 빨리 해결하고 사용자에게 미치는 영향을 피하거나 최소화할 수 있습니다.

일반적인 안티 패턴:

- 너무 많은 경보를 전송합니다.
- 실행 불가능한 경보를 전송합니다.
- 경보 임곗값을 너무 높거나(민감도 높음) 너무 낮게(민감도 낮음) 설정합니다.
- 외부 종속성에 대한 경보를 전송하지 않습니다.
- 모니터링 및 경보를 설계할 때 회색 장애를 고려하지 않습니다.
- 복구 자동화를 수행하지만 해당 팀에 복구가 필요하다는 사실을 알리지 않습니다.

이 모범 사례 확립의 이점: 운영 팀과 비즈니스 팀은 복구 알림을 통해 서비스 저하를 인지하고 즉시 대응하여 평균 탐지 시간(MTTD)과 평균 복구 시간(MTTR)을 모두 최소화할 수 있습니다. 또한 복구 이벤트에 대한 알림이 전송되면 어쩌다 발생하는 문제도 지나치지 않을 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간. 적절한 모니터링 및 이벤트 알림 메커니즘을 구현하지 않으면 자동 복구로 해결된 문제를 포함하여 문제의 패턴을 감지하지 못할 수 있습니다. 팀은 사용자가 고객 서비스에 문의할 때나 우연한 경우에만 시스템 성능 저하 사실을 인지합니다.

구현 지침

모니터링 전략을 정의할 때 경보가 울리는 것은 흔한 이벤트입니다. 이 이벤트에는 경보의 식별자, 즉 경보 상태(예: `IN ALARM` 또는 `OK`) 및 유발 원인에 대한 세부 정보가 포함되어 있을 것입니다. 대부분의 경우 경보 이벤트가 감지되고 이메일 알림이 전송되어야 합니다. 이것이 경보에 대한 작업의 예입니다. 경보 알림은 적절한 사용자에게 문제가 있음을 알려주기 때문에 관찰성에서 매우 중요합니다. 그러나 관찰성 솔루션에서 이벤트에 대한 작업이 성숙해지면 사람의 개입 없이 자동으로 문제를 해결할 수 있습니다.

KPI 모니터링 경보가 설정되면 임곗값을 초과할 경우 해당 팀에 알림이 전송되어야 합니다. 이러한 알림은 성능 저하를 해결하기 위한 자동화된 프로세스를 트리거하는 데에도 사용될 수 있습니다.

보다 복잡한 임곗값 모니터링의 경우 복합 경보를 고려해야 합니다. 복합 경보는 여러 KPI 모니터링 경보를 사용하여 운영 비즈니스 로직에 기반한 알림을 생성합니다. 이메일을 보내거나 Amazon SNS 통

합 또는 Amazon EventBridge를 사용하여 서드파티 인시던트 추적 시스템에 인시던트를 기록하도록 CloudWatch 경보를 구성할 수 있습니다.

구현 단계

워크로드 모니터링 방식에 따라 다음과 같은 다양한 유형의 경보를 생성합니다.

- 애플리케이션 경보는 워크로드의 일부가 제대로 작동하지 않는 경우를 감지하는 데 사용됩니다.
- 인프라 경보는 리소스 규모를 조정할 시기를 나타냅니다. 경보를 사용하면 대시보드에 경보를 시각적으로 표시하고, Amazon SNS 또는 이메일을 통해 알림을 전송하며, Auto Scaling을 통해 워크로드의 리소스를 스케일 인 또는 스케일 아웃할 수 있습니다.
- 지정된 평가 기간에 지표가 정적 임곗값을 위반하는 시점을 모니터링하기 위해 간단한 정적 경보를 생성할 수 있습니다.
- 여러 소스의 복잡한 경보에 대해서는 복합 경보를 고려할 수 있습니다.
- 경보가 생성되면 적절한 알림 이벤트를 생성합니다. Amazon SNS API 간접 호출을 직접 수행하여 알림을 보내고 문제 해결 또는 커뮤니케이션을 위한 자동화를 연결할 수 있습니다.
- AWS Health를 사용하여 서비스 성능 저하에 대한 최신 정보를 확인하세요. AWS 사용자 알림를 통해 이메일 및 채팅 채널에 적합한 AWS Health 이벤트 알림을 생성하고, Amazon EventBridge를 통해 모니터링 및 알림 도구와 프로그래밍 방식으로 통합할 수 있습니다.

리소스

관련 Well-Architected 모범 사례:

- 가용성 정의

관련 문서:

- 정적 임곗값을 기반으로 CloudWatch 경보 생성
- Amazon EventBridge란 무엇인가요?
- What is Amazon Simple Notification Service?
- 사용자 지정 지표 게시
- Amazon CloudWatch 경보 사용
- CloudWatch 복합 경보 설정
- What's new in AWS Observability at re:Invent 2022

관련 도구:

- CloudWatch
- CloudWatch X-Ray

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)을 충족하도록 제품을 설계합니다. 가용성 목표 또는 가동 시간 SLA를 게시하거나 비공개로 동의하는 경우 아키텍처 및 운영 프로세스가 이를 지원하도록 설계되었는지 확인합니다.

원하는 성과: 각 애플리케이션에는 비즈니스 성과를 달성하기 위해 모니터링 및 유지 관리할 수 있는 성과 지표에 대해 정의된 가용성 및 SLA 목표가 있습니다.

일반적인 안티 패턴:

- SLA를 설정하지 않고 워크로드를 설계 및 배포합니다.
- SLA 지표가 근거나 비즈니스 요구 사항 없이 너무 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 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

애플리케이션 설계는 비즈니스, 운영 및 재무 목표에서 파생된 다양한 요구 사항 세트를 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 관찰성과 계측이 중요합니다.
- 고장 빈도 감소(MTBF 연장), 고장 감지 시간 단축(MTTD 단축), 수리 시간 단축(MTTR 단축)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 사용되나요?
  - 워크로드는 내결함성을 어떻게 구현하나요?
  - 규모 조정, 자동 규모 조정, 중복성 및 고가용성 구성 요소에 대한 디자인 패턴은 무엇인가요?
  - 데이터 일관성 및 가용성에 대한 요구 사항은 무엇인가요?
  - 리소스 절약 또는 리소스 정적 안정성에 대한 고려 사항이 있나요?
  - 서비스 종속성은 무엇인가요?
- 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-BP04 카오스 엔지니어링을 이용한 복원력 테스트
- REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의
- 워크로드 상태 파악

관련 문서:

- Availability with redundancy
- 신뢰성 원칙 - 가용성
- Measuring availability
- AWS Fault Isolation Boundaries
- 복원력을 위한 공동 책임 모델
- 가용 영역을 사용한 정적 안정성
- AWS 서비스 수준에 관한 계약(SLA)
- Guidance for Cell-based Architecture on AWS
- AWS infrastructure
- Advanced Multi-AZ Resilience Patterns whitepaper

관련 서비스:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12. 신뢰성은 어떻게 테스트하나요?

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 확장성 및 성능 요구 사항 테스트
- REL12-BP04 카오스 엔지니어링을 이용한 복원력 테스트
- REL12-BP05 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(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 계획되지 않은 의사 결정을 내립니다.
- 데이터를 검색할 수 있을 만큼 오래 지표 및 이벤트를 유지하지 않습니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 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 확인하며,

결함 위치를 구분하고, 발생 요인을 확인(인시던트 사후 분석 수행)하는 데 필요한 정보와 지침을 포함해야 합니다.

- 플레이북을 코드로 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 간접 호출하거나 수행할 수도 있습니다. 또는 식별된 이벤트의 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 무엇인가요?
  - 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
- Using Canaries (Amazon CloudWatch Synthetics)
- Amazon EventBridge란 무엇인가요?
- AWS Lambda란 무엇입니까?

관련 예제:

- Automating operations with Playbooks and Runbooks

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 개발

합니다. 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.

원하는 성과: 팀이 인시던트 이후 분석을 처리하기 위해 일관되고 합의된 접근 방식을 취합니다. 한 가지 메커니즘으로 오류 수정 (COE) 프로세스가 있습니다. COE 프로세스는 인시던트의 근본 원인을 식별, 이해 및 해결하는 동시에 동일한 인시던트가 다시 발생할 가능성을 제한하는 메커니즘과 가드레일을 구축하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 발생 요인을 찾지만, 다른 잠재적 문제와 해결 방법을 계속 자세히 살펴보지는 않습니다.
- 인적 오류 원인만 식별하며, 인적 오류를 예방할 수 있는 교육이나 자동화는 제공하지 않습니다.
- 근본 원인을 이해하기보다는 책임을 전가하는 데 집중하여 공포의 문화를 조성하고 열린 의사소통을 방해합니다.
- 인사이트를 공유하는 데 실패하여 인시던트 분석 결과를 소수의 사람만 알고 있고, 배운 교훈을 다른 사람들이 활용하지 못하게 합니다.
- 제도적 지식을 포착할 메커니즘이 없기 때문에 학습한 교훈을 업데이트된 모범 사례의 형태로 보존하지 못해 귀중한 인사이트를 잃고 근본 원인이 동일하거나 유사한 인시던트가 반복적으로 발생합니다.

이 모범 사례 확립의 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

우수한 인시던트 사후 분석은 시스템의 다른 위치에서 사용되는 아키텍처 패턴이 있는 문제에 대해 공통 솔루션을 제안할 수 있는 기회를 제공합니다.

COE 프로세스의 초석은 문제를 문서화하고 해결하는 것입니다. 주요 근본 원인을 문서로 작성하고 검토 및 해결할 수 있는 표준 방식을 정의하는 것이 좋습니다. 인시던트 사후 분석 프로세스에 명확한 소유권을 부여합니다. 인시던트 조사 및 후속 조치를 감독할 책임이 있는 팀 또는 개인을 지정합니다.

책임을 전가하기보다는 학습과 개선에 초점을 맞추는 문화를 장려합니다. 개인에게 불이익을 주는 것이 아니라 향후 인시던트를 예방하는 것이 목표라는 점을 강조합니다.

인시던트 사후 분석을 수행하기 위한 잘 정의된 절차를 개발합니다. 이러한 절차에는 취해야 할 단계, 수집할 정보 및 분석 중에 해결해야 할 주요 질문이 요약되어 있어야 합니다. 즉각적인 원인을 넘어 근본 원인과 기여 요인을 식별하여 사고를 철저히 조사합니다. 다섯 가지 이유와 같은 기법을 사용하여 근본적인 문제를 심층적으로 분석합니다.

인시던트 분석을 통해 얻은 교훈의 리포지토리를 유지 관리합니다. 이러한 제도적 지식을 향후 인시던트 및 예방 노력에 참고할 수 있습니다. 인시던트 사후 분석에서 얻은 결과와 인사이트를 공유하고, 배운 교훈에 관해 논의하기 위해 누구나 참여할 수 있는 인시던트 사후 검토 모임을 개최합니다.

구현 단계

- 인시던트 사후 분석을 수행하는 과정에서 서로 비난하지 않도록 합니다. 이를 통해 인시던트에 관련된 사람들이 제안된 시정 조치에 대해 감정을 배제하고 팀 간의 솔직한 자체 평가와 협업을 촉진할 수 있습니다.
- 중요한 문제를 문서화하는 표준화된 방법을 정의합니다. 이러한 문서의 구조를 예로 들면 다음과 같습니다.
  - 어떻게 된 걸까요?
  - 문제가 고객과 비즈니스에 미친 영향
  - 근본 원인은 무엇인가요?
  - 문제 지원을 위한 데이터는 무엇인가요?
    - 예: 지표 및 그래프
  - 핵심 원칙(특히 보안)과 관려하여 어떤 의미가 있나요?
    - 워크로드를 설계할 때는 업무 상황에 따라 이러한 핵심 요소를 절충해야 합니다. 이러한 비즈니스 의사결정에 따라 엔지니어링 우선 순위가 달라질 수 있습니다. 예를 들어 개발 환경에서는 신뢰성이 다소 낮아지더라도 비용을 절감하도록 최적화할 수 있습니다. 또는 미션 크리티컬 솔루션의 경우에는 비용 증가를 감수하고 신뢰성을 기준으로 최적화할 수 있습니다. 하지만 고객 보호가 최우선이므로 모든 작업의 시작점은 항상 보안입니다.
  - 어떤 점을 배웠나요?
  - 어떤 시정 조치를 취했나요?
    - 작업 항목
    - 관련 항목
- 인시던트 사후 분석을 수행하기 위한 잘 정의된 표준 운영 절차를 만듭니다.

- 표준화된 인시던트 보고 프로세스를 설정합니다. 초기 인시던트 보고서, 로그, 통신 및 인시던트 중에 취한 조치를 포함하여 모든 인시던트를 포괄적으로 문서화합니다.
- 참고로 가동 중단이 일어나지 않더라도 인시던트일 수 있습니다. 중단이나 장애가 발생할 뻔한 상황, 시스템이 비즈니스 기능을 수행하면서도 예상치 못한 방식으로 작동하는 경우도 인시던트에 해당합니다.
- 피드백과 교훈을 바탕으로 인시던트 사후 분석 프로세스를 지속적으로 개선합니다.
- 지식 관리 시스템에서 주요 조사 결과를 캡처하고 개발자 가이드 또는 배포 전 체크리스트에 추가해야 할 패턴이 있는지 고려합니다.

리소스

관련 문서:

- Why you should develop a correction of error (COE)

관련 비디오:

- Amazon’s approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders’ Library: Operational Excellence at Amazon

REL12-BP03 확장성 및 성능 요구 사항 테스트

로드 테스트와 같은 기술을 사용하여 워크로드가 크기 조정 및 성능 요구 사항을 충족하는지 확인합니다.

클라우드에서는 워크로드에 대한 프로덕션 규모의 테스트 환경을 온디맨드로 생성할 수 있습니다. 프로덕션 동작의 부정확한 예측으로 이어질 수 있는 스케일 다운된 테스트 환경에 의존하는 대신 클라우드를 사용하여 예상 프로덕션 환경을 유사하게 반영하는 테스트 환경을 프로비저닝할 수 있습니다. 이 환경은 애플리케이션이 직면하는 실제 조건을 보다 정확하게 시뮬레이션하여 테스트하는 데 도움이 됩니다.

성능 테스트를 위한 노력과 더불어 기본 리소스, 크기 조정 설정, 서비스 할당량 및 복원력 설계가 로드 조건에서 예상대로 작동하는지 검증하는 것이 중요합니다. 이 전체론적 접근 방식을 통해 가장 까다로운 조건에서도 애플리케이션이 필요에 따라 안정적으로 크기가 조정되고 작동하는지 확인할 수 있습니다.

원하는 성과: 워크로드가 피크 로드에 노출되더라도 예상 동작을 유지합니다. 애플리케이션이 성장하고 발전함에 따라 발생할 수 있는 성능 관련 문제를 사전에 해결합니다.

일반적인 안티 패턴:

- 프로덕션 환경과 유사하지 않은 테스트 환경을 사용합니다.
- 로드 테스트를 배포 지속적 통합(CI) 파이프라인의 통합된 부분이 아닌 별도의 일회성 활동으로 취급합니다.
- 응답 시간, 처리량 및 확장성 목표와 같은 명확하고 측정 가능한 성능 요구 사항을 정의하지 않습니다.
- 비현실적이거나 불충분한 로드 시나리오로 테스트를 수행하며 피크 로드, 갑작스러운 스파이크 및 지속적으로 높은 로드를 테스트하지 못합니다.
- 예상 로드 한도를 초과하여 워크로드에 스트레스 테스트를 수행하지 않습니다.
- 불충분하거나 부적절한 로드 테스트 및 성능 프로파일링 도구를 사용합니다.
- 성능 지표를 추적하고 이상을 감지할 수 있는 포괄적인 모니터링 및 알림 시스템이 부족합니다.

이 모범 사례 확립의 이점:

- 로드 테스트를 통해 시스템이 프로덕션으로 전환되기 전에 시스템의 잠재적 성능 병목 현상을 식별할 수 있습니다. 프로덕션 수준 트래픽 및 워크로드를 시뮬레이션할 때 느린 응답 시간, 리소스 제약 또는 시스템 장애와 같이 시스템이 로드를 처리하는 데 어려움을 겪을 수 있는 영역을 식별할 수 있습니다.
- 다양한 로드 조건에서 시스템을 테스트할 때 워크로드를 지원하는 데 필요한 리소스 요구 사항을 더 잘 이해할 수 있습니다. 이 정보는 리소스 할당에 대해 정보에 입각한 결정을 내리고 리소스의 과다 프로비저닝 또는 과소 프로비저닝을 방지하는 데 도움이 될 수 있습니다.
- 잠재적 장애 지점을 식별하려면 로드가 높은 조건에서 워크로드가 어떻게 작동하는지 관찰하면 됩니다. 이 정보는 적절한 경우 내결함성 메커니즘, 장애 조치 전략 및 중복 조치를 구현하여 워크로드의 신뢰성과 복원력을 개선하는 데 도움이 됩니다.
- 성능 문제를 조기에 식별하고 해결하므로 시스템 중단, 느린 응답 시간 및 사용자 불만족으로 인한 비용이 많이 드는 결과를 방지할 수 있습니다.
- 테스트 중에 수집된 자세한 성능 데이터 및 프로파일링 정보는 프로덕션에서 발생할 수 있는 성능 관련 문제를 해결하는 데 도움이 될 수 있습니다. 이로 인해 인시던트 대응 및 해결 속도가 빨라져 사용자와 조직의 운영에 미치는 영향이 줄어들 수 있습니다.
- 특정 산업에서 사전 성능 테스트를 수행하면 워크로드가 규정 준수 표준을 충족하는 데 도움이 되므로 처벌 또는 법적 문제의 위험이 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

첫 번째 단계는 크기 조정 및 성능 요구 사항의 모든 측면을 아우르는 포괄적인 테스트 전략을 정의하는 것입니다. 시작하려면 처리량, 지연 시간 히스토그램, 오류율과 같은 비즈니스 요구 사항에 따라 워크로드의 서비스 수준 목표(SLO)를 명확하게 정의하세요. 다음으로, 평균 사용량부터 갑작스러운 스파이크 및 지속적인 피크 로드에 이르기까지 다양한 로드 시나리오를 시뮬레이션할 수 있는 테스트 세트를 설계하고 워크로드의 동작이 SLO를 충족하는지 확인합니다. 이러한 테스트는 개발 프로세스 초기에 성능 회귀를 포착하기 위해 자동화되어야 하고 지속적인 통합 및 배포 파이프라인에 통합되어야 합니다.

크기 조정 및 성능을 효과적으로 테스트하려면 적절한 도구와 인프라에 투자하세요. 여기에는 실제 사용자 트래픽을 생성할 수 있는 로드 테스트 도구, 병목 현상을 식별하는 성능 프로파일링 도구, 주요 지표를 추적하는 모니터링 솔루션이 포함됩니다. 중요한 점은 테스트 결과가 최대한 정확하도록 하려면 인프라 및 환경 조건 측면에서 테스트 환경이 프로덕션 환경과 최대한 일치하는지 확인해야 한다는 것입니다. 프로덕션과 유사한 설정을 안정적으로 복제하고 확장할 수 있도록 코드형 인프라와 컨테이너 기반 애플리케이션을 사용합니다.

크기 조정 및 성능 테스트는 일회성 활동이 아닌 지속적인 프로세스입니다. 포괄적인 모니터링 및 알림을 구현하여 애플리케이션의 프로덕션 성능을 추적하고 이 데이터를 사용하여 테스트 전략 및 최적화 노력을 지속적으로 개선합니다. 성능 데이터를 정기적으로 분석하여 새로운 문제를 식별하고, 새로운 크기 조정 전략을 테스트하고, 최적화를 구현하여 애플리케이션의 효율성과 신뢰성을 개선합니다. 반복적 접근 방식을 채택하고 프로덕션 데이터에서 지속적으로 교훈을 얻으면 애플리케이션이 다양한 사용자 요구 사항에 맞춰 조정하고 시간이 지남에 따라 복원력과 최적의 성능을 유지할 수 있는지 확인할 수 있습니다.

구현 단계

1. 응답 시간, 처리량 및 확장성 목표와 같은 명확하고 측정 가능한 성능 요구 사항을 설정합니다. 이러한 요구 사항은 워크로드의 사용 패턴, 사용자 기대치 및 비즈니스 요구 사항을 기반으로 해야 합니다.
2. 프로덕션 환경에서 로드 패턴 및 사용자 동작을 정확하게 모방할 수 있는 로드 테스트 도구를 선택하고 구성합니다.
3. 인프라 및 환경 조건을 포함하여 프로덕션 환경과 거의 일치하는 테스트 환경을 설정하여 테스트 결과의 정확도를 개선합니다.
4. 평균 사용 패턴부터 피크 로드, 빠른 스파이크, 지속적으로 높은 로드에 이르기까지 다양한 시나리오를 포함하는 테스트 세트를 생성합니다. 테스트를 지속적 통합 및 배포 파이프라인에 통합하여 개발 프로세스 초기에 성능 회귀를 파악합니다.

5. 로드 테스트를 수행하여 실제 사용자 트래픽을 시뮬레이션하고 애플리케이션이 다양한 로드 조건에서 어떻게 작동하는지 파악합니다. 애플리케이션에 스트레스 테스트를 수행하려면 예상 로드를 초과하고 응답 시간 저하, 리소스 소진 또는 시스템 장애와 같은 동작을 관찰하여 애플리케이션의 중단점을 식별하고 이를 크기 조정 전략에 반영합니다. 로드를 점진적으로 늘려 워크로드의 확장성을 평가하고 성능 영향을 측정하여 크기 조정 한도를 식별하고 향후 용량 요구 사항에 맞게 계획합니다.
6. 종합적인 모니터링 및 알림을 구현하여 성능 지표를 추적하고, 이상을 감지하고, 임계값이 초과되면 조정 작업 또는 알림을 시작합니다.
7. 성능 데이터를 지속적으로 모니터링하고 분석하여 개선이 필요한 영역을 식별합니다. 테스트 전략과 최적화 노력을 반복하여 개선합니다.

리소스

관련 모범 사례:

- REL01-BP04 할당량 모니터링 및 관리
- REL06-BP01 워크로드의 모든 구성 요소 모니터링(생성)
- REL06-BP03 알림 전송(실시간 처리 및 경보)

관련 문서:

- 부하 테스트 애플리케이션
- Distributed Load Testing on AWS
- 애플리케이션 성능 모니터링
- Amazon EC2 Testing Policy

관련 예제:

- Distributed Load Testing on AWS (GitHub)

관련 도구:

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter

- K6
- Vegeta
- Hey
- ab
- wrk
- Distributed Load Testing on AWS

REL12-BP04 카오스 엔지니어링을 이용한 복원력 테스트

시스템이 불리한 조건에서 어떻게 반응하는지 파악하려면 프로덕션 내 환경 또는 프로덕션과 최대한 가까운 환경에서 카오스 실험을 정기적으로 실행합니다.

원하는 성과:

워크로드의 복원력은 이벤트 중 워크로드의 알려진 예상 동작을 확인하는 복원력 테스트 이외에 결함 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: 중간

구현 지침

카오스 엔지니어링은 서비스 공급업체, 인프라, 워크로드 및 구성 요소 수준에서 고객에게 미치는 영향을 최소화하거나 아무런 영향을 미치지 않으면서 제어되는 방식으로 실제 중단(시뮬레이션)을 지속적

으로 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

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.

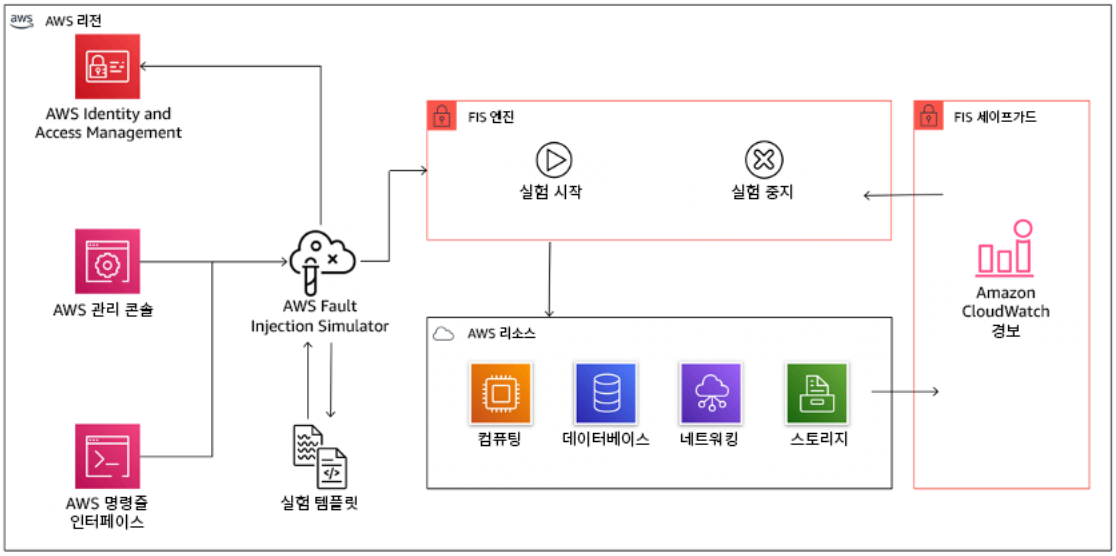
폴백 또는 장애 조치 메커니즘을 검증하려면 구성 요소가 몇 초에서 몇 시간까지 지속될 수 있는 지정된 기간에 서드파티 제공업체에 대한 액세스를 차단하여 해당 이벤트를 시뮬레이션해야 합니다.

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 신뢰할 수 없는 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)는 CD 파이프라인의 일부로 또는 파이프라인 외부에서 사용할 수 있는 결함 주입 실험을 실행하는 데 사용되는 완전관리형 서비스입니다. AWS FIS는 카오스 엔지니어링 게임 데이 중 사용하기 좋습니다. Amazon EC2, Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS), Amazon RDS를 비롯한 여러 유형의 리소

스에서 동시에 결함 주입 기능을 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.



워크로드에 결함 주입 실험을 실행할 수 있도록 AWS 리소스와 통합된 AWS Fault Injection Service.

결함 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 결함 범위를 확장하기 위해 AWS FIS는 Chaos Mesh 및 Litmus Chaos와 통합되어 여러 도구 간에 결함 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

1. 실험에 사용할 결함을 결정합니다.

   워크로드 설계의 복원력을 평가합니다. Well-Architected Framework의 모범 사례를 사용하여 생성된 이러한 설계는 중요한 종속성, 과거 이벤트, 알려진 문제 및 규정 준수 요구 사항을 기반으로 위험을 고려합니다. 복원력을 유지하기 위한 설계 요소와 완화하도록 설계된 장애를 나열합니다. 이러한 목록 작성에 대한 자세한 내용은 Operational Readiness Review 백서를 참조하세요. 여기에서는 이전 인시던트 재발을 방지하기 위한 프로세스 생성 방법을 안내합니다. 장애 모드 및 영향 분석(FMEA) 프로세스는 장애의 구성 요소 수준 및 장애가 워크로드에 미치는 영향을 분석하기 위한 프레임워크를 제공합니다. FMEA는 Adrian Cockcroft의 Failure Modes and Continuous Resilience에서 자세히 설명됩니다.

2. 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.

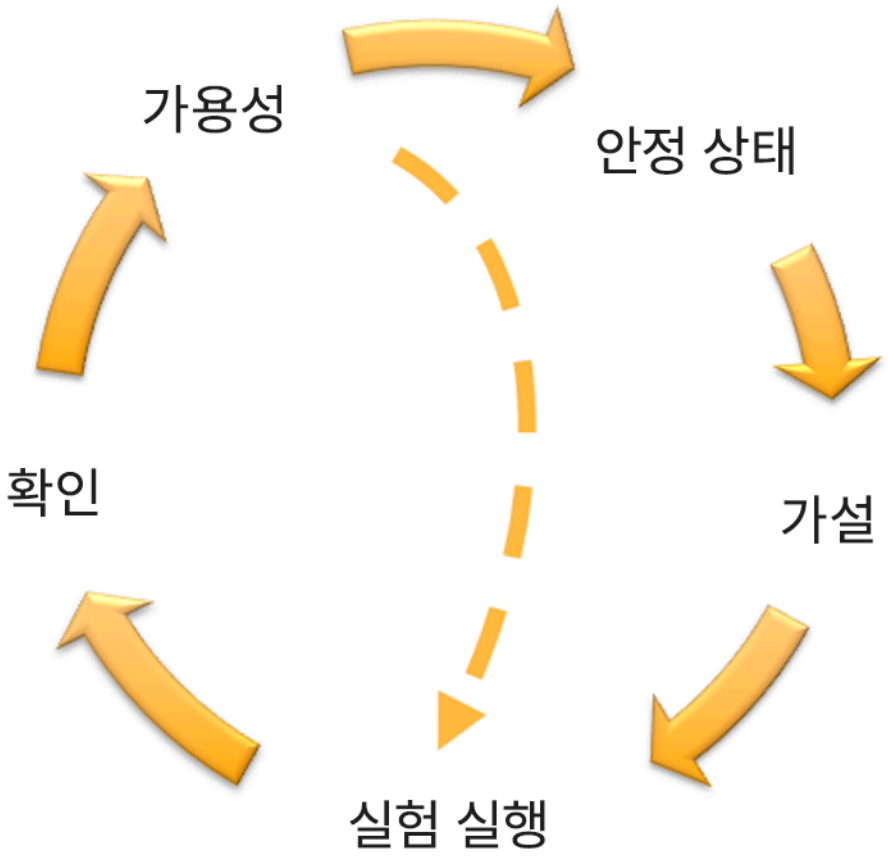
   주어진 장애의 빈도를 고려할 때 확인 가능한 경우 이 워크로드에 대한 이전 데이터를 분석합니다. 확인할 수 없는 경우에는 유사한 환경에서 실행되는 다른 워크로드의 데이터를 사용합니다.

   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 소스입니다.

   할당된 우선순위를 사용하여 어떤 결함을 먼저 실험할지, 새로운 결함 주입 실험을 개발할 순서를 결정합니다.

3. 수행하는 각 실험에 대해 카오스 애플리케이션 및 지속적인 복원력 플라이휠을 따릅니다(다음 그림 참조).



Adrian Hornsby의 과학적 방법을 사용하는 카오스 엔지니어링 및 연속 복원력 플라이휠.

a. 안정 상태를 정상 동작을 나타내는 워크로드의 측정 가능한 출력으로 정의합니다.

   예상한 대로 신뢰할 수 있는 상태로 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.

b. 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 또

는 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**
>
> ## ##가 발생하면 #### ## 워크로드가 ## ## ##를 기술하여 #### ## ## ## # #을 유지 관리합니다.

예시:

- 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 시스템의 Elastic Load Balancing 상태 확인은 Elastic Load Balancing에서 나머지 정상 인스턴스로만 요청을 보내는 반면, Amazon EC2 Auto Scaling은 장애가 발생한 인스턴스를 교체하여 서버 측(5xx) 오류의 증가율을 0.01% 미만으로 유지합니다(안정 상태).
- 기본 Amazon RDS 데이터베이스 인스턴스에서 장애가 발생하면 공급망 데이터 컬렉션 워크로드는 장애 조치를 취하고 대기 Amazon RDS 데이터베이스 인스턴스에 연결하여 데이터베이스 읽기 또는 쓰기 오류를 1분 미만으로 유지합니다(안정 상태).

c. 결함을 삽입하여 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-unknowns는 알고 있지만 완전히 파악하지 못한 항목이며, unknown-unknowns는 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. 결함 주입을 위한 옵션은 여러 가지가 있습니다. AWS의 워크로드에 대해 AWS FIS에서는 작업이라고 하는 미리 정의된 여러 장애 시뮬레이션을 제공합니다. 또한 AWS Systems Manager 문서를 사용하여 AWS FIS에서 실행하는 사용자 지정 작업을 정의할 수도 있습니다.

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 내보내며, 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 FIS를 사용한 다양한 실험에 대해 알아보려면 Resilient and Well-Architected Apps with Chaos Engineering lab도 참조하세요. AWS Resilience Hub에서는 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 배포를 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. 가상 모니터(사용자 canary라고도 함)는 일반적으로 사용자 프록시로 포함해야 하는 한 가지 지표입니다. AWS FIS의 중지 조건은 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.

결함 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). 사후 작업은 대상을 작업이 실행되기 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.

d. 가설을 확인합니다.

Principles of Chaos Engineering에서는 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.

e. 복원력을 위해 워크로드 설계를 개선합니다.

안정 상태가 유지되지 않는 경우 장애를 완화하기 위해 워크로드 설계를 어떻게 개선할 수 있는지 조사하여 AWS Well-Architected 신뢰성 원칙의 모범 사례를 적용합니다. 추가 지침 및 리소스는 AWS Builder's Library에서 찾을 수 있습니다. 여기에서는 특히, 상태 확인 개선 방법 또는 애플리케이션 코드에서 백오프를 사용하여 재시도 수행 방법에 대한 문서를 제공합니다.

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.

4. 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. 이를 수행하는 방법을 알아보려면 how to run AWS FIS experiments using AWS CodePipeline에서 이 블로그를 참조하세요. 반복된 AWS FIS 실험(CI/CD 파이프라인 내)에 관한 이 실습에서 실제로 수행해볼 수 있습니다.

   결함 주입 실험은 게임 데이의 일부이기도 합니다(REL12-BP05 정기적으로 게임 데이 진행 참조). 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.

5. 실험 결과를 캡처하고 저장합니다.

   결함 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 FIS에서 실험 로깅은 이 데이터 캡처에 포함될 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 무엇입니까?
- AWS Resilience Hub란 무엇입니까?

- Principles of Chaos Engineering
- Chaos Engineering: Planning your first experiment
- Resilience Engineering: Learning to Embrace Failure
- Chaos Engineering stories
- 분산 시스템의 폴백 방지
- Canary Deployment for Chaos Experiments

관련 비디오:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

관련 도구:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP05 정기적으로 게임 데이 진행

게임 데이를 수행하여 워크로드에 영향을 미치는 이벤트 및 장애에 대응하는 절차를 정기적으로 연습합니다. 프로덕션 시나리오를 처리할 책임이 있는 동일한 팀을 참여시킵니다. 이러한 연습은 프로덕션 이벤트로 인한 사용자 영향을 방지하기 위한 조치를 시행하는 데 도움이 됩니다. 현실적인 조건에서 대응 절차를 연습할 때 실제 이벤트가 발생하기 전에 격차나 약점을 식별하고 해결할 수 있습니다.

게임 데이는 프로덕션과 유사한 환경에서 이벤트를 시뮬레이션하여 시스템, 프로세스 및 팀 응답을 테스트합니다. 게임 데이의 목적은 이벤트가 실제로 발생할 때 팀이 수행해야 하는 것과 동일한 작업을 수행해보는 것입니다. 이 연습은 어느 분야에서 개선이 필요한지 파악하고, 조직이 이벤트 및 장애에 대처하는 경험을 쌓는 데 도움이 될 수 있습니다. 팀이 대응 방법을 습관으로 체득할 수 있도록 게임 데이를 정기적으로 진행해야 합니다.

게임 데이는 팀이 더 큰 확신을 갖고 프로덕션 이벤트를 처리할 수 있도록 준비합니다. 연습이 잘 된 팀은 다양한 시나리오를 더 빠르게 탐지하고 대응할 수 있습니다. 이로 인해 준비도와 복원력이 크게 향상됩니다.

원하는 성과: 일정에 따라 일관되게 복원력 게임 데이를 실행합니다. 이러한 게임 데이는 비즈니스 수행의 정상적이고 예상되는 부분으로 간주됩니다. 조직은 준비 문화를 구축했으며, 프로덕션 문제가 발생할 때 팀은 효과적으로 대응하고, 문제를 효율적으로 해결하고, 고객에게 미치는 영향을 완화할 준비가 잘 되어 있습니다.

일반적인 안티 패턴:

- 절차를 문서화하지만 결코 연습하지 않습니다.
- 테스트 연습에서는 비즈니스 의사 결정권자를 제외합니다.
- 게임 데이를 실행하지만 모든 관련 이해관계자에게 알리지는 않습니다.
- 기술 장애에만 집중하지만 비즈니스 이해관계자는 참여시키지 않습니다.
- 게임 데이에서 얻은 교훈을 복구 프로세스에 반영하지 않습니다.
- 장애 또는 버그에 대해 팀을 비난합니다.

이 모범 사례 확립의 이점:

- 대응 스킬 향상: 게임 데이에 팀은 시뮬레이션된 이벤트 중에 임무를 연습하고 커뮤니케이션 메커니즘을 테스트하여 프로덕션 상황에서 보다 조율되고 효율적인 대응 조치를 만듭니다.
- 종속성 식별 및 해결: 복잡한 환경에는 다양한 시스템, 서비스 및 구성 요소 간의 복잡한 종속성이 수반되는 경우가 많습니다. 게임 데이는 이러한 종속성을 식별하고 해결하는 데 도움이 될 수 있으며, 중요한 시스템과 서비스가 런북 절차에서 적절하게 다루어지고 적시에 스케일 업하거나 복구할 수 있는지 확인할 수 있습니다.
- 복원력 문화 조성: 게임 데이는 조직 내에서 복원력에 대한 사고방식을 키우는 데 도움이 될 수 있습니다. 여러 부서의 팀과 이해관계자를 참여시킬 때 이러한 연습은 조직 전체에서 복원력의 중요성에 대한 인식을 높이고 협업과 공통의 이해를 촉진합니다.
- 지속적인 개선 및 조정: 정기적인 게임 데이는 복원력 전략을 지속적으로 평가하고 조정하는 데 도움이 되므로 변화하는 상황에 대비하여 관련성과 효율성을 유지할 수 있습니다.
- 시스템에 대한 신뢰도 향상: 성공적인 게임 데이를 통해 시스템 중단 상황을 견디고 복구하는 능력에 대한 신뢰도를 높일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

필요한 복원력 조치를 설계하고 구현한 후에는 게임 데이를 수행하여 모든 것이 프로덕션에서 계획대로 작동하는지 확인합니다. 특히 게임 데이의 첫날에는 모든 팀원이 참여해야 하며 모든 이해관계자와 참가자에게 날짜, 시간 및 시뮬레이션된 시나리오에 대해 미리 알려야 합니다.

게임 데이를 진행하는 동안 관련 팀이 규정된 절차에 따라 다양한 이벤트와 잠재적 시나리오를 시뮬레이션합니다. 참가자는 이러한 시뮬레이션된 이벤트의 영향을 면밀히 모니터링하고 평가합니다. 시스템이 설계된 대로 작동하는 경우 자동 탐지, 크기 조정 및 자체 복구 메커니즘이 활성화되어 사용자에게 거의 또는 전혀 영향을 미치지 않습니다. 팀이 부정적인 영향을 발견하면 테스트를 롤백하고 해당 런북에 문서화된 자동화된 수단 또는 수동 개입을 통해 식별된 문제를 해결합니다.

복원력을 지속적으로 개선하려면 얻은 교훈을 문서화하고 반영하는 것이 중요합니다. 이 프로세스는 게임 데이의 인사이트를 체계적으로 캡처하고 이를 사용하여 시스템, 프로세스 및 팀 기능을 개선하는 피드백 루프입니다.

시스템 구성 요소 또는 서비스가 예기치 않게 실패할 수 있는 실제 시나리오를 재현하는 데 도움이 되도록 게임 데이 연습으로 시뮬레이션된 장애를 주입합니다. 팀은 시스템의 복원력과 내결함성을 테스트하고 통제된 환경에서 인시던트 대응 및 복구 프로세스를 시뮬레이션할 수 있습니다.

AWS에서는 코드형 인프라를 사용하여 프로덕션 환경의 복제본으로 게임 데이를 수행할 수 있습니다. 이 프로세스를 통해 프로덕션 환경과 매우 유사한 안전한 환경에서 테스트할 수 있습니다. 다양한 장애 시나리오를 생성하기 위해 AWS Fault Injection Service를 고려해 보세요. Amazon CloudWatch 및 AWS X-Ray와 같은 서비스를 사용하여 게임 데이 기간 동안 시스템 동작을 모니터링합니다. AWS Systems Manager를 사용하여 플레이북을 관리 및 실행하고 AWS Step Functions을 사용하여 반복되는 게임 데이 워크플로를 오케스트레이션합니다.

구현 단계

- 게임 데이 프로그램 설정: 게임 데이의 빈도, 범위 및 목표를 정의하는 구조화된 프로그램을 개발합니다. 이러한 연습을 계획하고 실행하는 데 주요 이해관계자와 주제 전문가를 참여시킵니다.
- 게임 데이 준비:
  1. 게임 데이에서 중점을 둘 핵심 비즈니스 크리티컬 서비스를 결정합니다. 이러한 서비스를 지원하는 사람, 프로세스 및 기술을 카탈로그화하고 매핑합니다.
  2. 게임 데이의 어젠다를 설정하고 관련 팀이 이벤트에 참여하도록 준비시킵니다. 계획된 시나리오를 시뮬레이션하고 적절한 복구 프로세스를 실행하도록 자동화 서비스를 준비합니다. AWS Fault Injection Service, AWS Step Functions 및 AWS Systems Manager와 같은 AWS 서비스는 장애 주입 및 복구 작업 시작과 같은 게임 데이의 다양한 측면을 자동화하는 데 도움이 될 수 있습니다.

- 시뮬레이션 실행: 게임 데이 당일에 계획된 시나리오를 실행합니다. 사람, 프로세스 및 기술이 시뮬레이션된 이벤트에 어떻게 반응하는지 관찰하고 문서화합니다.
- 연습 후 검토 수행: 게임 데이가 끝나면 되돌아보는 세션을 갖고 얻은 교훈을 검토합니다. 개선이 필요한 영역과 운영 복원력을 개선하는 데 필요한 모든 조치를 식별합니다. 조사 결과를 문서화하고 필요한 변경 사항을 추적하여 복원력 전략과 완료 준비도를 강화합니다.

리소스

관련 모범 사례:

- REL12-BP01 플레이북을 사용하여 장애 조사
- REL12-BP04 카오스 엔지니어링을 이용한 복원력 테스트
- OPS04-BP01 핵심 성과 지표 파악
- OPS07-BP03 런북을 사용한 절차 수행
- OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용

관련 문서:

- AWS GameDay란?
- AWS Well-Architected Concepts - Game Day

관련 비디오:

- AWS re:Invent 2,023 - Practice like you play: How Amazon scales resilience to new heights

관련 예제:

- AWS Workshop - Navigate the storm: Unleashing controlled chaos for resilient systems
- Build Your Own Game Day to Support Operational Resilience

REL 13. 재해 복구(DR)를 어떻게 계획하나요?

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 비

용도 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
- REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용
- REL13-BP03 재해 복구 구현을 테스트하여 구현 확인
- REL13-BP04 DR 사이트 또는 리전에서 구성 드리프트 관리
- REL13-BP05 자동 복구

REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의

장애가 발생하면 여러 가지 방법으로 비즈니스에 영향을 미칠 수 있습니다. 첫째, 장애는 서비스 중단(작동 중지 시간)을 유발할 수 있습니다. 둘째, 장애는 데이터 손실, 비일관성, 기한 경과를 유발할 수 있습니다. 장애에 대응하고 복구하는 방법을 안내하려면 각 워크로드에 대해 목표 복구 시간(RTO) 및 목표 복구 시점(RPO)을 정의하세요. Recovery Time Objective(RTO)는 서비스 중단과 서비스 복원 사이의 허용 가능한 최대 지연 시간입니다. 목표 복구 시점(RPO)은 마지막 데이터 복구 시점 후 허용되는 최대 시간입니다.

원하는 성과: 모든 워크로드에 기술적 고려 사항 및 비즈니스 영향을 기반으로 지정된 RTO 및 RPO가 있습니다.

일반적인 안티 패턴:

- 복구 목표를 지정하지 않았습니다.
- 임의의 복구 목표를 선택합니다.
- 너무 관대하고 비즈니스 목표를 충족하지 못하는 복구 목표를 선택합니다.
- 가동 중지 시간 및 데이터 손실의 영향을 평가하지 않았습니다.
- 워크로드 구성에서 달성할 수 없는 즉각 복구 또는 데이터 무손실과 같이 비현실적인 복구 목표를 선택합니다.
- 실제 비즈니스 목표보다 더 엄격한 복구 목표를 선택합니다. 이로 인해 워크로드에 필요한 수준 이상으로 복구 구현의 비용이 높아지고 복구 구현이 복잡해집니다.
- 종속된 워크로드의 복구 목표와 호환되지 않는 복구 목표를 선택합니다.
- 규제 및 규정 준수 요구 사항을 고려하지 않습니다.

이 모범 사례 확립의 이점: 워크로드에 대한 RTO 및 RPO를 설정할 때 비즈니스 요구 사항에 따라 명확하고 측정 가능한 복구 목표를 설정합니다. 이러한 목표를 설정한 후에는 목표에 맞게 조정된 재해 복구(DR) 계획을 수립할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

재해 복구 계획을 수립하는 데 도움이 되는 매트릭스 또는 워크시트를 구성합니다. 매트릭스에서 비즈니스 영향(예: 중요, 높음, 중간, 낮음)과 각각에 대해 목표로 삼을 관련 RTO 및 RPO를 기반으로 다양한 워크로드 범주 또는 계층을 생성합니다. 다음 매트릭스를 예시로 따라 만들 수 있습니다(RTO 값과 RPO 값이 실제와 다를 수 있음).

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

재해 복구 매트릭스 예

각 워크로드에서 가동 중단 시간 및 데이터 손실이 비즈니스에 미치는 영향을 조사하고 이해합니다. 영향은 일반적으로 가동 중지 시간 및 데이터 손실에 따라 증가하지만, 영향의 형태는 워크로드 유형에 따라 다를 수 있습니다. 예를 들어 최대 1시간의 가동 중지는 영향이 낮을 수 있지만, 그 후에는 영향이 빠르게 심해질 수 있습니다. 영향은 재정적 영향(예: 수익 손실), 평판 영향(고객 신뢰 상실 포함), 운영 영향(예: 급여 누락 또는 생산성 감소), 규제 위험을 포함한 다양한 형태로 나타날 수 있습니다. 완료되면 워크로드를 적절한 계층에 할당합니다.

장애의 영향을 분석할 때 다음 질문을 고려하세요.

1. 허용할 수 없는 영향을 비즈니스에 미치기 전에 워크로드가 사용 불가능해도 되는 시간은 최대 어느 정도인가요?
2. 워크로드 중단으로 인해 비즈니스에 어떤 종류의 영향이 얼마나 많이 나타나나요? 재무, 평판, 운영 및 규제를 포함한 모든 종류의 영향을 고려합니다.

3. 허용할 수 없는 영향을 비즈니스에 미치기 전에 손실되어도 되거나 복구할 수 없어도 되는 데이터의 양은 최대 어느 정도인가요?
4. 손실된 데이터를 다른 소스에서 다시 생성할 수 있나요(파생 데이터라고도 함)? 그렇다면 워크로드 데이터를 다시 생성하는 데 사용되는 모든 소스 데이터의 RPO도 고려해 보세요.
5. 이 워크로드가 의존하는 워크로드(다운스트림)의 복구 목표 및 가용성 기대치는 어느 정도인가요? 다운스트림 종속성의 복구 기능을 고려할 때 워크로드의 목표를 달성할 수 있어야 합니다. 이 워크로드의 복구 기능을 개선할 수 있는 가능한 다운스트림 종속성 해결 방법 또는 완화 방법을 고려합니다.
6. 이 워크로드에 의존하는 워크로드(업스트림)의 복구 목표 및 가용성 기대치는 어느 정도인가요? 업스트림 워크로드 목표를 사용하려면 이 워크로드가 처음 보기보다 더 엄격한 복구 기능을 갖추어야 할 수 있습니다.
7. 인시던트 유형에 따라 복구 목표가 다른가요? 예를 들어 인시던트가 하나의 가용 영역에 영향을 미치는지, 아니면 전체 리전에 영향을 미치는지에 따라 RTO와 RPO가 다를 수 있습니다.
8. 복구 목표가 특정 이벤트 또는 연중 특정 시간에 변경되나요? 예를 들어 연말 쇼핑 시즌, 스포츠 이벤트, 특별 세일 및 신제품 출시 시기에는 각기 다른 RTO와 RPO가 있을 수 있습니다.
9. 사업부 및 조직의 재해 복구 전략이 있다면 복구 목표가 그러한 전략에 어떻게 부합하나요?
10. 고려해야 할 법적 또는 계약상의 영향이 있나요? 예를 들어 계약상 지정된 RTO 또는 RPO에 따라 서비스를 제공할 의무가 있나요? 이를 충족하지 못하면 어떤 처벌을 받을 수 있나요?
11. 규제 또는 규정 준수 요구 사항을 충족하기 위해 데이터 무결성을 유지해야 하나요?

다음 워크시트는 각 워크로드를 평가하는 데 도움이 될 수 있습니다. 질문을 더 추가하는 등 특정 요구 사항에 맞게 이 워크시트를 수정할 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 단계

1. 각 워크로드를 담당하는 비즈니스 이해관계자와 기술 팀을 식별하고 참여시킵니다.
2. 워크로드가 조직에 미치는 영향에 관한 중요도를 나타내는 범주 또는 계층을 생성합니다. 범주의 예로는 치명적, 높음, 중간, 낮음이 있습니다. 각 범주에서 비즈니스 목표와 요구 사항을 반영하는 RTO 및 RPO를 선택합니다.
3. 이전 단계에서 생성한 영향 범주 중 하나를 각 워크로드에 할당합니다. 워크로드가 범주에 매핑되는 방법을 결정하려면 비즈니스에 대한 워크로드의 중요성과 중단 또는 데이터 손실의 영향을 고려하고 위의 질문을 활용하세요. 그러면 각 워크로드에 대한 RTO 및 RPO가 도출됩니다.
4. 이전 단계에서 결정된 각 워크로드에 대한 RTO 및 RPO를 고려합니다. 워크로드의 비즈니스 및 기술 팀을 참여시켜 목표를 조정해야 하는지 결정합니다. 예를 들어 비즈니스 이해관계자는 더 엄격한 목표가 필요하다고 판단할 수 있습니다. 반면 기술 팀은 가용 리소스와 기술적 제약을 기준으로 목표를 달성할 수 있도록 수정해야 한다고 판단할 수 있습니다.

리소스

관련 모범 사례:

- REL09-BP04 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행
- REL12-BP01 플레이북을 사용하여 장애 조사
- REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용
- REL13-BP03 재해 복구 구현을 테스트하여 구현 확인

관련 문서:

- AWS Architecture Blog: Disaster Recovery Series
- AWS에서 워크로드 재해 복구: 클라우드에서의 복구(AWS 백서)
- Managing resiliency policies with AWS Resilience Hub
- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품

관련 비디오:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- Disaster Recovery of Workloads on AWS

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 성과: 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음. DR 전략을 계획, 구현, 테스트하지 않으면 재해 발생 시 복구 목표를 달성하지 못할 가능성이 큽니다.

구현 지침

DR 전략은 기본 위치에서 워크로드를 실행할 수 없게 되었을 때 복구 사이트에서 워크로드를 실행하는 능력에 달려 있습니다. 가장 흔한 복구 목표는 REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의에서 논의한 RTO 및 RPO입니다.

하나의 AWS 리전 내에서 여러 가용 영역에 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 설계할 때 다음 전략 중 하나를 사용해야 합니다. 전략은 복잡성과 비용이 증가하고 RTO와 RPO가 감소하는 순서로 나열되어 있습니다. 복구 리전은 워크로드에 사용되는 기본 리전이 아닌 AWS 리전을 말합니다.

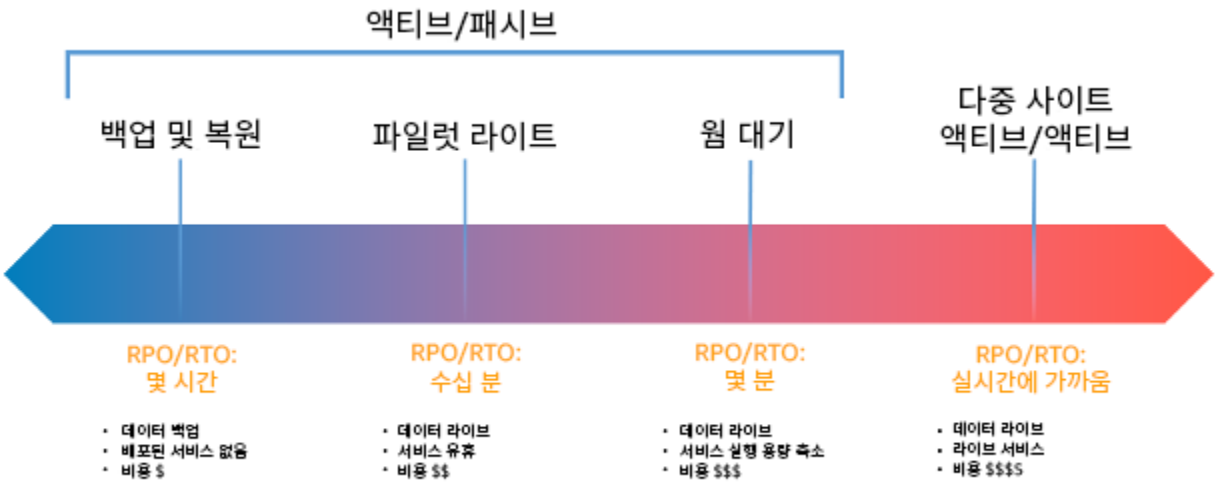


그림 17: 재해 복구(DR) 전략

- 백업 및 복원(시간 단위 RPO, 24시간 이하의 RTO): 데이터와 애플리케이션을 복구 리전에 백업합니다. 자동화된 백업 또는 지속적인 백업을 사용하면 시점 복구(PITR)가 가능하여 경우에 따라서는 RPO를 5분까지 줄일 수 있습니다. 재해 이벤트 시 인프라를 배포하고(RTO를 단축하기 위해 코드형 인프라 사용), 코드를 배포하며, 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): 항상 복구 리전에서 실행되는 모든 기능을 갖춘 워크로드의 스케일 다운된 버전을 유지합니다. 비즈니스 크리티컬 시스템은 완전히 복제되고 항상 실행되지만 플릿은 축소됩니다. 데이터가 복구 리전에 복제되며 실행됩니다. 복구 시기가 되면 시스템은 프로덕션 로드를 처리하기 위해 신속하게 스케일 업됩니다. 웜 대기 방식이 스케일 업될수록 RTO 및 컨트롤 플레인 의존도는 낮아집니다. 완전히 확장된 경우 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 선택합니다.
>
> 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.

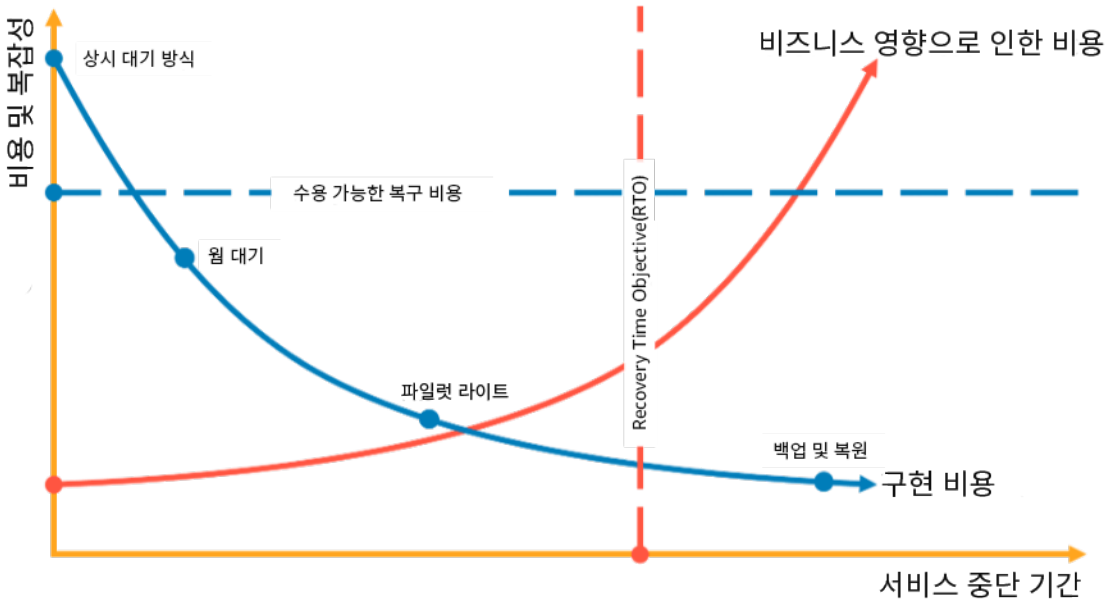


   그림 18: RTO 및 비용에 따른 DR 전략 선택

   자세한 내용은 비즈니스 연속성 계획(BCP)을 참조하세요.

2. 선택한 DR 전략을 구현할 수 있는 방법에 대한 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 복원

백업 및 복원은 구현하기 가장 복잡한 전략이지만 워크로드를 복원하는 데 더 많은 시간과 노력이 필요하므로 RTO 및 RPO도가 높아집니다. 항상 데이터의 백업을 만들어 다른 사이트(예: 다른 AWS 리전)에 복사해 두는 것이 좋습니다.

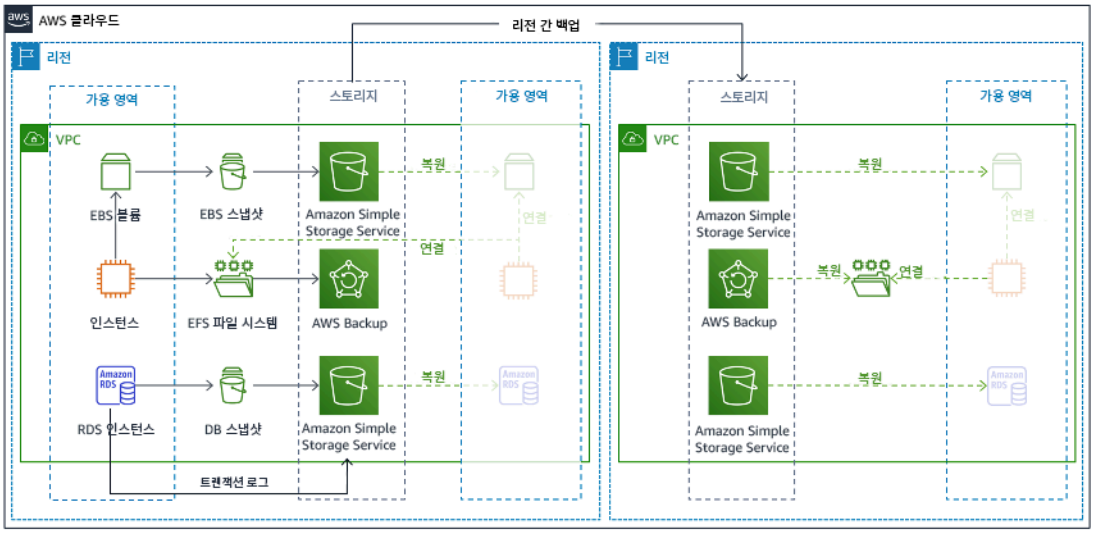


그림 19: 백업 및 복원 아키텍처

이 전략에 대한 자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery를 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.

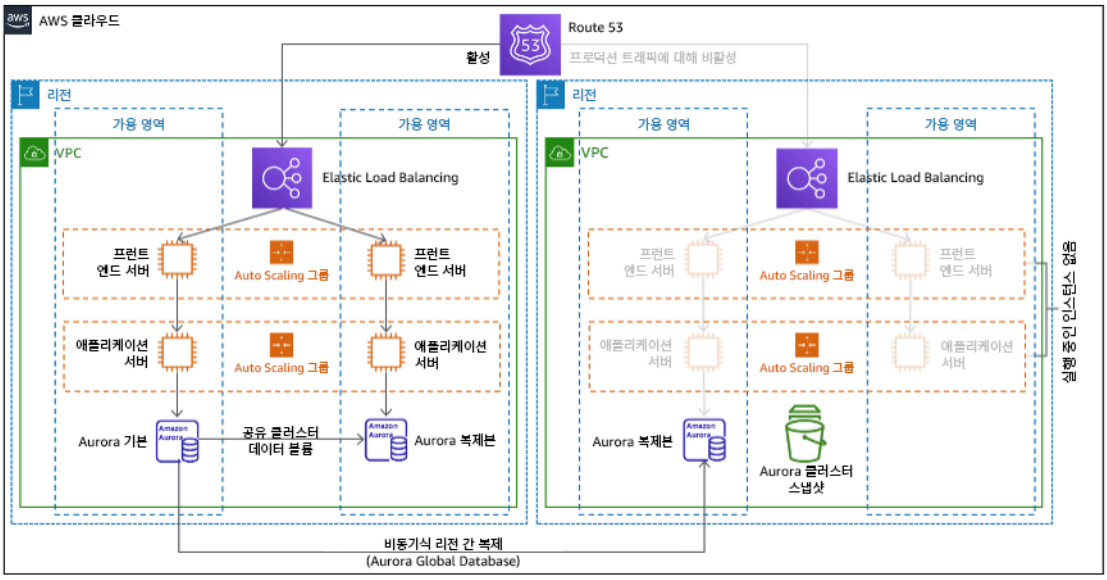


그림 20: 파일럿 라이트 아키텍처

이 전략에 대한 자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby를 참조하세요.

웜 대기

웜 대기 접근 방식에는 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 상시 대기 방식이라고 합니다.

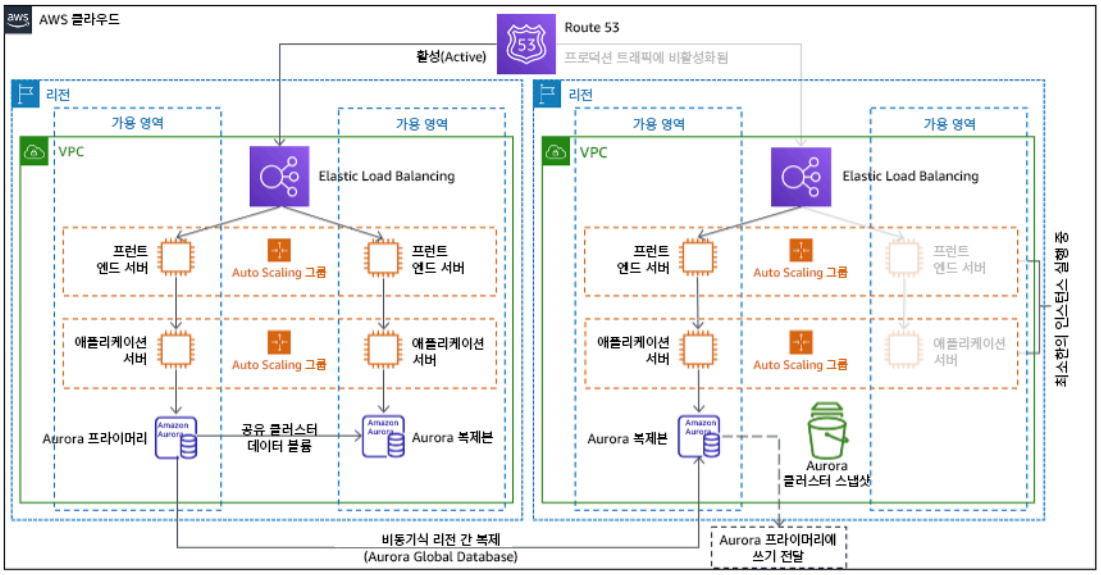


그림 21: 웜 대기 방식의 아키텍처

웜 대기 방식 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 간접 호출에 즉시 응답할 준비가 되도록 실행 환경을 제공할 수 있습니다.

이 전략에 대한 자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby를 참조하세요.

다중 사이트 액티브/액티브

다중 사이트 액티브/액티브 전략의 일환으로 여러 리전에서 동시에 워크로드를 실행할 수 있습니다. 다중 사이트 액티브/액티브는 배포된 모든 리전에서 트래픽을 처리합니다. 고객은 DR 이외의 이유로 이 전략을 선택할 수도 있습니다. 이 전략은 가용성을 높이기 위해서 또는 글로벌 사용자에게 워크로드를 배포하는 경우(사용자에게 엔드포인트를 가까이 가져가거나 해당 리전의 사용자에게 현지화된 스택을 배포하기 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.

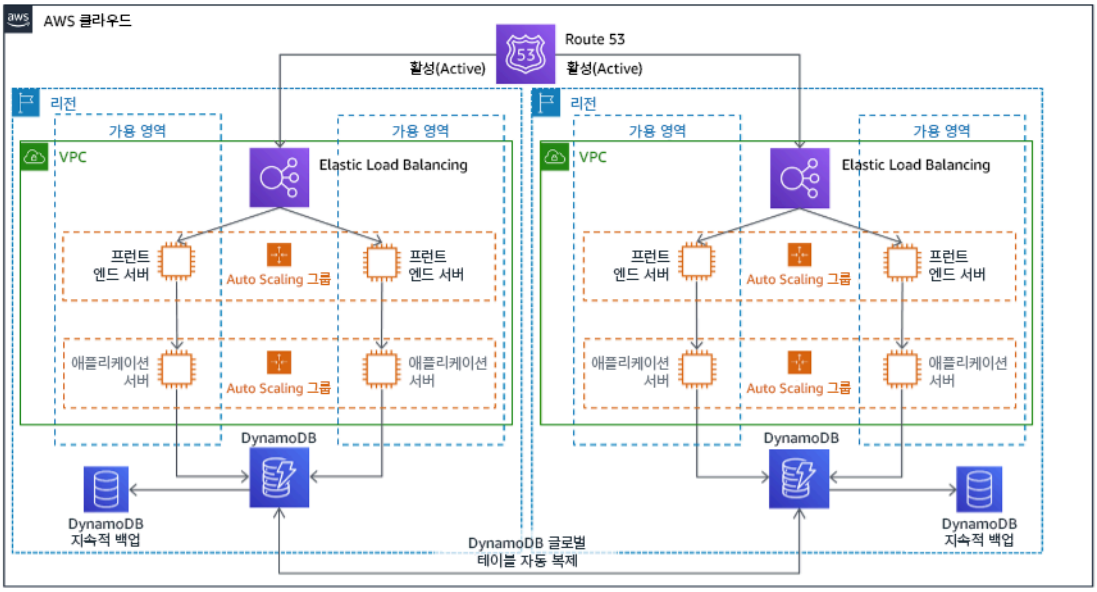


그림 22: 다중 사이트 액티브/액티브 아키텍처

이 전략에 대한 자세한 내용은 Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active를 참조하세요.

AWS Elastic Disaster Recovery

재해 복구를 위한 파일럿 라이트 또는 웜 대기 방식 전략을 고려 중인 경우 AWS Elastic Disaster Recovery에서는 향상된 이점을 제공하는 대안 접근 방식을 제공할 수 있습니다. Elastic Disaster Recovery에서는 웜 대기 방식과 유사한 RPO 및 RTO 목표를 제공하면서도 파일럿 라이트의 비용이 저렴한 접근 방식을 유지할 수 있습니다. Elastic Disaster Recovery는 지속적인 데이터 보호를 통해 기본 리전에서 복구 리전으로 데이터를 복제하여 초 단위의 RPO와 분 단위로 측정 가능한 RTO를 달성할 수 있습니다. 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.

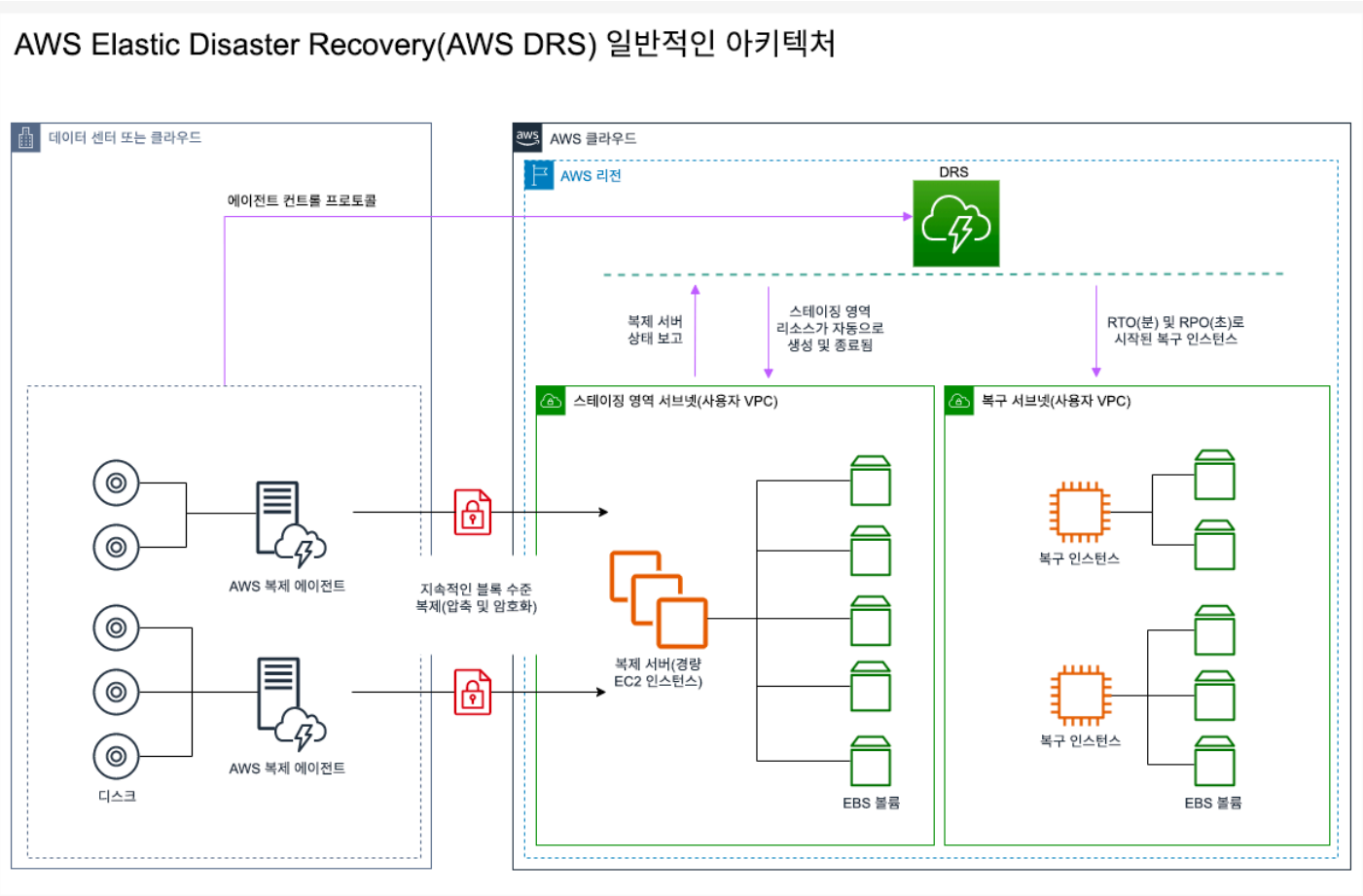


그림 23: AWS Elastic Disaster Recovery 아키텍처

데이터 보호를 위한 추가 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.

단일 AWS 리전에서 단일 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 액티브/액티브 접근 방식을 사용합니다. 또한 이 아키텍처는 기본 Amazon RDS 인스턴스에 장애가 발생하면(또는 AZ 자체에 장애 발생) 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. Amazon EBS 볼륨 또는 Amazon Redshift 클러스터와 같이 단일 영역으로 제한되는 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 Application Recovery Controller, Part 1: Single-Region stack에 나와 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. 워크로드 리소스를 평가하고, 장애 조치(정상 운영 중) 전에 복구 리전에 존재하는 해당 리소스의 구성을 평가합니다.

   인프라 및 AWS 리소스의 경우 코드형 인프라(예: AWS CloudFormation 또는 Hashicorp Terraform과 같은 서드파티 도구)를 사용합니다. 단일 작업으로 여러 계정 및 리전에 배포하기 위해 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 방식 전략의 경우 배포된 인프라가 프로덕션으로 사용되려면 추가 조치가 필요합니다. CloudFormation 파라미터 및 조건부 로직을 사용하면 단일 템플릿으로 배포된 스택이 활성 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 방식 및 다중 사이트 액티브/액티브의 경우 기본 리전의 데이터를 Amazon Relational Database Service(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 서비스 운영 방식에 대해 자세히 알아보려면 Creating a Multi-Region Application with AWS Services의 이 블로그 시리즈를 참조하세요.

4. 필요한 경우 재해 이벤트 중 장애 조치를 위해 복구 리전을 구성하는 방법을 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 방식 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. 자세한 내용은 REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제 섹션을 참조하세요. 인프라 재구축에는 필요한 Amazon Virtual Private Cloud(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. 필요한 경우 재해 이벤트 중 장애 조치를 위해 트래픽을 다시 라우팅하는 방법을 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. 한 가지 옵션은 Amazon Route 53을 사용하는 것입니다. Amazon Route 53을 사용하면 하나 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 플레인 API를 제공하는 Amazon Application Recovery Controller를 사용할 수 있습니다. 장애 조치를 구현할 때 데이터 플레인 작업을 사용하고 컨트롤 플레인 작업을 피합니다(REL11-BP04 복구 중 컨트롤 플레인이 아닌 데이터 플레인 사용 참조).

이 옵션 및 기타 옵션에 대해 자세히 알아보려면 재해 복구 백서의 이 섹션을 참조하세요.

6. 워크로드 페일백 방식에 대한 계획을 설계합니다.

   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 사용하는 경우, 기본 리전의 테이블을 사용할 수 없게 되었더라도 다시 온라인 상태가 되면 DynamoDB는 보류 중인 모든 쓰기의 전파를 재개합니다. Amazon Aurora Global Database를 사용하고 관리형 계획된 장애 조치 기능을 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.

   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 구성원이 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 Performing a failback을 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 플레인 사용”
- the section called “REL13-BP01 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의”

관련 문서:

- AWS Architecture Blog: Disaster Recovery Series
- AWS에서 워크로드 재해 복구: 클라우드에서의 복구(AWS 백서)
- 클라우드의 재해 복구 옵션
- Build a serverless multi-region, active-active backend solution in an hour
- Multi-region serverless backend - reloaded
- RDS: 리전 간 읽기 전용 복제본 복제
- Route 53: Configuring DNS Failover
- S3: 크로스 리전 복제
- AWS Backup란 무엇입니까?
- Amazon Application Recovery Controller란 무엇입니까?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS
- APN 파트너: 재해 복구를 지원할 수 있는 파트너
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품

관련 비디오:

- Disaster Recovery of Workloads on AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

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 않습니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 있습니다. 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 설

계 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 Recovery에서는 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 Architecture Blog: Disaster Recovery Series
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품
- AWS Elastic Disaster Recovery
- AWS에서 워크로드 재해 복구: 클라우드에서의 복구(AWS 백서)
- AWS Elastic Disaster Recovery Preparing for Failover
- The Berkeley/Stanford recovery-oriented computing project
- What is AWS Fault Injection Simulator?

관련 비디오:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

REL13-BP04 DR 사이트 또는 리전에서 구성 드리프트 관리

성공적인 재해 복구(DR) 절차를 수행하려면 DR 환경이 온라인 상태가 되면 관련 기능 또는 데이터의 손실 없이 워크로드가 적시에 정상 작업을 재개할 수 있어야 합니다. 이 목표를 달성하려면 DR 환경과 기본 환경 간에 일관된 인프라, 데이터 및 구성을 유지해야 합니다.

원하는 성과: 재해 복구 사이트의 구성 및 데이터가 기본 사이트와 동등하여 필요할 때 빠르고 완전하게 복구할 수 있습니다.

일반적인 안티 패턴:

- 기본 위치를 변경할 때 복구 위치를 업데이트하지 못하여 오래된 구성으로 인해 복구 작업이 지연됩니다.
- 기본 위치와 복구 위치 간의 서비스 차이와 같은 잠재적 제한 사항을 고려하지 않아 장애 조치 중에 예상치 못한 장애가 발생할 수 있습니다.
- 수동 프로세스를 사용하여 DR 환경을 업데이트하고 동기화하므로 인적 오류와 불일치의 위험이 증가합니다.
- 구성 드리프트를 감지하지 못하여 인시던트 발생 전에 DR 사이트 준비 상태를 잘못 인식합니다.

이 모범 사례 확립의 이점: DR 환경과 기본 환경 간의 일관성은 인시던트 후 성공적인 복구 가능성을 크게 개선하고 복구 절차 실패 위험을 줄입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

구성 관리 및 장애 조치 준비에 대한 포괄적인 접근 방식을 통해 DR 사이트가 지속적으로 업데이트되고 기본 사이트 장애 발생 시 인계받을 준비가 되었는지 확인할 수 있습니다.

기본 환경과 재해 복구(DR) 환경 간의 일관성을 얻으려면 전송 파이프라인이 기본 사이트와 DR 사이트 모두에 애플리케이션을 배포하는지 검증합니다. 적절한 평가 기간(시차 배포라고도 함) 후 DR 사이트에 대한 변경 사항을 롤아웃하여 기본 사이트의 문제를 감지하고 문제가 퍼지기 전에 배포를 중지합니다. 모니터링을 구현하여 구성 드리프트를 감지하고 환경 전반의 변경 사항 및 규정 준수를 추적합니다. DR 사이트에서 자동 수정을 수행하여 완전한 일관성을 유지하고 인시던트 발생 시 즉시 인계할 수 있도록 합니다.

구현 단계

1. DR 리전에 DR 계획을 성공적으로 실행하는 데 필요한 AWS 서비스와 기능이 포함되어 있는지 검증합니다.
2. 코드형 인프라(IaC)를 사용합니다. 프로덕션 인프라 및 애플리케이션 구성 템플릿을 정확하게 유지하고 재해 복구 환경에 정기적으로 적용합니다. AWS CloudFormation은 CloudFormation 템플릿이 지정하는 것과 실제로 배포된 것 사이의 드리프트를 감지할 수 있습니다.
3. 기본 및 DR 사이트를 포함한 모든 환경에 애플리케이션 및 인프라 업데이트를 배포하도록 CI/CD 파이프라인을 구성합니다. AWS CodePipeline과 같은 CI/CD 솔루션은 배포 프로세스를 자동화할 수 있으므로 구성 드리프트의 위험이 줄어듭니다.

4. 기본 환경과 DR 환경의 배포에 시차를 둡니다. 이 접근 방식을 사용하면 기본 환경에서 업데이트를 처음 배포하고 테스트할 수 있습니다. 이렇게 하면 문제가 DR 사이트에 전파되기 전에 기본 사이트에 격리됩니다. 이 접근 방식은 결함이 동시에 프로덕션 및 DR 사이트로 푸시되는 것을 방지하고 DR 환경의 무결성을 유지합니다.
5. 기본 환경과 DR 환경 모두에서 리소스 구성을 지속적으로 모니터링합니다. AWS Config와 같은 솔루션은 구성 규정 준수를 적용하고 드리프트를 감지하는 데 도움이 되므로 환경 전반에서 일관된 구성을 유지하는 데 유용합니다.
6. 구성 드리프트, 데이터 복제 중단 또는 지연을 추적하고 알릴 수 있는 알림 메커니즘을 구현합니다.
7. 감지된 구성 드리프트의 수정을 자동화합니다.
8. 기본 구성과 DR 구성 간의 지속적인 일치를 확인하기 위해 정기적인 감사 및 규정 준수 검사 일정을 수립합니다. 정기 검토를 통해 정의된 규칙을 준수하고 해결해야 할 불일치를 식별할 수 있습니다.
9. AWS 프로비저닝된 용량, 서비스 할당량, 스로틀 제한, 구성 및 버전 불일치가 있는지 확인합니다.

리소스

관련 모범 사례:

- REL01-BP01 서비스 할당량 및 제약 조건 인식
- REL01-BP02 계정 및 리전 전체에서 서비스 할당량 관리
- REL01-BP04 할당량 모니터링 및 관리
- REL13-BP03 재해 복구 구현을 테스트하여 구현 확인

관련 문서:

- Remediating Noncompliant AWS Resources by AWS Config 규칙
- AWS Systems Manager Automation
- AWS CloudFormation: 스택 및 리소스에 대한 비관리형 구성 변경 감지
- AWS CloudFormation: 전체 CloudFormation 스택의 드리프트 감지
- AWS Systems Manager Automation
- AWS에서 워크로드 재해 복구: 클라우드에서의 복구(AWS 백서)
- AWS에서 인프라 구성 관리 솔루션을 구현하려면 어떻게 해야 하나요?
- Remediating Noncompliant AWS Resources by AWS Config 규칙

관련 비디오:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

관련 예제:

- AWS CloudFormation 레지스트리
- Quota Monitor for AWS
- Implement automatic drift remediation for AWS CloudFormation using Amazon CloudWatch and AWS Lambda
- AWS Architecture Blog: Disaster Recovery Series
- AWS Marketplace: 재해 복구에 사용할 수 있는 제품
- 안전하고 간편한 배포 자동화

REL13-BP05 자동 복구

장애로 인한 위험과 비즈니스 영향을 줄이기 위해 안정적이고 관찰 가능하며 재현 가능하며 테스트되고 자동화된 복구 메커니즘을 구현합니다.

원하는 성과: 복구 프로세스를 위해 잘 문서화되고 표준화되고 철저하게 테스트된 자동화 워크플로를 구현했습니다. 복구 자동화는 데이터 손실 또는 사용 불가 위험이 낮은 사소한 문제를 자동으로 수정합니다. 심각한 인시던트에 대한 복구 프로세스를 빠르게 호출하고, 운영 중에 복구 동작을 관찰하고, 위험한 상황이나 장애가 관찰되면 프로세스를 종료할 수 있습니다.

일반적인 안티 패턴:

- 복구 계획의 일환으로 실패하거나 성능이 저하된 상태에 있는 구성 요소 또는 메커니즘에 의존합니다.
- 복구 프로세스에 콘솔 액세스(클릭 작업이라고도 함)와 같은 수동 개입이 필요합니다.
- 데이터 손실 또는 사용 불가 위험이 높은 상황에서 복구 절차를 자동으로 시작합니다.
- 작동하지 않거나 추가 위험이 있는 복구 절차를 중단하는 메커니즘(예: Andon 코드 또는 큰 빨간색 중지 버튼)을 포함하지 않습니다.

이 모범 사례 확립의 이점:

- 복구 작업의 신뢰성, 예측 가능성 및 일관성이 높아집니다.

- 목표 복구 시간(RTO) 및 목표 복구 시점(RPO)을 포함하여 더 엄격한 복구 목표를 충족할 수 있습니다.
- 인시던트 발생 시 복구 실패 가능성이 줄어듭니다.
- 인적 오류가 발생하기 쉬운 수동 복구 프로세스와 관련된 장애 위험이 감소합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

자동 복구를 구현하려면 AWS 서비스와 모범 사례를 사용하는 포괄적인 접근 방식이 필요합니다. 시작하려면 워크로드에서 중요한 구성 요소와 잠재적 장애 지점을 식별하세요. 사람의 개입 없이 워크로드와 데이터를 장애로부터 복구할 수 있는 자동화된 프로세스를 개발합니다.

코드형 인프라(IaC) 원칙을 사용하여 복구 자동화를 개발합니다. 이렇게 하면 복구 환경이 소스 환경과 일관적이고 복구 프로세스의 버전을 관리할 수 있습니다. 복잡한 복구 워크플로를 오케스트레이션하려면 AWS Systems Manager Automations 또는 AWS Step Functions과 같은 솔루션을 고려하세요.

복구 프로세스의 자동화는 상당한 이점을 제공하며 목표 복구 시간(RTO) 및 목표 복구 시점(RPO)을 보다 쉽게 달성하는 데 도움이 될 수 있습니다. 그러나 예상치 못한 상황이 발생하여 장애가 발생하거나 추가 가동 중지 시간 및 데이터 손실과 같은 자체 위험을 초래할 수 있습니다. 이 위험을 완화하려면 진행 중인 복구 자동화를 빠르게 중단할 수 있는 기능을 제공합니다. 중단되면 조사하고 수정 조치를 취할 수 있습니다.

지원되는 워크로드의 경우 자동 장애 조치를 제공하기 위해 AWS Elastic Disaster Recovery(AWS DRS)와 같은 솔루션을 고려합니다. AWS DRS는 운영 체제, 시스템 상태 구성, 데이터베이스, 애플리케이션 및 파일을 포함한 시스템을 대상 AWS 계정 및 선호 리전의 스테이징 영역에 지속적으로 복제합니다. 인시던트가 발생하면 AWS DRS는 복제된 서버를 AWS의 복구 리전에서 완전히 프로비저닝된 워크로드로 자동 변환합니다.

자동 복구의 유지 관리 및 개선은 지속적인 프로세스입니다. 얻은 교훈을 기반으로 복구 절차를 지속적으로 테스트하고 개선하며 복구 기능을 향상할 수 있는 새로운 AWS 서비스와 기능에 대한 최신 정보를 파악하세요.

구현 단계

1. 자동 복구 계획
   a. 워크로드 아키텍처, 구성 요소 및 종속성을 철저히 검토하여 자동화된 복구 메커니즘을 식별하고 계획합니다. 워크로드의 종속성을 하드 종속성과 소프트 종속성으로 분류합니다. 하드 종속성은 존재하지 않으면 워크로드가 작동할 수 없고 대체할 수 없는 종속성입니다. 소프트 종속성은 워

크로드가 일반적으로 사용하지만 임시 대체 시스템 또는 프로세스로 대체할 수 있거나 단계적 성능 저하로 처리할 수 있는 종속성입니다.

b. 누락되거나 손상된 데이터를 식별하고 복구하는 프로세스를 설정합니다.

c. 복구 작업이 완료된 후 복구된 정상 상태를 확인하는 단계를 정의합니다.

d. 사전 워밍 및 캐시 채우기 등 복구된 시스템을 완전한 서비스를 위해 준비하는 데 필요한 모든 작업을 고려합니다.

e. 복구 프로세스 중에 발생할 수 있는 문제와 이를 감지하고 해결하는 방법을 고려합니다.

f. 기본 사이트와 해당 컨트롤 플레인에 액세스할 수 없는 시나리오를 고려합니다. 기본 사이트에 의존하지 않고 복구 작업을 독립적으로 수행할 수 있는지 확인합니다. DNS 레코드를 수동으로 변경하지 않고도 트래픽을 리디렉션할 수 있는 Amazon Application Recovery Controller(ARC)와 같은 솔루션을 고려해 보세요.

2. 자동 복구 프로세스 개발

a. 핸즈프리 복구를 위한 자동 장애 탐지 및 장애 조치 메커니즘을 구현합니다. Amazon CloudWatch와 같은 대시보드를 구축하여 자동 복구 절차의 진행 상황과 상태를 보고합니다. 성공적인 복구를 검증하는 절차를 포함합니다. 진행 중인 복구를 중단하는 메커니즘을 제공합니다.

b. 자동으로 복구할 수 없는 장애에 대한 대체 프로세스로 플레이북을 구축하고 재해 복구 계획을 고려합니다.

c. REL13-BP03에서 설명한 대로 복구 프로세스를 테스트합니다.

3. 복구 준비

a. 복구 사이트의 상태를 평가하고 중요한 구성 요소를 미리 배포합니다. 자세한 내용은 REL13-BP04를 참조하세요.

b. 조직 전반에서 관련 이해관계자 및 팀을 관여시켜 복구 작업에 대한 명확한 역할, 책임 및 의사 결정 프로세스를 정의합니다.

c. 복구 프로세스를 시작할 조건을 정의합니다.

d. 복구 프로세스를 되돌리고 필요한 경우 또는 안전한 것으로 간주된 후 기본 사이트로 되돌릴 계획을 수립합니다.

리소스

관련 모범 사례:

- REL07-BP01 리소스를 확보하거나 조정할 때 자동화 사용
- REL11-BP01 워크로드의 모든 구성 요소를 모니터링하여 장애 감지
- REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용

- REL13-BP03 재해 복구 구현을 테스트하여 구현 확인
- REL13-BP04 사이트 또는 리전에서 구성 드리프트 관리

관련 문서:

- AWS Architecture Blog: Disaster Recovery Series
- AWS에서 워크로드 재해 복구: 클라우드에서의 복구(AWS 백서)
- Orchestrate Disaster Recovery Automation using Amazon Route 53 ARC and AWS Step Functions
- Build AWS Systems Manager Automation runbooks using AWS CDK
- AWS Marketplace: Products That Can Be Used for Disaster Recovery
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Using Elastic Disaster Recovery for Failover and Failback
- AWS Elastic Disaster Recovery 리소스
- APN 파트너: 재해 복구를 지원할 수 있는 파트너

관련 비디오:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

성능 효율성

성능 효율성 원칙에는 클라우드 리소스를 성능 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력이 포함됩니다. 구현에 대한 권장 가이드는 성능 효율성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 아키텍처 선택
- 컴퓨팅 및 하드웨어
- 데이터 관리
- 네트워킹 및 콘텐츠 전송

- 프로세스 및 문화

아키텍처 선택

Questions

- PERF 1. 워크로드에 적합한 클라우드 리소스 및 아키텍처를 어떻게 선택하나요?

PERF 1. 워크로드에 적합한 클라우드 리소스 및 아키텍처를 어떻게 선택하나요?

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.

모범 사례

- PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해
- PERF01-BP02 클라우드 제공업체 또는 적절한 파트너의 지침을 사용하여 아키텍처 패턴 및 모범 사례에 대해 알아보기
- PERF01-BP03 아키텍처 결정에 비용 고려
- PERF01-BP04 장단점이 고객과 아키텍처 효율성에 미치는 영향 평가
- PERF01-BP05 정책 및 참조 아키텍처 사용
- PERF01-BP06 벤치마킹을 사용하여 아키텍처 결정
- PERF01-BP07 아키텍처 선택에 데이터 기반 접근 방식 사용

PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해

워크로드 아키텍처에서 더 나은 아키텍처 결정을 내리고 성능 효율성을 높일 수 있도록 사용 가능한 서비스 및 구성에 대해 지속적으로 학습하고 살펴보세요.

일반적인 안티 패턴:

- 클라우드를 코로케이션된 데이터 센터로 사용합니다.
- 클라우드로 마이그레이션한 후에는 애플리케이션을 현대화하지 않습니다.
- 지속해야 하는 모든 항목에 하나의 스토리지 유형만 사용합니다.
- 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 확립의 이점: 새로운 서비스 및 구성을 고려하여 성능을 현저히 개선하고 비용을 절감하며 워크로드를 유지 관리하는 데 필요한 노력을 최적화할 수 있습니다. 또한 클라우드 지원 제품의 가치 실현 시간을 단축하는 데 도움이 될 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS는 성능을 개선하고 클라우드 워크로드 비용을 절감할 수 있는 새로운 서비스와 기능을 지속적으로 출시하고 있습니다. 클라우드에서 성능 효율성을 유지하려면 이러한 새로운 서비스와 기능을 최신 상태로 유지해야 합니다. 또한 워크로드 아키텍처를 현대화하면 생산성을 가속화하고 혁신을 촉진하며 더 많은 성장 기회를 확보하는 데 도움이 됩니다.

구현 단계

- 관련 서비스에 대한 워크로드 소프트웨어 및 아키텍처 인벤토리를 생성합니다. 자세히 알아볼 제품 범주를 결정합니다.
- AWS 제품 및 서비스를 탐색하여 성능을 개선하고 비용 및 운영 복잡성을 줄이는 데 도움이 되는 관련 서비스 및 구성 옵션을 식별하고 알아봅니다.
  - Amazon Web Services Cloud
  - AWS Academy
  - AWS의 새로운 소식
  - AWS 블로그
  - AWS Skill Builder
  - AWS 이벤트 및 웨비나
  - AWS 교육 및 자격증
  - AWS YouTube 채널
  - AWS 워크숍
  - AWS Communities
- Amazon Q를 사용하여 서비스에 대한 관련 정보와 조언을 얻을 수 있습니다.
- 샌드박스(비프로덕션) 환경을 사용하여 추가 비용 없이 새로운 서비스를 알아보고 실험할 수 있습니다.
- 새로운 클라우드 서비스 및 기능에 대해 지속적으로 알아보세요.

리소스

관련 문서:

- Overview of Amazon Web Services
- Amazon EC2 features
- AWS 파트너 학습 계획으로 단계별 학습 시작하기
- AWS 교육 및 자격증
- My learning path to become an AWS solutions architect
- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터
- AWS에서 현대적 애플리케이션 구축

관련 비디오:

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2022 - Reduce your operational and infrastructure costs with Amazon ECS
- AWS re:Invent 2023 - Build with the efficiency, agility & innovation of the cloud with AWS
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- This is my Architecture

관련 예제:

- AWS 샘플
- AWS SDK Examples

PERF01-BP02 클라우드 제공업체 또는 적절한 파트너의 지침을 사용하여 아키텍처 패턴 및 모범 사례에 대해 알아보기

클라우드 회사 리소스(예: 설명서, 솔루션스 아키텍트, 전문 서비스 또는 적절한 파트너)를 의사 결정의 지침으로 사용하세요. 이러한 리소스는 최적의 성능을 위해 아키텍처를 검토하고 개선하는 데 도움이 됩니다.

일반적인 안티 패턴:

- AWS를 공통된 클라우드 제공업체로 사용합니다.
- AWS 서비스를 설계 의도와 다른 방식으로 사용합니다.
- 비즈니스 상황을 고려하지 않고 모든 지침을 따릅니다.

이 모범 사례 확립의 이점: 클라우드 제공업체 또는 적절한 파트너의 지침을 사용하면 워크로드에 적합한 아키텍처를 선택하고 의사 결정에 확신을 얻을 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS는 효율적인 클라우드 워크로드를 구축하고 관리하는 데 도움이 되는 다양한 지침, 설명서 및 리소스를 제공합니다. AWS 설명서에는 코드 샘플, 자습서, 자세한 서비스 설명이 포함되어 있습니다. AWS는 설명서 외에도 고객이 클라우드 서비스의 다양한 측면을 탐색하고 AWS에서 효율적인 클라우드 아키텍처를 구현하는 데 도움이 되는 교육 및 인증 프로그램, 솔루션 아키텍트, 전문 서비스를 제공합니다.

이러한 리소스를 활용하면 유용한 지식과 모범 사례에 대한 인사이트를 얻고, 시간을 절약하며, AWS 클라우드에서 더 나은 성과를 달성할 수 있습니다.

구현 단계

- AWS 설명서 및 지침을 검토하고 모범 사례를 따릅니다. 이러한 리소스는 서비스를 효과적으로 선택 및 구성하고 더 나은 성능을 달성하는 데 도움이 될 수 있습니다.
  - AWS 설명서(예: 사용 설명서 및 백서)
  - AWS 블로그
  - AWS 교육 및 자격증
  - AWS YouTube 채널
- AWS 파트너 이벤트(예: AWS Global Summits, AWS re:Invent. 사용자 그룹, 워크숍)에 참가하여 AWS 전문가로부터 AWS 서비스 사용 모범 사례에 대해 알아보세요.
  - AWS 파트너 학습 계획으로 단계별 학습 시작하기
  - AWS 이벤트 및 웨비나
  - AWS 워크숍
  - AWS Communities

- 추가 지침이나 제품 정보가 필요한 경우 AWS에 문의하여 도움을 받으세요. AWS 솔루션스 아키텍트 및 AWS Professional Services는 솔루션 구현에 대한 지침을 제공합니다. AWS 파트너는 비즈니스 민첩성 및 혁신을 달성하는 데 도움이 되는 AWS 전문 지식을 제공합니다.
- 지원를 사용하여 서비스를 효과적으로 사용하는 데 필요한 기술 지원을 받으세요. 지원 플랜은 성능을 최적화하고, 위험을 통제하며, 비용을 제어하면서 AWS를 성공적으로 사용할 수 있도록 적절한 도구 조합 및 전문 지식에 대한 액세스를 제공하도록 설계되었습니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터
- AWS Enterprise Support

관련 비디오:

- This is my Architecture
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Implementing distributed design patterns on AWS
- AWS re:Invent 2023 - Application architecture as code

관련 예제:

- AWS 샘플
- AWS SDK Examples
- AWS Analytics Reference Architecture

PERF01-BP03 아키텍처 결정에 비용 고려

클라우드 워크로드의 리소스 사용률과 성능 효율성을 개선하기 위해 비용 측면을 고려하여 아키텍처를 결정하세요. 클라우드 워크로드의 비용이 미치는 영향을 인식하면 효율적인 리소스를 활용하고 낭비적인 작업을 줄일 가능성이 커집니다.

일반적인 안티 패턴:

- 인스턴스 패밀리는 하나만 사용합니다.
- 라이선스가 부여된 솔루션과 오픈 소스 솔루션을 비교 평가하지 않습니다.
- 스토리지 수명 주기 정책을 정의하지 않습니다.
- AWS 클라우드의 새로운 서비스 및 기능을 검토하지 않습니다.
- 블록 스토리지만 사용합니다.

이 모범 사례 확립의 이점: 의사 결정에 비용을 고려하면 보다 효율적인 리소스를 사용하고 다른 투자를 모색할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

워크로드를 비용에 맞게 최적화하면 리소스 사용률을 높이고 클라우드 워크로드에서 낭비를 방지할 수 있습니다. 아키텍처 결정 시 비용을 고려하면 일반적으로 워크로드 구성 요소의 크기를 규모에 맞게 조정하고 탄력성을 활성화하여, 클라우드 워크로드 성능 효율성의 향상으로 이어집니다.

구현 단계

- 클라우드 워크로드에 대한 예산 한도와 같은 비용 목표를 설정합니다.
- 워크로드 비용을 유발하는 주요 구성 요소(인스턴스 및 스토리지 등)를 파악합니다. AWS Pricing Calculator 및 AWS Cost Explorer를 사용하여 워크로드의 주요 비용 요인을 식별할 수 있습니다.
- 온디맨드, 예약형 인스턴스, ,절감형 플랜, 스팟 인스턴스에서 가격 책정 모델을 이해합니다.
- Well-Architected 비용 최적화 모범 사례를 사용하여 이러한 주요 구성 요소를 비용에 맞게 최적화합니다.
- 지속적으로 비용을 모니터링하고 분석하여 워크로드에서 비용 최적화 기회를 파악합니다.
  - AWS Budgets를 사용하여 수용 불가능한 비용에 대해 알림을 받습니다.
  - AWS Compute Optimizer 또는 AWS Trusted Advisor를 사용하여 비용 최적화 권장 사항을 받습니다.
  - AWS Cost Anomaly Detection을 사용하여 자동화된 비용 이상 탐지 및 근본 원인 분석을 수행할 수 있습니다.

리소스

관련 문서:

- What is AWS Billing and Cost Management?
- Cost Optimization with AWS
- Choosing an AWS cost management strategy
- A Beginner's Guide to AWS Cost Management
- A Detailed Overview of the Cost Intelligence Dashboard
- AWS 아키텍처 센터
- AWS Solutions Library
- AWS 지식 센터

관련 비디오:

- This is my Architecture
- AWS re:Invent 2023 - What's new with AWS cost optimization
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

관련 예제:

- AWS Compute Optimizer Demo code
- Cost Optimization 워크숍
- 클라우드 재무 관리 기술 구현 플레이북
- Startup optimization: Tuning application performance for maximum efficiency
- Serverless Optimization 워크숍(Performance and Cost)
- Scaling cost effective architectures

PERF01-BP04 장단점이 고객과 아키텍처 효율성에 미치는 영향 평가

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, 이러한 변화의 최종적으로 일관된 특성이 고객에게 미치는 영향을 평가하는 것이 중요합니다.

일반적인 안티 패턴:

- 구현에 대한 장단점이 있더라도 모든 성능 이점을 구현해야 한다고 가정합니다.
- 성능 문제가 심각한 지점에 도달했을 때만 워크로드 변경을 평가합니다.

이 모범 사례 확립의 이점: 잠재적인 성능 관련 개선 사항을 평가할 때는 변경 사항에 대한 장단점이 워크로드 요구 사항에서 수용 가능한지 여부를 결정해야 합니다. 경우에 따라 장단점을 보상하기 위해 추가 제어를 구현해야 할 수도 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

성능과 고객 영향 측면에서 아키텍처의 중요한 영역을 파악합니다. 성능을 개선할 수 있는 방법과 해당 개선 과정의 장단점, 개선 사항이 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.

구현 단계

- 워크로드 요구 사항과 SLA를 이해합니다.
- 평가 요소를 명확하게 정의합니다. 평가 요소는 워크로드의 비용, 신뢰성, 보안 및 성능과 관련될 수 있습니다.
- 요구 사항을 충족할 수 있는 아키텍처 및 서비스를 선택합니다.
- 실험 및 개념 증명(POC)을 수행하여 장단점과 고객 및 아키텍처 효율성에 미치는 영향을 평가합니다. 일반적으로 가용성, 성능, 보안성이 높은 워크로드는 더 많은 클라우드 리소스를 소비하는 반면, 더 나은 고객 경험을 제공합니다. 워크로드의 복잡성, 성능, 비용 전반의 장단점을 이해합니다. 일반적으로 세 번째 요소 대신 첫 두 요소가 우선 고려됩니다.

리소스

관련 문서:

- Amazon Builders' Library
- QuickSight KPIs
- Amazon CloudWatch RUM
- X-Ray 설명서
- Understand resiliency patterns and trade-offs to architect efficiently in the cloud

관련 비디오:

- Optimize applications through Amazon CloudWatch RUM
- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

관련 예제:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP05 정책 및 참조 아키텍처 사용

워크로드를 설계하고 구현할 때 보다 효율적으로 서비스 및 구성을 선택할 수 있도록 내부 정책과 기존 참조 아키텍처를 사용하세요.

일반적인 안티 패턴:

- 회사의 관리 오버헤드에 영향을 줄 수 있는 다양한 기술을 허용합니다.

이 모범 사례 확립의 이점: 아키텍처, 기술 및 공급업체 선택에 대한 정책을 수립하면 신속하게 의사 결정을 내릴 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

리소스와 아키텍처를 선택할 때 내부 정책을 마련해 두면 아키텍처 선택 시 따라야 할 표준과 지침을 갖출 수 있습니다. 이러한 지침은 올바른 클라우드 서비스를 선택할 때 의사 결정 프로세스를 간소화하고 성능 효율성을 개선하는 데 도움이 될 수 있습니다. 정책 또는 참조 아키텍처를 사용하여 워크로드를 배포합니다. 서비스를 클라우드 배포에 통합한 다음, 성능 테스트를 사용하여 성능 요구 사항을 계속해서 충족할 수 있는지 확인합니다.

구현 단계

- 클라우드 워크로드의 요구 사항을 명확하게 이해합니다.
- 내부 및 외부 정책을 검토하여 가장 관련성이 높은 정책을 파악합니다.
- AWS에서 제공하는 적절한 참조 아키텍처 또는 업계 모범 사례를 사용합니다.

- 정책, 표준, 참조 아키텍처 및 일반적인 상황에 대한 규범적 지침으로 구성된 일련의 지침을 생성합니다. 이렇게 하면 팀이 더 빠르게 움직일 수 있습니다. 해당되는 경우 업종에 맞게 자산을 조정합니다.
- 샌드박스 환경에서 워크로드에 대한 이러한 정책과 참조 아키텍처를 검증합니다.
- 정책 및 참조 아키텍처가 클라우드 워크로드를 최적화하는 데 도움이 되도록 업계 표준 및 AWS 업데이트를 최신 상태로 유지합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터
- AWS 아키텍처 블로그

관련 비디오:

- This is my Architecture
- AWS re:Invent 2022 - Accelerate value for your business with SAP & AWS reference architecture

관련 예제:

- AWS 샘플
- AWS SDK Examples

PERF01-BP06 벤치마킹을 사용하여 아키텍처 결정

기존 워크로드의 성능을 벤치마킹하여 클라우드에서 어떻게 작동하는지 파악하고 해당 데이터를 기반으로 아키텍처 결정을 내릴 수 있습니다.

일반적인 안티 패턴:

- 워크로드의 특성을 나타내지 않는 일반적인 벤치마크를 사용합니다.
- 고객의 피드백과 관점을 유일한 벤치마크로 삼습니다.

이 모범 사례 확립의 이점: 현재 구현을 벤치마킹하면 성능 개선을 측정할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

통합 테스트에서 벤치마킹을 사용하여 워크로드 구성 요소의 성능을 평가합니다. 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 구축하거나 TPC-DS 등의 업계 표준 테스트를 사용하여 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 확인하세요.

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 테스트에서 프로덕션 환경의 워크로드 성능에 대한 정보를 얻을 수 있기 때문입니다.

구현 단계

- 계획 및 정의:
  - 벤치마크의 목표, 기준, 테스트 시나리오, 지표(예: CPU 사용률, 지연 시간, 처리량) 및 KPI를 정의합니다.
  - 사용자 경험 및 응답 시간, 접근성과 같은 요소의 측면에서 사용자 요구 사항에 중점을 둡니다.
  - 워크로드에 적합한 벤치마킹 도구를 식별합니다. AWS 서비스(예: Amazon CloudWatch) 또는 워크로드와 호환되는 서드파티 도구를 사용할 수 있습니다.
- 구성 및 계측:
  - 환경을 설정하고 리소스를 구성합니다.
  - 모니터링 및 로깅을 구현하여 테스트 결과를 캡처합니다.
- 벤치마크 및 모니터링:
  - 벤치마크 테스트를 수행하고 테스트 중에 지표를 모니터링합니다.
- 분석 및 문서화:

  - 벤치마킹 프로세스 및 조사 결과를 문서화합니다.
  - 결과를 분석하여 병목 현상, 추세 및 개선 영역을 식별합니다.
  - 테스트 결과를 사용하여 아키텍처 결정을 내리고 워크로드를 조정합니다. 이 과정에는 서비스가 변경되거나 새로운 기능이 도입될 수 있습니다.
- 최적화 및 반복:
  - 벤치마크를 기반으로 리소스 구성 및 할당을 조정합니다.
  - 조정 후 워크로드를 다시 테스트하여 개선 여부를 검증합니다.
  - 발견한 사실을 문서화하고 프로세스를 반복하여 개선이 필요한 다른 영역을 식별합니다.

리소스

관련 문서:

- AWS 아키텍처 센터
- AWS Partner Network
- AWS Solutions Library
- AWS 지식 센터
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Genomics workflows, Part 5: automated benchmarking
- Benchmark and optimize endpoint deployment in Amazon SageMaker AI JumpStart

관련 비디오:

- AWS re:Invent 2023 - Benchmarking AWS Lambda cold starts
- Benchmarking stateful services in the cloud
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

관련 예제:

- AWS 샘플

- AWS SDK Examples
- Distributed Load Tests
- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 아키텍처 선택에 데이터 기반 접근 방식 사용

아키텍처 선택에 대한 명확한 데이터 기반 접근 방식을 정의하여 특정 비즈니스 요구 사항을 충족하는 데 적합한 클라우드 서비스 및 구성이 사용되는지 확인합니다.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 업데이트하지 않아도 된다고 가정합니다.
- 아키텍처는 추측과 가정을 기반으로 선택됩니다.
- 시간이 지나면 타당한 이유 없이 아키텍처 변경을 도입합니다.

이 모범 사례 확립의 이점: 아키텍처 선택에 대한 접근 방식을 잘 정의하면 데이터를 사용하여 워크로드 설계에 영향을 미치고 시간이 지남에 따라 정보에 입각한 의사 결정을 내릴 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

클라우드에 대한 내부 경험과 지식 또는 공개된 사용 사례나 백서 등의 외부 리소스를 사용하여 아키텍처에서 리소스와 서비스를 선택합니다. 워크로드에 사용할 수 있는 서비스를 실험하고 벤치마킹할 수 있는 잘 정의된 프로세스를 갖추고 있어야 합니다.

중요한 워크로드의 백로그는 비즈니스 및 사용자와 관련된 기능을 제공하는 사용자 스토리뿐만 아니라 워크로드의 아키텍처 런웨이를 형성하는 기술 스토리로 구성되어야 합니다. 이 런웨이는 새로운 기술 발전과 새로운 서비스에 대한 정보를 수집하고 데이터와 적절한 근거를 바탕으로 이를 채택합니다. 이를 통해 아키텍처가 미래에 대비할 수 있고 정체되지 않을 수 있습니다.

구현 단계

- 주요 이해관계자와 협력하여 성능, 가용성 및 비용 고려 사항을 포함한 워크로드 요구 사항을 정의합니다. 워크로드의 사용자 수 및 사용 패턴과 같은 요소를 고려하세요.
- 기능 백로그와 함께 우선순위가 지정된 아키텍처 런웨이 또는 기술 백로그를 생성합니다.

- 다양한 클라우드 서비스를 평가합니다(자세한 내용은 PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해 참조).
- 성능 요구 사항을 충족하는 마이크로서비스 또는 서버리스와 같은 다양한 아키텍처 패턴을 탐색합니다(자세한 내용은 PERF01-BP02 클라우드 제공업체 또는 적절한 파트너의 지침을 사용하여 아키텍처 패턴 및 모범 사례에 대해 알아보기 참조).

- 다른 팀, 아키텍처 다이어그램 및 리소스(예: AWS Solution Architects, AWS 아키텍처 센터, AWS Partner Network)를 참조하여 워크로드에 적합한 아키텍처를 선택하는 데 도움을 받으세요.

- 워크로드의 성능을 평가하는 데 도움이 될 수 있는 처리량 및 응답 시간과 같은 성과 지표를 정의합니다.
- 정의된 지표를 실험하고 사용하여 선택한 아키텍처의 성능을 검증합니다.
- 아키텍처의 성능을 최적으로 유지하기 위해 지속적으로 모니터링하고 필요에 따라 조정합니다.
- 선택한 아키텍처와 결정 사항을 문서화하여 향후 업데이트 및 학습을 위한 참고 자료로 활용합니다.
- 학습한 내용, 새로운 기술, 현재 접근 방식에서 필요한 변경이나 문제를 나타내는 지표를 기반으로 아키텍처 선택 접근 방식을 지속적으로 검토하고 업데이트합니다.

리소스

관련 문서:

- AWS Solutions Library
- AWS 지식 센터
- Architectural Patterns to Build End-to-End Data Driven Applications on AWS

관련 비디오:

- This is my Architecture
- AWS re:Invent 2021 - Data-driven enterprise: Going from vision to value
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results

관련 예제:

- AWS 샘플
- AWS SDK Examples

컴퓨팅 및 하드웨어

Questions

- PERF 2. 워크로드에서 컴퓨팅 리소스를 선택하고 사용하는 방법은 무엇인가요?

PERF 2. 워크로드에서 컴퓨팅 리소스를 선택하고 사용하는 방법은 무엇인가요?

특정 워크로드에 대한 최적의 컴퓨팅 선택은 애플리케이션 설계, 사용량 패턴 및 구성 설정에 따라 다를 수 있습니다. 아키텍처는 다양한 컴포넌트에 대해 서로 다른 컴퓨팅 옵션을 사용하고 다양한 기능을 활성화하여 성능을 개선할 수 있습니다. 아키텍처에 대해 잘못된 컴퓨팅 옵션을 선택하면 성능 효율성 저하로 이어질 수 있습니다.

모범 사례

- PERF02-BP01 워크로드에 가장 적합한 컴퓨팅 옵션 선택
- PERF02-BP02 사용 가능한 컴퓨팅 구성 및 기능 파악
- PERF02-BP03 컴퓨팅 관련 지표 수집
- PERF02-BP04 컴퓨팅 리소스 구성 및 적정 크기 조정
- PERF02-BP05 컴퓨팅 리소스 동적 확장
- PERF02-BP06 최적화된 하드웨어 기반 컴퓨팅 액셀러레이터 사용

PERF02-BP01 워크로드에 가장 적합한 컴퓨팅 옵션 선택

워크로드에 가장 적합한 컴퓨팅 옵션을 선택하면 성능을 향상하고 불필요한 인프라 비용을 줄이며 워크로드를 유지하는 데 필요한 운영 노력을 줄일 수 있습니다.

일반적인 안티 패턴:

- 온프레미스에서 사용한 것과 동일한 컴퓨팅 옵션을 사용합니다.
- 클라우드 컴퓨팅 옵션, 기능 및 솔루션과 이러한 솔루션이 컴퓨팅 성능을 어떻게 개선할 수 있는지 잘 모릅니다.
- 대체 컴퓨팅 옵션이 워크로드 특성에 보다 적절한데도 규모 조정 또는 성능 요구 사항을 충족하려고 기존 컴퓨팅 옵션을 과도하게 프로비저닝합니다.

이 모범 사례 확립의 이점: 컴퓨팅 요구 사항을 파악하고 사용 가능한 옵션과 비교하여 평가하면 워크로드를 보다 리소스 효율적으로 만들 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

성능 효율성을 위해 클라우드 워크로드를 최적화하려면 사용 사례 및 성능 요구 사항에 가장 적합한 컴퓨팅 옵션을 선택하는 것이 중요합니다. AWS에서는 클라우드의 다양한 워크로드에 맞는 다양한 컴퓨팅 옵션을 제공합니다. 예를 들어, 가상 서버를 시작하고 관리하려면 Amazon EC2를 사용하고, 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행하려면 AWS Lambda를 사용하며, 컨테이너를 실행하고 관리하려면 Amazon ECS 또는 Amazon EKS를 사용하고, 대량의 데이터를 병렬로 처리하려면 AWS Batch을 사용할 수 있습니다. 규모와 컴퓨팅 요구 사항에 따라 상황에 맞는 최적의 컴퓨팅 솔루션을 선택하고 구성해야 합니다. 또한 각 솔루션에는 고유한 장단점이 있으므로 단일 워크로드에 여러 유형의 컴퓨팅 솔루션을 사용하는 것도 고려할 수 있습니다.

다음 단계에서는 워크로드 특성 및 성능 요구 사항에 맞는 올바른 컴퓨팅 옵션을 선택하는 방법을 안내합니다.

구현 단계

- 워크로드 컴퓨팅 요구 사항을 이해합니다. 고려해야 할 주요 요구 사항에는 처리 요구 사항, 트래픽 패턴, 데이터 액세스 패턴, 규모 조정 요구 사항, 지연 시간 요구 사항 등이 포함됩니다.
- 워크로드에 맞는 다양한 AWS 컴퓨팅 서비스에 대해 알아보세요. 자세한 내용은 PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해 단원을 참조하십시오. 다음은 몇 가지 주요 AWS 컴퓨팅 옵션, 특성 및 일반적인 사용 사례입니다.

AWS 서비스	주요 특징	일반 사용 사례
Amazon Elastic Compute Cloud(Amazon EC2)	하드웨어, 라이선스 요구 사항, 다양한 인스턴스 제품군, 프로세서 유형 및 컴퓨팅 액셀러레이터에 대한 전용 옵션 제공	리프트 앤 시프트 마이그레이션, 단일 애플리케이션, 하이브리드 환경, 엔터프라이즈 애플리케이션
Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS)	간편한 배포, 일관된 환경, 확장 가능	마이크로서비스, 하이브리드 환경

AWS 서비스	주요 특징	일반 사용 사례
AWS Lambda	서버리스 컴퓨팅 서비스(이벤트에 응답하여 코드를 실행하고 기본 컴퓨팅 리소스를 자동으로 관리하는 서비스).	마이크로서비스, 이벤트 기반 애플리케이션
AWS Batch	효율적이고 동적으로 프로비저닝 및 규모 조정 Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS) 및 AWS Fargate 컴퓨팅 리소스(작업 요구 사항에 따라 온디맨드 또는 스팟 인스턴스를 사용할 수 있는 옵션 포함)	HPC, ML 모델 학습
Amazon Lightsail	소규모 워크로드를 실행할 수 있도록 사전 구성된 Linux 및 Windows 애플리케이션	간단한 웹 애플리케이션, 맞춤형 웹 사이트

- 각 컴퓨팅 옵션과 관련된 비용(시간당 과금 또는 데이터 전송 등) 및 관리 오버헤드(패치 및 규모 조정 등)를 평가합니다.
- 비프로덕션 환경에서 실험 및 벤치마킹을 수행하여 워크로드 요구 사항을 가장 잘 해결할 수 있는 컴퓨팅 옵션을 파악합니다.
- 새로운 컴퓨팅 솔루션을 실험하고 파악한 후에는 마이그레이션을 계획하고 성과 지표를 검증합니다.
- Amazon CloudWatch와 같은 AWS 모니터링 도구 및 AWS Compute Optimizer와 같은 최적화 서비스를 사용해 실제 사용 패턴에 따라 지속적으로 컴퓨팅 리소스를 최적화합니다.

리소스

관련 문서:

- AWS에서의 클라우드 컴퓨팅
- Amazon EC2 인스턴스 유형

- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS 컨테이너: Amazon ECS 컨테이너
- 함수: Lambda 함수 구성
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

관련 비디오:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AMS
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Powering next-gen Amazon Elastic Compute Cloud: Deep dive on the Nitro System
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- AWS re:Invent 2019 - Amazon Elastic Compute Cloud foundations
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- 추론을 위한 ML 모델을 고성능 및 저렴한 비용으로 배포

관련 예제:

- Migrating the Web application to containers
- Run a Serverless Hello World
- Amazon EKS 워크샵
- Amazon EC2 워크숍
- Efficient and Resilient Workloads with Amazon Elastic Compute Cloud Auto Scaling
- Migrating to AWS Graviton with Container Services

PERF02-BP02 사용 가능한 컴퓨팅 구성 및 기능 파악

컴퓨팅 서비스에 사용 가능한 구성 옵션과 기능을 이해하면 적절한 양의 리소스를 프로비저닝하고 성능 효율성을 개선하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 워크로드 특성에 대해 컴퓨팅 옵션 또는 사용 가능한 인스턴스 제품군을 평가하지 않습니다.
- 최대 수요 요구 사항을 충족하기 위해 컴퓨팅 리소스를 과도하게 프로비저닝합니다.

이 모범 사례 확립의 이점: AWS 컴퓨팅 기능 및 구성을 숙지하여 워크로드 특성과 필요 사항에 맞게 최적화된 컴퓨팅 솔루션을 사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

각 컴퓨팅 솔루션에는 다양한 워크로드 특성과 요구 사항을 지원하기 위해 사용할 수 있는 고유한 구성과 기능이 있습니다. 이러한 옵션을 통해 워크로드를 보완할 수 있는 방식을 알아보고 애플리케이션에 가장 적합한 구성 옵션을 결정합니다. 이러한 옵션의 예로는 인스턴스 패밀리, 크기, 기능(GPU, I/O), 버스팅, 시간 초과, 함수 크기, 컨테이너 인스턴스 및 동시성이 있습니다. 워크로드에서 4주 이상 동일한 컴퓨팅 옵션을 사용하고 있으며 앞으로도 특성이 변하지 않을 것으로 예상되는 경우 AWS Compute Optimizer를 사용하여 CPU 및 메모리 관점에서 현재 컴퓨팅 옵션이 워크로드에 적합한지 알아볼 수 있습니다.

구현 단계

- 워크로드 요구 사항(예: CPU 필요량, 메모리, 지연 시간)을 이해합니다.
- AWS 설명서 및 모범 사례를 검토하여 컴퓨팅 성능을 개선하는 데 도움이 되는 권장 구성 옵션에 대해 알아봅니다. 고려해야 할 몇 가지 주요 구성 옵션은 다음과 같습니다.

구성 옵션	예시
인스턴스 유형	• 컴퓨팅 최적화 인스턴스는 높은 vCPU 대 메모리 비율이 필요한 워크로드에 이상적입니다. • 메모리 최적화 인스턴스는 상당한 메모리를 제공하여 메모리를 많이 사용하는 워크로드를 지원합니다.

구성 옵션	예시
	• 스토리지 최적화 인스턴스는 로컬 스토리지에 대해 높은 순차 읽기 및 쓰기 액세스(IOPS)가 필요한 워크로드에 적합하도록 설계되었습니다.
요금 모델	• 온디맨드 인스턴스를 사용하면 장기 약정 없이 시간 또는 초 단위로 컴퓨팅 용량을 이용할 수 있습니다. 이러한 인스턴스는 성능 기준 요구를 넘어서 버스팅하는 데 도움이 됩니다. • 절감형 플랜은 1년 또는 3년 동안 특정 양의 컴퓨팅 성능을 사용하기로 약정하는 대신 온디맨드 인스턴스에 비해 상당한 비용을 절감할 수 있습니다. • 스팟 인스턴스를 사용하면 내결함성을 갖춘 상태 비저장 워크로드를 위해 미사용 인스턴스 용량을 할인된 가격으로 활용할 수 있습니다.
Auto Scaling	Auto Scaling 구성을 사용하여 컴퓨팅 리소스를 트래픽 패턴에 일치시킵니다.
크기 조정	• Compute Optimizer를 사용하여 기계 학습 기반 추천을 통해 컴퓨팅 특성에 가장 적합한 컴퓨팅 구성을 추천받습니다. • AWS Lambda Power Tuning을 사용하여 Lambda 함수에 가장 적합한 구성을 선택합니다.

구성 옵션	예시
하드웨어 기반 컴퓨팅 액셀러레이터	• 가속 컴퓨팅 인스턴스는 그래픽 처리 또는 데이터 패턴 일치와 같은 기능을 CPU 기반 대안보다 더 효율적으로 수행합니다. • 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia, Amazon EC2 DL1과 같이 워크로드에 특정한 목적별 하드웨어를 활용합니다.

리소스

관련 문서:

- AWS에서의 클라우드 컴퓨팅
- Amazon EC2 인스턴스 유형
- Amazon EC2 Linux 인스턴스에 대한 프로세서 상태 제어
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS 컨테이너: Amazon ECS 컨테이너
- 함수: Lambda 함수 구성

관련 비디오:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations
- AWS re:Invent 2022 – Optimizing Amazon EKS for performance and cost on AWS

관련 예제:

- Compute Optimizer demo code

- Amazon EC2 spot instances 워크숍
- Efficient and Resilient Workloads with Amazon EC2 AWS Auto Scaling
- Graviton 개발자 워크숍
- AWS for Microsoft workloads immersion day
- AWS for Linux workloads immersion day
- AWS Compute Optimizer Demo code
- Amazon EKS 워크숍

PERF02-BP03 컴퓨팅 관련 지표 수집

컴퓨팅 관련 지표를 기록하고 추적하여 컴퓨팅 리소스의 성능을 더 잘 파악하고 성능과 사용률을 높입니다.

일반적인 안티 패턴:

- 지표에 대해 수동 로그 파일 검색만 사용합니다.
- 해당 모니터링 소프트웨어에서 기록한 기본 지표만 사용합니다.
- 문제가 발생한 경우에만 지표를 검토합니다.

이 모범 사례 확립의 이점: 성능 관련 지표를 수집하면 애플리케이션 성능을 비즈니스 요구 사항에 맞게 조정하여 워크로드 필요 사항을 충족하는 데 도움이 됩니다. 또한 워크로드의 리소스 성능과 사용률을 지속적으로 개선하는 데 도움이 될 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

클라우드 워크로드를 통해 지표, 로그 및 이벤트와 같은 대량의 데이터가 생성될 수 있습니다. AWS 클라우드에서 지표 수집은 보안, 비용 효율성, 성능 및 지속 가능성을 개선하기 위한 중요한 단계입니다. AWS는 Amazon CloudWatch와 같은 모니터링 서비스를 통해 다양한 성능 관련 지표를 제공하여 중요한 인사이트를 제공합니다. CPU 사용률, 메모리 사용률, 디스크 I/O, 네트워크 인바운드 및 아웃바운드와 같은 지표는 사용률 수준 또는 성능 병목 현상에 대한 인사이트를 제공할 수 있습니다. 데이터 기반 접근 방식의 일환으로 이 지표를 사용하면 워크로드 리소스를 능동적으로 튜닝하고 최적화할 수 있습니다. 비용 및 운영 목표를 지원하기 위해 보존 정책이 구현된 단일 플랫폼에서 컴퓨팅 리소스와 관련된 모든 지표를 수집하는 것이 이상적입니다.

구현 단계

- 워크로드와 관련된 성능 관련 지표를 파악합니다. 리소스 사용률과 클라우드 워크로드의 운영 방식(응답 시간 및 처리량 등)에 대한 지표를 수집해야 합니다.
  - Amazon EC2 기본 지표
  - Amazon ECS 기본 지표
  - Amazon EKS 기본 지표
  - Lambda 기본 지표
  - Amazon EC2 메모리 및 디스크 지표
- 워크로드에 적합한 로깅 및 모니터링 솔루션을 선택하고 설정합니다.
  - AWS 네이티브 관찰성
  - AWS Distro for OpenTelemetry
  - Amazon Managed Service for Prometheus
- 워크로드 요구 사항에 따라 지표에 필요한 필터 및 집계를 정의합니다.
  - Quantify custom application metrics with Amazon CloudWatch Logs and metric filters
  - Collect custom metrics with Amazon CloudWatch strategic tagging
- 보안 및 운영 목표에 맞게 지표에 대한 데이터 보존 정책을 구성합니다.
  - CloudWatch 지표에 대한 기본 데이터 보존
  - CloudWatch Logs에 대한 기본 데이터 보존
- 필요한 경우 지표에 대한 경보 및 알림을 생성하여 성능 관련 문제에 미리 대응할 수 있습니다.
  - Create alarms for custom metrics using Amazon CloudWatch anomaly detection
  - Create metrics and alarms for specific web pages with Amazon CloudWatch RUM
- 자동화를 사용하여 지표 및 로그 집계 에이전트를 배포합니다.
  - AWS Systems Manager 자동화
  - OpenTelemetry Collector

리소스

관련 문서:

- 모니터링 및 관찰성
- Best practices: implementing observability with AWS
- Amazon CloudWatch 설명서

- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- AWS Lambda에 대한 Amazon CloudWatch logs 액세스
- 컨테이너 인스턴스와 함께 CloudWatch Logs 사용
- 사용자 지정 지표 게시
- AWS Answers: Centralized Logging
- CloudWatch 지표를 게시하는 AWS 서비스
- Monitoring Amazon EKS on AWS Fargate

관련 비디오:

- AWS re:Invent 2023 – [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 – Implementing application observability
- AWS re:Invent 2023 – Building an effective observability strategy
- AWS re:Invent 2023 – Seamless observability with AWS Distro for OpenTelemetry
- Application Performance Management on AWS

관련 예제:

- AWS for Linux Workloads Immersion Day- Amazon CloudWatch
- Monitoring Amazon ECS clusters and containers
- Monitoring with Amazon CloudWatch dashboards
- Amazon EKS 워크숍

PERF02-BP04 컴퓨팅 리소스 구성 및 적정 크기 조정

워크로드의 성능 요구 사항에 맞게 컴퓨팅 리소스를 구성하고 규모에 맞게 크기를 조정하여 리소스의 사용률이 부족하거나 과도하게 활용되는 것을 방지하세요.

일반적인 안티 패턴:

- 워크로드 성능 요구 사항을 무시하여 컴퓨팅 리소스의 프로비저닝이 과도하거나 부족해지는 경우가 있습니다.
- 모든 워크로드에 사용할 수 있는 최소 또는 최대 인스턴스만 선택합니다.

- 관리하기 쉽도록 하나의 인스턴스 제품군만 사용합니다.
- 규모에 맞는 크기로 조정하기 위해 AWS Cost Explorer 또는 Compute Optimizer의 권장 사항을 무시합니다.
- 새로운 인스턴스 유형의 적합성을 위해 워크로드를 재평가하지 않습니다.
- 조직에 대해 소수의 인스턴스 구성만 인증합니다.

이 모범 사례 확립의 이점: 컴퓨팅 리소스의 크기를 규모에 맞게 적절히 조정하면 리소스의 프로비저닝이 과도하거나 부족해지는 것을 방지하여 클라우드 운영을 최적화할 수 있습니다. 일반적으로 컴퓨팅 리소스의 크기를 적절하게 조정하면 성능과 고객 경험이 향상되는 동시에 비용도 절감됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

규모에 맞게 크기를 조정하면 조직은 비즈니스 요구 사항을 해결하면서 효율적이고 비용 효과적인 방식으로 클라우드 인프라를 운영할 수 있습니다. 클라우드 리소스를 과도하게 프로비저닝하면 추가 비용이 발생할 수 있는 반면, 과소 프로비저닝은 성능 저하와 부정적인 고객 경험으로 이어질 수 있습니다. AWS는 AWS Compute Optimizer 및 AWS Trusted Advisor와 같은 도구를 제공합니다. 이러한 도구는 과거 데이터를 사용하여 컴퓨팅 리소스를 적정 크기로 조정하기 위한 권장 사항을 제공합니다.

구현 단계

- 요구 사항을 가장 잘 충족하는 인스턴스 유형을 선택하세요.
  - 워크로드에 적합한 EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
  - Amazon EC2 플릿의 속성 기반 인스턴스 유형 선택
  - 속성 기반 인스턴스 유형 선택을 사용하여 Auto Scaling 그룹 생성
  - Optimizing your Kubernetes compute costs with Karpenter consolidation
- 워크로드의 다양한 성능 특성 그리고 이러한 특성과 메모리, 네트워크, CPU 사용량 간의 관계를 분석합니다. 이 데이터를 사용하면 워크로드 프로필 및 성과 목표에 가장 적합한 리소스를 선택할 수 있습니다.
- Amazon CloudWatch와 같은 AWS 모니터링 도구를 사용하여 리소스 사용량을 모니터링합니다.
- 컴퓨팅 리소스에 적합한 구성을 선택합니다.
  - 임시 워크로드의 경우 인스턴스 Amazon CloudWatch 지표(예: `CPUUtilization`)를 평가하여 인스턴스의 사용률이 낮거나 높은지 식별합니다.

  - 안정적인 워크로드를 위해 정기적으로 AWS Compute Optimizer 및 AWS Trusted Advisor 등의 AWS 적정 크기 조정 도구를 확인하여 컴퓨팅 리소스를 최적화하고 크기를 조정할 수 있는 기회를 식별합니다.
- 실제 환경에서 구현하기 전에 비프로덕션 환경에서 구성 변경 사항을 테스트합니다.
- 새로운 컴퓨팅 오퍼링을 지속적으로 재평가하고 워크로드 요구 사항과 비교합니다.

리소스

관련 문서:

- AWS에서의 클라우드 컴퓨팅
- Amazon EC2 인스턴스 유형
- Amazon ECS 컨테이너: Amazon ECS 컨테이너
- Amazon EKS Containers: Amazon EKS Worker Nodes
- 함수: Lambda 함수 구성
- Amazon EC2 Linux 인스턴스에 대한 프로세서 상태 제어

관련 비디오:

- Amazon EC2 foundations
- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

관련 예제:

- AWS Compute Optimizer Demo code
- Amazon EKS 워크숍
- Right-sizing recommendations

PERF02-BP05 컴퓨팅 리소스 동적 확장

클라우드의 탄력성을 사용하여 필요에 따라 컴퓨팅 리소스를 동적으로 스케일 업 또는 스케일 다운하고 워크로드에 대한 프로비저닝 용량이 과도하거나 부족하지 않도록 방지할 수 있습니다.

일반적인 안티 패턴:

- 용량을 수동으로 늘려 경보에 대응합니다.
- 온프레미스에서와 동일한 크기 조정 가이드라인(일반적으로 정적 인프라)을 사용합니다.
- 조정 이벤트 후에 다시 스케일 다운하는 대신 증가된 용량을 그대로 둡니다.

이 모범 사례 확립의 이점: 컴퓨팅 리소스의 탄력성을 구성 및 테스트하면 비용을 절감하고, 성능 벤치마크를 유지하며, 트래픽 변화에 따른 신뢰성을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS는 다양한 조정 메커니즘을 통해 리소스를 동적으로 스케일 업 또는 스케일 다운하여 수요 변화에 대응할 수 있는 유연성을 제공합니다. 컴퓨팅 관련 지표와 결합된 동적 규모 조정을 통해 워크로드는 변경 사항에 자동으로 대응하고 최적의 컴퓨팅 리소스 세트를 사용하여 목표를 달성할 수 있습니다.

다양한 접근 방식을 사용하여 리소스 공급과 수요를 일치시킬 수 있습니다.

- 타겟 추적 접근 방식: 규모 조정 지표를 모니터링하고 필요에 따라 용량을 자동으로 늘리거나 줄입니다.
- 예측 규모 조정: 일별 및 주별 추세를 고려하여 스케일 인합니다.
- 일정 기반 접근 방식: 예측 가능한 로드 변화에 따라 자체 규모 조정 일정을 설정합니다.
- 서비스 규모 조정: 설계상 자동으로 규모가 조정되는 서비스(예: 서버리스)를 선택합니다.

워크로드 배포에서 스케일 업 및 스케일 다운 이벤트를 모두 처리할 수 있는지 확인해야 합니다.

구현 단계

- 컴퓨팅 인스턴스, 컨테이너 및 함수는 Auto Scaling과 함께 또는 서비스의 기능으로 탄력성을 위한 메커니즘을 제공합니다. 다음은 자동 규모 조정 메커니즘 예제입니다.

Auto Scaling 메커니즘	사용 장소
Amazon EC2 Auto Scaling	애플리케이션의 로드를 처리할 수 있는 올바른 수의 Amazon EC2 인스턴스를 유지하도록 보장합니다.
Application Auto Scaling	AWS Lambda 함수 또는 Amazon Elastic Container Service(Amazon ECS) 서비스와 같이 Amazon EC2 이외의 개별 AWS 서비스에서 리소스 규모를 자동으로 조정합니다.
Kubernetes Cluster Autoscaler/Karpenter	Kubernetes 클러스터를 자동으로 조정합니다.

- 규모 조정은 대개 Amazon EC2 인스턴스나 AWS Lambda 함수 등의 컴퓨팅 서비스와 관련하여 설명하는 경우가 많습니다. 이때 AWS Glue와 같은 비컴퓨팅 서비스의 구성도 수요에 맞게 고려해야 합니다.
- 규모 조정에 대한 지표가 배포 중인 워크로드의 특성과 일치하는지 확인합니다. 동영상 트랜스코딩 애플리케이션을 배포하는 경우 100%의 CPU 활용률이 예상되므로, 기본 지표로 사용해서는 안 됩니다. 대신 트랜스코딩 작업 대기열의 깊이를 사용합니다. 필요한 경우 규모 조정 정책에 사용자 지정 지표를 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 그룹의 인스턴스 수에 비례하여 증가하거나 감소합니다.
- Auto Scaling 그룹의 경우 수동 규모 조정 대신 동적 규모 조정을 사용해야 합니다. 또한 동적 규모 조정에서 목표 추적 조정 정책을 사용하는 것이 좋습니다.
- 워크로드 배포가 규모 조정 이벤트(스케일 업 및 스케일 다운)를 모두 처리할 수 있는지 확인합니다. 예를 들어 활동 내역을 사용하여 Auto Scaling 그룹의 조정 활동을 확인할 수 있습니다.
- 워크로드의 예측 가능한 패턴을 평가하고 예측 및 계획된 수요 변화에 따라 사전 예방적으로 확장합니다. 예측 규모 조정에서는 용량을 과도하게 프로비저닝할 필요가 없습니다. 자세한 내용은 Predictive Scaling with Amazon EC2 Auto Scaling을 참조하세요.

리소스

관련 문서:

- AWS에서의 클라우드 컴퓨팅

- Amazon EC2 인스턴스 유형
- Amazon ECS 컨테이너: Amazon ECS 컨테이너
- Amazon EKS Containers: Amazon EKS Worker Nodes
- 함수: Lambda 함수 구성
- Amazon EC2 Linux 인스턴스에 대한 프로세서 상태 제어
- Deep Dive on Amazon ECS Cluster Auto Scaling
- Introducing Karpenter – An Open-Source High-Performance Kubernetes Cluster Autoscaler

관련 비디오:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

관련 예제:

- Amazon EC2 Auto Scaling Group Examples
- Amazon EKS 워크샵
- Scale your Amazon EKS workloads by running on IPv6

PERF02-BP06 최적화된 하드웨어 기반 컴퓨팅 액셀러레이터 사용

하드웨어 액셀러레이터를 사용하면 CPU 기반 대안보다 특정 기능을 더 효율적으로 수행할 수 있습니다.

일반적인 안티 패턴:

- 워크로드에서 범용 인스턴스를 더 높은 성능과 더 낮은 비용을 제공할 수 있는 목적별 인스턴스와 비교하여 벤치마킹하지 않았습니다.
- CPU 기반 대안을 사용하는 것이 더 효율적일 수 있는 작업에 하드웨어 기반 컴퓨팅 액셀러레이터를 사용합니다.

- GPU 사용을 모니터링하지 않습니다.

이 모범 사례 확립의 이점: 그래픽 처리 디바이스(GPU) 및 Field Programmable Gate Array(FPGA)와 같은 하드웨어 기반 액셀러레이터를 사용하면 특정 프로세싱 기능을 보다 효율적으로 수행할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

가속 컴퓨팅 인스턴스는 GPU 및 FPGA와 같은 하드웨어 기반 컴퓨팅 가속기에 대한 액세스를 제공합니다. 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. 이 하드웨어는 필요한 시간 동안만 실행하고 필요하지 않은 경우 자동화를 통해 해제하여 전반적인 성능 효율성을 향상합니다.

구현 단계

- 요구 사항을 해결할 수 있는 가속 컴퓨팅 인스턴스를 식별합니다.
- 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia 및 Amazon EC2 DL1과 같이 워크로드에 따라 특정한 목적별 하드웨어의 장점을 활용합니다. AWS Inf2 인스턴스와 같은 Inferentia 인스턴스는 동급 Amazon EC2 인스턴스에 비해 최대 50% 더 우수한 와트당 성능을 제공합니다.
- 가속 컴퓨팅 인스턴스의 사용량 지표를 수집합니다. 예를 들어, Amazon CloudWatch의 NVIDIA GPU 지표 수집에서와 같이 CloudWatch 에이전트를 사용하여 GPU에 대한 `utilization_gpu` 및 `utilization_memory`와 같은 지표를 수집할 수 있습니다.
- 코드, 네트워크 운영 및 하드웨어 액셀러레이터 설정을 최적화하여 기본 하드웨어가 반드시 제대로 활용되도록 해야 합니다.
  - GPU 설정 최적화
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- 최신 고성능 라이브러리 및 GPU 드라이버를 사용합니다.
- 자동화를 사용하여 사용하지 않는 GPU 인스턴스 사용을 해제합니다.

리소스

관련 문서:

- Amazon Elastic Container Service에서 GPU 작업
- GPU 인스턴스
- Instances with AWS Trainium
- Instances with AWS Inferentia
- Let's Architect! Architecting with custom chips and accelerators

- 가속 컴퓨팅
- Amazon EC2 VT1 Instances
- 워크로드에 적합한 EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

관련 비디오:

- AWS re:Invent 2021 - How to select Amazon Elastic Compute Cloud GPU instances for deep learning
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

관련 예제:

- Amazon SageMaker AI and NVIDIA GPU Cloud (NGC)
- Use SageMaker AI with Trainium and Inferentia for optimized deep learning training and inferencing workloads
- Optimizing NLP models with Amazon Elastic Compute Cloud Inf1 instances in Amazon SageMaker AI

데이터 관리

Questions

- PERF 3. 워크로드의 데이터를 어떻게 저장, 관리, 액세스하나요?

PERF 3. 워크로드의 데이터를 어떻게 저장, 관리, 액세스하나요?

특정 시스템에 대한 최적의 스토리지 솔루션은 데이터 유형의 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 아카이브), 업데이트 빈도(WORM, 동적) 및 가용성과 내구성 제약 사항에 따라 다릅니다. Well-Architected 워크로드는 용도에 맞게 구축된 데이터 저장소를 사용하므로 다양한 기능을 통해 성능을 개선할 수 있습니다.

모범 사례

- PERF03-BP01 데이터 액세스 및 스토리지 요구 사항을 가장 잘 지원하는 목적별 데이터 스토어 사용
- PERF03-BP02 데이터 스토어에 사용 가능한 구성 옵션 평가
- PERF03-BP03 데이터 스토어 성과 지표 수집 및 기록
- PERF03-BP04 데이터 스토어에서 쿼리 성능을 개선하기 위한 전략 구현
- PERF03-BP05 캐싱을 활용하는 데이터 액세스 패턴 구현

PERF03-BP01 데이터 액세스 및 스토리지 요구 사항을 가장 잘 지원하는 목적별 데이터 스토어 사용

데이터 특성(공유 가능 여부, 크기, 캐시 크기, 액세스 패턴, 지연 시간, 처리량, 데이터 지속성 등)을 이해하여 워크로드에 적합한 목적별 데이터 스토어(스토리지 또는 데이터베이스)를 선택해야 합니다.

일반적인 안티 패턴:

- 하나의 특정 데이터베이스 솔루션에 대한 내부 경험과 지식만 갖춘 탓에 하나의 데이터 스토어만 고수합니다.
- 모든 워크로드의 데이터 스토리지 및 액세스 요구 사항이 비슷하다고 가정합니다.
- 데이터 자산의 인벤토리 등록을 위한 데이터 카탈로그를 구현하지 않았습니다.

이 모범 사례 확립의 이점: 데이터 특성과 요구 사항을 파악하면 워크로드 요구 사항을 충족하는 가장 효율적이고 성능이 뛰어난 스토리지 기술을 결정할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

데이터 스토리지를 선택하고 구현할 때는 쿼리, 규모 조정 및 스토리지 특성이 워크로드 데이터 요구 사항을 지원하는지 확인해야 합니다. AWS는 블록 스토리지, 객체 스토리지, 스트리밍 스토리지, 파일 시스템, 관계형, 키 값, 문서, 인메모리, 그래프, 시계열, 원장 데이터베이스를 포함한 다양한 데이터 스토리지 및 데이터베이스 기술을 제공합니다. 각 데이터 관리 솔루션에는 사용 사례 및 데이터 모델을

지원하는 옵션과 구성이 있습니다. 데이터 특성과 요구 사항을 이해하면 모놀리식 스토리지 기술과 제한적인 획일적 접근 방식에서 벗어나 데이터를 적절하게 관리하는 데 집중할 수 있습니다.

구현 단계

- 워크로드에 존재하는 다양한 데이터 유형의 인벤토리를 수행합니다.
- 다음과 같은 데이터 특성 및 요구 사항을 이해하고 문서화합니다.
  - 데이터 형식(비정형, 반정형, 관계형)
  - 데이터 볼륨 및 증가
  - 데이터 내구성: 영구, 임시, 일시적
  - 원자성, 일관성, 격리, 내구성(ACID) 요구 사항
  - 데이터 액세스 패턴(읽기 중심 또는 쓰기 중심)
  - 지연 시간
  - 처리량
  - 초당 입출력 연산 수(IOPS)
  - 데이터 보존 기간
- 데이터 특성에 맞는 AWS의 워크로드에 사용할 수 있는 다양한 데이터 스토어(스토리지 및 데이터베이스 서비스)에 대해 알아봅니다(PERF01-BP01 사용 가능한 클라우드 서비스 및 기능 학습 및 이해 참조). 다음은 AWS 스토리지 기술의 몇 가지 예와 주요 특성입니다.

유형	AWS 서비스	주요 특징
객체 스토리지	Amazon S3	무제한 확장성, 고가용성, 다양한 접근성 옵션. Amazon S3 안팎에서 객체를 전송하고 객체에 액세스할 때 전송 가속화 또는 액세스 포인트와 같은 서비스를 사용하여 위치, 보안 요구 사항 및 액세스 패턴을 지원할 수 있습니다.
아카이빙 스토리지	Amazon S3 Glacier	데이터 보관을 위해 제작되었습니다.
스트리밍 스토리지	Amazon Kinesis	스트리밍 데이터의 효율적인 수집 및 저장.

유형	AWS 서비스	주요 특징
	Amazon Managed Streaming for Apache Kafka(Amazon MSK)	
공유 파일 시스템	Amazon Elastic File System(Amazon EFS)	여러 유형의 컴퓨팅 솔루션에서 액세스할 수 있는 탑재 가능한 파일 시스템입니다.
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 서비스. Amazon EBS에는 IOPS 집약적 트랜잭션 워크로드를 위한 SSD 지원 스토리지와 처리량 집약적 워크로드를 위한 HDD 지원 스토리지가 포함됩니다.

유형	AWS 서비스	주요 특징
관계형 데이터베이스	Amazon Aurora, Amazon RDS, Amazon Redshift.	원자성, 일관성, 격리, 내구성(ACID) 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.
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.
문서 데이터베이스	Amazon DocumentDB	반정형 데이터를 JSON과 유사한 문서로 저장하도록 설계되었습니다. 이러한 데이터베이스는 개발자가 콘텐츠 관리, 카탈로그 및 사용자 프로필과 같은 애플리케이션을 신속하게 구축하고 업데이트하는 데 도움이 됩니다.

유형	AWS 서비스	주요 특징
인 메모리 데이터베이스	Amazon ElastiCache, Amazon MemoryDB for Redis	데이터에 대한 실시간 액세스, 최저 지연 시간 및 최대 처리량을 요하는 애플리케이션에 사용됩니다. 애플리케이션 캐싱, 세션 관리, 게임 순위표, 저지연 ML 특성 저장소, 마이크로서비스 메시징 시스템, 고처리량 스트리밍 메커니즘에 인 메모리 데이터베이스를 사용할 수 있습니다.
그래프 데이터베이스	Amazon Neptune	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.
시계열 데이터베이스	Amazon Timestream	시간이 지남에 따라 변화하는 데이터에서 효율적으로 분석 정보를 수집, 통합 및 도출합니다. IoT 애플리케이션, DevOps 및 산업용 원격 측정에 시계열 데이터베이스를 활용할 수 있습니다.

유형	AWS 서비스	주요 특징
와이드 컬럼	Amazon Keyspaces(Apache Cassandra용)	테이블, 행 및 열을 사용하지만 관계형 데이터베이스와 달리 동일한 테이블 내의 열의 이름과 형식이 행마다 다를 수 있습니다. 일반적으로 와이드 컬럼 스토어는 대규모 산업 앱에서 장비 유지 관리, 플릿 관리 및 라우팅 최적화를 위해 사용됩니다.
원장	Amazon Quantum Ledger Database(Amazon QLDB)	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.

- 데이터 플랫폼을 구축하는 경우 AWS의 최신 데이터 아키텍처를 활용하여 데이터 레이크, 데이터 웨어하우스, 목적별 데이터 스토어를 통합합니다.
- 워크로드에 맞는 데이터 스토어를 선택할 때 고려해야 할 주요 질문은 다음과 같습니다.

질문	고려할 사항
데이터는 어떻게 구성되어 있나요?	• 비정형 데이터인 경우 Amazon S3와 같은 객체 저장소 또는 Amazon DocumentDB와 같은 NoSQL 데이터베이스를 고려하세요. • 키 값 데이터의 경우 DynamoDB, Amazon ElastiCache(Redis OSS) 또는 Amazon MemoryDB를 고려하세요.

질문	고려할 사항
어느 정도 수준의 참조 무결성이 필요한가요?	• 외래 키 제약 조건의 경우 Amazon RDS 및 Aurora와 같은 관계형 데이터베이스가 이 수준의 무결성을 제공할 수 있습니다. • 일반적으로 NoSQL 데이터 모델 내에서는 문서나 테이블을 조인하는 대신 단일 요청으로 검색하도록 데이터를 단일 문서 또는 문서 모음으로 비정규화합니다.
원자성, 일관성, 격리, 내구성(ACID) 규정을 준수해야 하나요?	• 관계형 데이터베이스와 연결된 ACID 속성이 필요한 경우 Amazon RDS 및 Aurora와 같은 관계형 데이터베이스를 고려하세요. • NoSQL 데이터베이스에 강력한 일관성이 필요한 경우 DynamoDB를 사용하여 강력하게 일관된 읽기를 사용할 수 있습니다.
시간의 경과에 따라 스토리지 요구 사항이 어떻게 변하나요? 이러한 변화가 확장성에 어떤 영향을 미치나요?	• DynamoDB 및 Amazon Quantum Ledger Database(Amazon QLDB)와 같은 서버리스 데이터베이스는 동적으로 규모가 조정됩니다. • 관계형 데이터베이스는 프로비저닝된 스토리지에 대한 상한선이 있으며, 한도에 도달하면 샤딩과 같은 메커니즘을 통해 수평으로 분할해야 하는 경우가 많습니다.
쓰기 쿼리 대비 읽기 쿼리의 비율이 어떻게 되나요? 캐싱으로 성능이 향상될 가능성이 있나요?	• 읽기 중심의 워크로드는 데이터베이스가 DynamoDB인 경우 ElastiCache 또는 DAX와 같은 캐싱 계층을 활용할 수 있습니다. • Amazon RDS와 같은 관계형 데이터베이스를 사용하여 읽기 복제본으로 읽기를 오프로드할 수도 있습니다.

질문	고려할 사항
저장 및 수정(OLTP - 온라인 트랜잭션 처리) 또는 검색 및 보고(OLAP - 온라인 분석 처리)의 우선순위가 더 높은가요?	• 처리량이 많은 읽기 그대로의 트랜잭션 처리의 경우 DynamoDB와 같은 NoSQL 데이터베이스를 고려합니다. • 처리량이 많고 일관성이 있는 복잡한 읽기 패턴(예: 조인)의 경우 Amazon RDS를 사용합니다. • 분석 쿼리의 경우 Amazon Redshift와 같은 열 형식 데이터베이스를 고려하거나 Amazon S3로 데이터를 내보내고 Athena 또는 Amazon QuickSight를 사용하여 분석을 수행하는 방법을 고려하세요.
데이터에 필요한 내구성은 어느 정도인가요?	• Aurora는 리전 내 3개의 가용 영역에서 데이터를 자동으로 복제하므로, 데이터 손실 가능성이 적으면서도 데이터의 내구성이 매우 높아집니다. • DynamoDB는 여러 가용 영역에 걸쳐 자동으로 복제되므로, 높은 가용성과 데이터 내구성을 제공합니다. • 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.
상용 데이터베이스 엔진을 쓰고 싶지 않거나 라이선싱 비용을 들이고 싶지 않은가요?	• Amazon RDS 또는 Aurora에서 PostgreSQL 및 MySQL과 같은 오픈 소스 엔진을 고려합니다. • AWS Database Migration Service 및 AWS Schema Conversion Tool을 활용하여 상용 데이터베이스 엔진에서 오픈 소스로 마이그레이션을 수행합니다.

질문	고려할 사항
운영상 데이터베이스에 대해 어떤 점을 기대하나요? 주된 관심 사항이 관리형 서비스로 전환하는 것인가요?	• Amazon EC2 대신 Amazon RDS를 활용하고 자체 호스팅한 NoSQL 데이터베이스 대신 DocumentDB 또는 Amazon DocumentDB를 활용하여 운영 오버헤드를 줄일 수 있습니다.
현재 데이터베이스에 어떻게 액세스하고 있나요? 애플리케이션 액세스만 가능한가요? 아니면 비즈니스 인텔리전스(BI) 사용자와 기타 연결된 상용 애플리케이션이 있나요?	• 외부 도구에 대한 종속성이 있는 경우 해당 도구가 지원하는 데이터베이스와의 호환성을 유지해야 할 수 있습니다. Amazon RDS는 Microsoft SQL Server, Oracle, MySQL, PostgreSQL 등 지원하는 다양한 엔진 버전과 완벽하게 호환됩니다.

- 비프로덕션 환경에서 실험 및 벤치마킹을 수행하여 워크로드 요구 사항을 가장 잘 해결할 수 있는 데이터 스토어를 파악합니다.

리소스

관련 문서:

- Amazon EBS 볼륨 유형
- Amazon EC2 스토리지
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon S3 Glacier: S3 Glacier 설명서
- Amazon S3: 요청 속도 및 성능 고려 사항
- AWS 기반 클라우드 스토리지
- Amazon EBS I/O Characteristics
- AWS 클라우드 데이터베이스
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora 모범 사례
- Amazon Redshift 성능

- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB 모범 사례
- Choose between Amazon EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

관련 비디오:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimizing storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Building data mesh architectures on AWS
- AWS re:Invent 2023: Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023: Advanced data modeling with Amazon DynamoDB
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- Amazon DynamoDB deep dive: Advanced design patterns

관련 예제:

- AWS Purpose Built Databases 워크숍
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS
- Amazon S3 예제
- Optimize Data Pattern using Amazon Redshift Data Sharing
- Database Migrations
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On 워크숍
- Amazon Neptune Samples

PERF03-BP02 데이터 스토어에 사용 가능한 구성 옵션 평가

데이터 스토어에 사용할 수 있는 다양한 기능과 구성 옵션을 이해하고 평가하여 워크로드에 맞는 스토리지 공간과 성능을 최적화하세요.

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.
- 선택한 데이터 관리 솔루션의 구성 옵션을 알지 못합니다.
- 다른 사용 가능한 구성 옵션을 고려하지 않고 인스턴스 크기만 늘립니다.
- 데이터 스토어의 규모 조정 특성을 테스트하고 있지 않습니다.

이 모범 사례 확립의 이점: 데이터 스토어 구성을 탐색하고 실험하여 인프라 비용을 절감하고, 성능을 개선하며, 워크로드를 유지하는 데 들여야 하는 수고를 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

워크로드에는 데이터 스토리지 및 액세스 요구 사항에 따라 하나 이상의 데이터 스토어가 사용될 수 있습니다. 성능 효율성과 비용을 최적화하려면 데이터 액세스 패턴을 평가하여 적절한 데이터 스토어 구성을 결정해야 합니다. 데이터 스토어 옵션을 탐색하는 동안 스토리지 옵션, 메모리, 컴퓨팅, 읽기 복제본, 일관성 요구 사항, 연결 풀링 및 캐싱 옵션과 같은 다양한 측면을 고려하세요. 이러한 다양한 구성 옵션을 실험하여 성능 효율성 지표를 개선해 보세요.

구현 단계

- 데이터 스토어의 현재 구성(인스턴스 유형, 스토리지 크기, 데이터베이스 엔진 버전 등)을 파악합니다.
- AWS 설명서 및 모범 사례를 검토하여 데이터 스토어의 성능을 개선하는 데 도움이 되는 권장 구성 옵션에 대해 알아보세요. 다음과 같은 주요 데이터 스토어 옵션을 고려해야 합니다.

구성 옵션	예시
읽기 오프로드(예: 읽기 전용 복제본 및 캐싱)	• DynamoDB 테이블의 경우 캐싱을 위해 DAX를 사용하여 읽기를 오프로드할 수 있습니다.

구성 옵션	예시
	• Amazon ElastiCache(Redis OSS) 클러스터를 생성하고 요청된 항목이 없는 경우 데이터베이스로 되돌아가서 캐시에서 먼저 읽도록 애플리케이션을 구성할 수 있습니다. • Amazon RDS 및 Aurora와 같은 관계형 데이터베이스와 Neptune 및 Amazon DocumentDB와 같은 프로비저닝된 NoSQL 데이터베이스는 모두 읽기 전용 복제본을 추가하여 워크로드의 읽기 부분을 오프로드할 수 있도록 지원합니다. • DynamoDB와 같은 서버리스 데이터베이스는 자동으로 규모가 조정됩니다. 워크로드를 처리하기에 충분한 읽기 용량 단위(RCU)가 프로비저닝되었는지 확인합니다.

구성 옵션	예시
쓰기 규모 조정(예: 파티션 키 샤딩 또는 대기열 도입)	• 관계형 데이터베이스의 경우 인스턴스 크기를 늘려 확장된 워크로드를 수용하거나 프로비저닝된 IOPS를 늘려 기본 스토리지에 대한 처리량을 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 처리량을 향상시킬 수 있습니다. • DynamoDB와 같은 서버리스 데이터베이스는 자동으로 쓰기 처리량을 확장하거나 용량 모드에 따라 프로비저닝된 쓰기 용량 단위(WCU)를 조정하여 확장할 수 있습니다. • 특정 파티션 키에 대한 처리량 제한에 도달하면 핫 파티션과 관련된 문제가 발생할 수 있습니다. 이 문제는 보다 고르게 분산된 파티션 키를 선택하거나 파티션 키를 쓰기 샤딩하여 해결할 수 있습니다.

구성 옵션	예시
데이터 세트의 수명 주기 관리 정책	• Amazon S3 수명 주기를 사용하여 전체 수명 주기 동안 객체를 관리할 수 있습니다. 액세스 패턴을 알 수 없거나 패턴이 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 Storage Lens 지표를 활용하여 수명 주기 관리에서 최적화 기회와 격차를 식별할 수 있습니다. • Amazon EFS 수명 주기 관리는 파일 시스템에 대한 파일 스토리지를 자동으로 관리합니다.
연결 관리 및 풀링	• Amazon RDS 프록시는 Amazon RDS 및 Aurora와 함께 사용하여 데이터베이스에 대한 연결을 관리할 수 있습니다. • DynamoDB와 같은 서버리스 데이터베이스에는 연계된 연결이 없으니, 프로비저닝된 용량 및 자동 규모 조정 정책을 고려하여 로드 급증을 처리합니다.

- 비프로덕션 환경에서 실험 및 벤치마킹을 수행하여 워크로드 요구 사항을 해결할 수 있는 구성 옵션을 파악합니다.
- 실험을 마친 후에는 마이그레이션을 계획하고 성과 지표를 검증합니다.
- AWS 모니터링(예: Amazon CloudWatch) 및 최적화(예: Amazon S3 Storage Lens) 도구를 활용하고 실제 사용 패턴에 기반하여 데이터 스토어를 지속적으로 최적화합니다.

리소스

관련 문서:

- AWS 기반 클라우드 스토리지
- Amazon EBS 볼륨 유형

- Amazon EC2 스토리지
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon S3 Glacier: S3 Glacier 설명서
- Amazon S3: 요청 속도 및 성능 고려 사항
- Amazon EBS I/O Characteristics
- AWS 클라우드 데이터베이스
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora 모범 사례
- Amazon Redshift 성능
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB 모범 사례

관련 비디오:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimize storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2023: What's new with AWS file storage
- AWS re:Invent 2023: Dive deep into Amazon DynamoDB

관련 예제:

- AWS Purpose Built Databases 워크숍
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Amazon EBS Autoscale
- Amazon S3 예제

- Amazon DynamoDB Examples
- AWS Database migration samples
- Database Modernization 워크숍
- Working with parameters on your Amazon RDS for Postgress DB

PERF03-BP03 데이터 스토어 성과 지표 수집 및 기록

데이터 스토어에 대한 관련 성과 지표를 추적하고 기록하여 데이터 관리 솔루션의 성능을 파악할 수 있습니다. 이러한 지표는 데이터 스토어를 최적화하고, 워크로드 요구 사항이 충족되는지 확인하며, 워크로드 성능에 대한 명확한 개요를 제공하는 데 도움이 될 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 확립의 이점: 성능 기준선을 설정하면 워크로드의 정상적인 동작과 요구 사항을 이해하는 데 도움이 됩니다. 비정상적인 패턴을 더 빨리 식별하고 디버깅할 수 있어 데이터 스토어의 성능과 신뢰성이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

데이터 스토어의 성능을 모니터링하려면 일정 기간에 걸쳐 여러 성과 지표를 기록해야 합니다. 이를 통해 이상 징후를 탐지할 수 있을뿐더러 비즈니스 지표를 기준으로 성능을 측정하여 워크로드 요구 사항을 충족하는지 확인할 수 있습니다.

지표에는 데이터 스토어를 지원하는 기본 시스템과 데이터베이스 지표가 모두 포함되어야 합니다. 기본 시스템 지표에는 CPU 사용률, 메모리, 사용 가능한 디스크 스토리지, 디스크 I/O, 캐시 적중률, 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 데이터 관리 솔루션의 사용 방법을 이해하는 데 중요합니다. 이러한 지표를 데이터 중심 전략에 포함시켜 워크로드의 리소스를 조정하고 최적화할 수 있습니다.

데이터베이스 성능과 관련된 성능 측정값을 기록하는 도구, 라이브러리 및 시스템을 사용합니다.

구현 단계

- 추적할 데이터 스토어의 주요 성과 지표를 식별하세요.
  - Amazon S3 지표 및 차원
  - Amazon RDS 인스턴스에서 지표 모니터링
  - Amazon RDS의 성능 개선 도우미로 DB 로드 모니터링
  - Enhanced Monitoring 개요
  - DynamoDB 지표 및 차원
  - Monitoring DynamoDB Accelerator
  - Monitoring Amazon MemoryDB with Amazon CloudWatch
  - 어떤 메트릭을 모니터링해야 합니까?
  - Amazon Redshift 클러스터 성능 모니터링
  - Timestream metrics and dimensions
  - Amazon Aurora에 대한 Amazon CloudWatch 지표
  - Logging and monitoring in Amazon Keyspaces (for Apache Cassandra)
  - Monitoring Amazon Neptune Resources
- 승인된 로깅 및 모니터링 솔루션을 사용하여 이러한 지표를 수집합니다. Amazon CloudWatch는 아키텍처의 리소스 전반에서 지표를 수집할 수 있습니다. 또한 사용자 지정 지표를 수집하고 게시하여 비즈니스 또는 파생 지표를 파악할 수도 있습니다. CloudWatch 또는 서드파티 솔루션을 사용하여 임곗값 위반 시점을 나타내는 경보를 설정합니다.
- 데이터 스토어 모니터링에 성능 이상을 탐지하는 기계 학습 솔루션의 이점을 활용할 수 있는지 확인합니다.
  - Amazon RDS에 대한 Amazon DevOps Guru에서는 성능 문제에 대한 가시성을 제공하고 권장 수정 조치를 제안합니다.
- 보안 및 운영 목표에 맞게 모니터링 및 로깅 솔루션에서 데이터 보존을 구성합니다.
  - CloudWatch 지표에 대한 기본 데이터 보존
  - CloudWatch Logs에 대한 기본 데이터 보존

리소스

관련 문서:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora 모범 사례
- DynamoDB Accelerator
- Amazon DynamoDB 모범 사례
- Amazon Redshift Spectrum best practices
- Amazon Redshift 성능
- AWS 클라우드 데이터베이스
- Amazon RDS 성능 개선 도우미

관련 비디오:

- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Database Performance Monitoring and Tuning with Amazon DevOps Guru for Amazon RDS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

관련 예제:

- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring 워크숍
- AWS Purpose Built Databases 워크숍

PERF03-BP04 데이터 스토어에서 쿼리 성능을 개선하기 위한 전략 구현

데이터를 최적화하고 데이터 쿼리를 개선하는 전략을 구현하여 워크로드에 대한 확장성과 효율적인 성능을 향상할 수 있습니다.

일반적인 안티 패턴:

- 데이터 스토어에서는 데이터를 파티션하지 않습니다.
- 데이터 스토어에 하나의 파일 형식으로만 데이터를 저장합니다.
- 데이터 스토어에서 인덱스를 사용하지 않습니다.

이 모범 사례 확립의 이점: 데이터 및 쿼리 성능을 최적화하면 효율성이 향상되고 비용이 절감되며 사용자 경험이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

데이터 최적화 및 쿼리 튜닝은 전체 클라우드 워크로드의 성능과 응답성에 영향을 미치기 때문에 데이터 스토어의 성능 효율성에 있어 매우 중요한 측면입니다. 쿼리가 최적화되지 않으면 리소스 사용량과 병목 현상이 증가하여 데이터 스토어의 전반적인 효율성이 저하될 수 있습니다.

데이터 최적화는 효율적인 데이터 스토리지와 액세스를 위한 여러 기술을 포함합니다. 또한 데이터 저장소의 쿼리 성능을 개선하는 데도 도움이 됩니다. 주요 전략에는 데이터 파티셔닝, 데이터 압축, 데이터 비정규화가 포함되며, 이는 스토리지와 액세스 모두에 대해 데이터를 최적화하는 데 도움이 됩니다.

구현 단계

- 데이터 스토어에서 수행되는 중요한 데이터 쿼리를 이해하고 분석합니다.
- 데이터 스토어에서 느린 쿼리를 식별하고 쿼리 계획을 사용하여 현재 상태를 파악합니다.
  - Amazon Redshift에서 쿼리 계획 분석
  - Athena에서 EXPLAIN 및 EXPLAIN ANALYZE 사용
- 쿼리 성능을 개선하기 위한 전략을 구현합니다. 몇 가지 주요 전략은 다음과 같습니다.
  - 열 파일 형식(예: Parquet 또는 ORC)을 사용합니다.
  - 스토리지 공간 및 I/O 작업을 줄이기 위해 데이터 스토어의 데이터 압축.
  - 데이터를 더 작은 부분으로 분할하고 데이터 스캔 시간을 줄이기 위한 데이터 파티셔닝.
    - Athena에서 데이터 분할
    - 파티션 및 데이터 배포
  - 쿼리의 공통 열에서 데이터 인덱싱.
  - 빈번하게 사용되는 쿼리에는 구체화된 뷰를 사용하세요.
    - Understanding materialized views
    - Amazon Redshift에서 구체화된 뷰 생성

  - 쿼리에 적합한 조인 작업을 선택합니다. 두 테이블을 조인하는 경우 조인 왼쪽에 큰 테이블을 지정하고 조인 오른쪽에 작은 테이블을 지정합니다.
  - 지연 시간을 개선하고 데이터베이스 I/O 작업 횟수를 줄이기 위한 분산 캐싱 솔루션.
  - 정리, 리인덱싱, 통계 실행과 같은 정기적 유지 관리.
- 비운영 환경에서의 실험 및 테스트 전략.

리소스

관련 문서:

- Amazon Aurora 모범 사례
- Amazon Redshift 성능
- Amazon Athena top 10 performance tips
- AWS Database Caching
- Best Practices for Implementing Amazon ElastiCache
- Athena에서 데이터 분할

관련 비디오:

- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Optimize Amazon Athena Queries with New Query Analysis Tools

관련 예제:

- AWS Purpose Built Databases 워크숍

PERF03-BP05 캐싱을 활용하는 데이터 액세스 패턴 구현

자주 액세스하는 데이터를 빠르게 검색하기 위해 데이터를 캐싱하여 이점을 얻을 수 있는 액세스 패턴을 구현하세요.

일반적인 안티 패턴:

- 자주 변경되는 데이터를 캐시합니다.

- 캐시된 데이터가 마치 영구적으로 저장되고 항상 사용 가능한 것처럼 의존합니다.
- 캐시된 데이터의 일관성은 고려하지 않습니다.
- 캐싱 구현의 효율성을 모니터링하지 않습니다.

이 모범 사례 확립의 이점: 캐시에 데이터를 저장하면 읽기 지연 시간, 읽기 처리량, 사용자 경험 및 전반적인 효율성을 개선하고 비용을 절감할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

캐시는 동일한 데이터에 대한 향후 요청을 더 빠르고 효율적으로 처리할 수 있도록 데이터를 저장하는 것을 목적으로 하는 소프트웨어 또는 하드웨어 구성 요소입니다. 캐시에 저장된 데이터는 이전 계산을 반복하거나 다른 데이터 저장소에서 가져와서 손실된 경우 재구성할 수 있습니다.

데이터 캐싱은 전체 애플리케이션 성능을 개선하고 기본 데이터 소스의 부담을 줄이는 가장 효과적인 전략 중 하나가 될 수 있습니다. 데이터를 원격 직접 호출을 수행하는 애플리케이션 내에서(클라이언트 측 캐싱이라고 함) 또는 빠른 보조 서비스를 사용하여 데이터를 저장하는 등(원격 캐싱이라고 함) 애플리케이션의 여러 수준에서 데이터를 캐싱할 수 있습니다.

클라이언트 측 캐싱

클라이언트측 캐싱을 사용하면 각 클라이언트(백엔드 데이터 스토어를 쿼리하는 애플리케이션 또는 서비스)가 고유한 쿼리 결과를 지정된 시간 동안 로컬에 저장할 수 있습니다. 이렇게 하면 로컬 클라이언트 캐시를 먼저 확인하여 네트워크를 통해 데이터 스토어에 대한 요청 수를 줄일 수 있습니다. 결과가 없는 경우 애플리케이션은 데이터 스토어를 쿼리하고 해당 결과를 로컬에 저장할 수 있습니다. 이 패턴을 사용하면 각 클라이언트가 가능한 가장 가까운 위치(클라이언트 자체)에 데이터를 저장할 수 있으므로 지연 시간을 최소화할 수 있습니다. 또한 클라이언트는 백엔드 데이터 스토어를 사용할 수 없는 경우에도 일부 쿼리를 계속 제공하여 전체 시스템의 가용성을 높일 수 있습니다.

이 접근 방식의 한 가지 단점은 여러 클라이언트가 관련된 경우 동일한 캐시된 데이터를 로컬에 저장할 수 있다는 것입니다. 이로 인해 클라이언트 간에 중복 스토리지 사용과 데이터 불일치가 모두 발생합니다. 한 클라이언트가 쿼리 결과를 캐시하고 1분 후에 다른 클라이언트가 동일한 쿼리를 실행하여 다른 결과를 얻을 수 있습니다.

원격 캐싱

클라이언트 간 데이터 중복 문제를 해결하기 위해 빠른 외부 서비스 또는 원격 캐시를 사용하여 쿼리된 데이터를 저장할 수 있습니다. 각 클라이언트는 로컬 데이터스토어를 확인하는 대신 백엔드 데이터스토어를 쿼리하기 전에 원격 캐시를 확인합니다. 이 전략을 사용하면 클라이언트와 독립적으로 스토리

지 공간이 확장되므로 클라이언트 간 응답 일관성이 향상되고 저장된 데이터의 효율성이 향상되며 캐시된 데이터의 볼륨이 커집니다.

원격 캐시의 단점은 원격 캐시를 확인하기 위해 추가 네트워크 홉이 필요하기 때문에 전체 시스템의 지연 시간이 더 길어질 수 있다는 것입니다. 클라이언트 측 캐싱은 다중 레벨 캐싱을 위한 원격 캐싱과 함께 사용하여 지연 시간을 개선할 수 있습니다.

구현 단계

- 캐싱의 이점을 누릴 수 있는 데이터베이스, API 및 네트워크 서비스를 식별합니다. 읽기 워크로드가 많거나 쓰기 대 읽기 비율이 높거나 규모 조정 비용이 많이 드는 서비스는 캐싱의 대상입니다.
  - 데이터베이스 캐싱
  - 응답성 향상을 위한 API 캐싱 활성화
- 액세스 패턴에 가장 적합한 유형의 캐싱 전략을 식별하세요.
  - 캐싱 전략
  - AWS Caching Solutions
- 데이터 스토어에 대한 Caching Best Practices를 따릅니다.
- 데이터의 최신 상태를 유지하고 백엔드 데이터스토어에 대한 부담을 줄이는 모든 데이터에 대해 Time-To-Live(TTL)와 같은 캐시 무효화 전략을 구성하세요.
- 자동 연결 재시도, 지수 백오프, 클라이언트 측 시간 제한, 연결 풀링과 같은 기능을 클라이언트에서 활성화하여 성능과 신뢰성을 개선할 수 있습니다.
  - Best practices: Redis clients and Amazon ElastiCache (Redis OSS)
- 모니터 캐시 적중률을 80% 이상 목표로 합니다. 값이 낮으면 캐시 크기가 충분하지 않거나 액세스 패턴이 캐싱의 이점을 얻지 못한다는 의미일 수 있습니다.
  - Which metrics should I monitor?
  - Best practices for monitoring Redis workloads on Amazon ElastiCache
  - Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- 데이터 복제를 구현하여 여러 인스턴스로 읽기를 오프로드하고 데이터 읽기 성능과 가용성을 개선합니다.

리소스

관련 문서:

- Using the Amazon ElastiCache Well-Architected Lens

- Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- 어떤 메트릭을 모니터링해야 합니까?
- Performance at Scale with Amazon ElastiCache 백서
- 캐싱 관련 당면 과제 및 전략

관련 비디오:

- Amazon ElastiCache Learning Path
- Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2020 - Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2023 - [LAUNCH] Introducing Amazon ElastiCache Serverless
- AWS re:Invent 2022 - 5 great ways to reimagine your data layer with Redis
- AWS re:Invent 2021 - Deep dive on Amazon ElastiCache (Redis OSS)

관련 예제:

- Boosting MySQL database performance with Amazon ElastiCache (Redis OSS)

네트워킹 및 콘텐츠 전송

Questions

- PERF 4. 워크로드에서 네트워킹 리소스를 어떻게 선택하고 구성하나요?

PERF 4. 워크로드에서 네트워킹 리소스를 어떻게 선택하고 구성하나요?

워크로드에 대한 최적의 네트워킹 솔루션은 지연 시간, 처리량 요구 사항, 지터, 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

- PERF04-BP01 네트워킹이 성능에 미치는 영향 파악
- PERF04-BP02 사용 가능한 네트워킹 기능 평가
- PERF04-BP03 워크로드에 적합한 전용 연결 또는 VPN 선택

- PERF04-BP04 로드 밸런싱을 사용하여 여러 리소스에 트래픽 분산
- PERF04-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택
- PERF04-BP06 네트워크 요구 사항에 따라 워크로드의 위치 선택
- PERF04-BP07 지표를 기준으로 네트워크 구성 최적화

PERF04-BP01 네트워킹이 성능에 미치는 영향 파악

네트워크 관련 의사 결정이 워크로드에 미치는 영향을 분석하고 이해하여 효율적인 성능과 향상된 사용자 경험을 제공합니다.

일반적인 안티 패턴:

- 모든 트래픽이 기존 데이터 센터를 통과합니다.
- 클라우드 네이티브 네트워크 보안 도구를 사용하는 대신 중앙 방화벽을 통해 모든 트래픽을 라우팅합니다.
- 실제 사용 요구 사항을 이해하지 않고 AWS Direct Connect 연결을 프로비저닝합니다.
- 네트워킹 솔루션을 정의할 때 워크로드 특성과 암호화 오버헤드를 고려하지 않습니다.
- 클라우드의 네트워킹 솔루션에 온프레미스 개념과 전략을 적용합니다.

이 모범 사례 확립의 이점: 네트워킹이 워크로드 성능에 미치는 영향을 이해하면 잠재적인 병목 현상을 식별하고, 사용자 경험을 개선하며, 신뢰성을 높이고, 워크로드 변화에 따라 운영 유지 관리 작업을 줄이는 데 도움이 됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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 및 라우팅 규칙에도 영향을 받습니다.

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.

구현 단계:

- 네트워크 지연 시간, 대역폭, 프로토콜, 위치, 트래픽 패턴(급증 및 빈도), 처리량, 암호화, 검사 및 라우팅 규칙과 같은 지표를 포함한 네트워킹 성능 요구 사항을 정의하고 문서화하세요.
- VPC, AWS Direct Connect, Elastic Load Balancing(ELB), Amazon Route 53과 같은 주요 AWS 네트워킹 서비스에 대해 알아봅니다.
- 다음과 같은 주요 네트워킹 특성을 파악하세요.

특성	도구 및 지표
기본적인 네트워킹 특성	• VPC 흐름 로그 • AWS Transit Gateway 흐름 로그 • AWS Transit Gateway 지표 • AWS PrivateLink 지표
애플리케이션 네트워킹 특성	• Elastic Fabric Adapter(EFA) • AWS App Mesh 지표 • Amazon API Gateway 지표
엣지 네트워킹 특성	• Amazon CloudFront 지표 • Amazon Route 53 지표 • AWS Global Accelerator 지표
하이브리드 네트워킹 특성	• AWS Direct Connect 지표 • AWS Site-to-Site VPN 지표 • AWS Client VPN 지표 • AWS 클라우드 WAN 지표
보안 네트워킹 특성	• AWS Shield, AWS WAF, AWS Network Firewall 지표
추적 특성	• AWS X-Ray • VPC Reachability Analyzer • Network Access Analyzer • Amazon Inspector

특성	도구 및 지표
	• Amazon CloudWatch RUM

- 네트워크 성능을 벤치마킹하고 테스트합니다.
  - 네트워크 처리량을 벤치마킹합니다(인스턴스가 동일한 VPC에 있는 경우 일부 요인이 Amazon EC2 네트워크 성능에 영향을 미칠 수 있음). 동일한 VPC에 있는 Amazon EC2 Linux 인스턴스 간의 네트워크 대역폭을 측정합니다.
  - 로드 테스트를 수행하여 네트워킹 솔루션과 옵션을 실험합니다.

리소스

관련 문서:

- Application Load Balancer
- Linux 기반 EC2의 향상된 네트워킹
- Windows 기반 EC2의 향상된 네트워킹
- EC2 배치 그룹
- Linux 인스턴스에서 Elastic Network Adapter(ENA)로 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC 엔드포인트

관련 비디오:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - What can networking do for your application?
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2023 - A developer’s guide to cloud networking
- AWS re:Invent 2019 - Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2019 - Optimizing Network Performance for Amazon EC2 Instances
- AWS Summit Online - Improve Global Network Performance for Applications

- AWS re:Invent 2020 - Networking best practices and tips with the Well-Architected Framework
- AWS re:Invent 2020 - AWS networking best practices in large-scale migrations

관련 예제:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 워크숍
- Hands-on Network Firewall 워크숍
- Observing and Diagnosing your Network on AWS
- Finding and addressing Network Misconfigurations on AWS

PERF04-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, 네트워크 거리 또는 지터를 줄이는 데 사용할 수 있는 네트워크 수준 기능을 활용합니다.

일반적인 안티 패턴:

- 본사가 물리적으로 위치한 한 리전 내에 머무릅니다.
- 트래픽 필터링에 보안 그룹 대신 방화벽을 사용합니다.
- 보안 그룹, 엔드포인트 정책 및 기타 클라우드 네이티브 기능에 의존하지 않고 트래픽 검사를 위해 TLS를 중단합니다.
- 보안 그룹 대신 서브넷 기반 조각화만 사용합니다.

이 모범 사례 확립의 이점: 모든 서비스 기능 및 옵션을 평가하면 워크로드 성능을 개선하고 인프라 비용을 줄이며 워크로드를 유지 관리하는 데 필요한 작업을 줄이며 전반적인 보안 상태를 개선할 수 있습니다. 전 세계의 AWS 백본을 사용하여 고객에게 최상의 네트워킹 환경을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

AWS에서는 네트워크 성능을 개선하는 데 도움이 되는 AWS Global Accelerator 및 Amazon CloudFront와 같은 서비스를 제공하지만, 대부분의 AWS 서비스에는 네트워크 트래픽을 최적화하는 제품 기능(예: Amazon S3 Transfer Acceleration 기능)이 있습니다.

사용 가능한 네트워크 관련 구성 옵션과 이러한 옵션이 워크로드에 미치는 영향을 검토합니다. 성능 최적화는 이러한 옵션이 아키텍처와 상호 작용하는 방식과 측정된 성능 및 사용자 경험 모두에 미치는 영향을 이해하는 데 달려 있습니다.

구현 단계

- 워크로드 구성 요소 목록을 만듭니다.
  - 통합 글로벌 네트워크를 구축할 때 조직 네트워크를 구축, 관리 및 모니터링하를 데 AWS 클라우드 WAN 사용을 고려하세요.
  - Amazon CloudWatch Logs 지표로 글로벌 및 코어 네트워크를 모니터링합니다. 사용자의 디지털 경험을 식별하고, 이해하며, 개선하는 데 도움이 되는 인사이트를 제공하는 Amazon CloudWatch RUM을 활용하세요.
  - AWS Network Manager를 사용하여 애플리케이션 성능이 기본 AWS 네트워크의 성능과 어떤 관계가 있는지 파악할 수 있도록 AWS 리전 및 가용 영역 간 그리고 각 가용 영역 내부의 총 네트워크 지연 시간을 확인합니다.
  - 기존 구성 관리 데이터베이스(CMDB) 도구를 사용하거나 AWS Config 등과 같은 도구를 사용하여 워크로드 인벤토리 또는 구성 방식을 생성합니다.
- 기존 워크로드인 경우에는 성과 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.
- 새 워크로드인 경우 로드 테스트를 수행하여 성능 병목 현상을 식별합니다.
- 파악한 성능 병목 현상에 대해 솔루션의 구성 옵션을 검토하여 성능 개선 기회를 파악합니다. 다음과 같은 주요 네트워킹 옵션 및 기능을 확인합니다.

개선 기회	Solution
네트워크 경로 또는 라우트	Network Access Analyzer를 사용하여 경로나 라우트를 식별합니다.
네트워크 프로토콜	PERF04-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택 섹션 참조
네트워크 토폴로지	여러 계정을 연결할 때 VPC 피어링 및 AWS Transit Gateway 사이에서 운영 및 성능과 관련된 장단점을 평가합니다. AWS Transit

개선 기회	Solution
	Gateway에서는 수천 개의 AWS 계정에 걸쳐 있는 모든 VPC를 온프레미스 네트워크로 상호 연결하는 방법을 단순화합니다. AWS Resource Access Manager를 사용하여 여러 계정 사이에서 AWS Transit Gateway를 공유합니다. PERF04-BP03 워크로드에 적합한 전용 연결 또는 VPN 선택 섹션 참조
네트워크 서비스	AWS Global Accelerator는 AWS 글로벌 네트워크 인프라를 사용하여 사용자 트래픽의 성능을 최대 60%까지 개선하는 네트워킹 서비스입니다. Amazon CloudFront는 전 세계적으로 워크로드 콘텐츠 전송 성능 및 지연 시간을 개선할 수 있습니다. Lambda@Edge를 사용하여 CloudFront가 사용자에게 더 가까이 제공하는 콘텐츠를 사용자 지정하고, 지연 시간을 줄이며, 성능을 개선하는 함수를 실행합니다. Amazon Route 53은 지연 시간 기반 라우팅, 지리적 위치 라우팅, 지리 근접 라우팅 및 IP 기반 라우팅 옵션을 제공하여 전 세계 고객을 대상으로 워크로드 성능을 개선할 수 있습니다. 워크로드가 전 세계에 분산되어 있는 경우, 워크로드 트래픽 및 사용자 위치를 검토하여 어떤 라우팅 옵션이 워크로드 성능을 최적화하는지 파악합니다.

개선 기회	Solution
스토리지 리소스 기능	Amazon S3 Transfer Acceleration은 외부 사용자가 Amazon S3로 데이터를 업로드할 때 CloudFront의 네트워킹 최적화 이점을 활용할 수 있는 기능입니다. 이렇게 하면 AWS 클라우드 전용 연결을 사용할 수 없는 원격 위치에서 대량의 데이터를 전송할 수 있습니다. Amazon S3 다중 리전 액세스 포인트는 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.

개선 기회	Solution
컴퓨팅 리소스 기능	Amazon EC2 인스턴스, 컨테이너 및 Lambda 함수에서 사용하는 탄력적 네트워크 인터페이스(ENA)는 흐름 기준으로 제한됩니다. 배치 그룹을 검토하여 EC2 네트워킹 처리량을 최적화합니다. 흐름 기준에서 병목 현상을 방지하려면 여러 흐름을 사용하도록 애플리케이션을 설계합니다. 컴퓨팅 관련 네트워크 지표를 모니터링하고 이러한 지표에 대한 가시성을 얻으려면 CloudWatch 지표 및 ethtool을 사용합니다. `ethtool` 명령은 ENA 드라이버에 포함되어 있으며 사용자 지정 지표로 CloudWatch에 게시할 수 있는 추가 네트워크 관련 지표를 노출할 수 있습니다. Amazon Elastic Network Adapter(ENA)는 클러스터 배치 그룹 내에서 인스턴스에 대해 더 나은 처리량을 제공하여 한층 더 최적화합니다. Elastic Fabric Adapter(EFA)는 AWS에서 높은 수준의 대규모 노드 간 통신이 필요한 워크로드를 실행할 때 사용할 수 있는 Amazon EC2 인스턴스용 네트워크 인터페이스입니다. Amazon EBS 최적화 인스턴스는 최적화된 구성 스택을 사용하고 Amazon EBS I/O를 늘리기 위해 전용 용량을 추가로 제공합니다.

리소스

관련 문서:

- Application Load Balancer
- Linux 기반 EC2의 향상된 네트워킹
- Windows 기반 EC2의 향상된 네트워킹
- EC2 배치 그룹

- Linux 인스턴스에서 Elastic Network Adapter(ENA)로 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC 엔드포인트
- VPC 흐름 로그

관련 비디오:

- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2018 – Optimizing Network Performance for Amazon EC2 Instances
- AWS Global Accelerator

관련 예제:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 워크숍
- Observing and diagnosing your network
- Finding and addressing network misconfigurations on AWS

PERF04-BP03 워크로드에 적합한 전용 연결 또는 VPN 선택

온프레미스 및 클라우드 리소스를 연결하는 데 하이브리드 연결이 필요한 경우 성능 요구 사항을 충족할 수 있는 충분한 대역폭을 프로비저닝해야 합니다. 하이브리드 워크로드에 대한 대역폭 및 지연 시간 요구 사항을 예측합니다. 이 수치는 크기 조정 요구 사항을 결정합니다.

일반적인 안티 패턴:

- 네트워크 암호화 요구 사항에 대해서만 VPN 솔루션을 평가합니다.

- 백업 또는 이중화된 연결 옵션은 평가하지 않습니다.
- 모든 워크로드 요구 사항(암호화, 프로토콜, 대역폭 및 트래픽 요구 사항)을 식별할 수는 없습니다.

이 모범 사례 확립의 이점: 적절한 연결 솔루션을 선택하고 구성하면 워크로드의 신뢰성이 향상되고 성능이 극대화됩니다. 워크로드 요구 사항을 파악하고, 미리 계획하며, 하이브리드 솔루션을 평가하여 비용이 많이 드는 물리적 네트워크 변경과 운영 오버헤드를 최소화하는 동시에 가치 실현 시간을 단축할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

대역폭 요구 사항을 기반으로 하이브리드 네트워킹 아키텍처를 개발합니다. AWS Direct Connect에서는 AWS와 온프레미스 네트워크를 비공개로 연결할 수 있습니다. 일관된 성능을 달성하면서 고대역폭 그리고 짧은 지연 시간이 필요할 때 적합합니다. VPN 연결은 인터넷을 통해 보안 연결을 설정합니다. 임시 연결만 필요한 경우, 비용이 중요한 경우 또는 AWS Direct Connect를 사용 시 복원력이 뛰어난 물리적 네트워크 연결이 설정되기를 기다리는 동안 비상용으로 사용됩니다.

대역폭 요구 사항이 높으면 여러 AWS Direct Connect 또는 VPN 서비스를 고려할 수 있습니다. 트래픽은 서비스 간에 부하를 분산할 수 있지만 지연 시간과 대역폭 차이 때문에 AWS Direct Connect와 VPN 간의 부하 분산을 권장하지 않습니다.

구현 단계

- 기존 애플리케이션의 대역폭 및 지연 시간 요구 사항을 추정합니다.
  - AWS로 이동하는 기존 앱의 경우 내부 네트워크 모니터링 시스템의 데이터를 활용합니다.
  - 모니터링 데이터가 없는 새로운 또는 기존 워크로드의 경우 제품 소유자에게 문의하여 적절한 성과 지표를 결정하고 우수한 사용자 경험을 제공합니다.
- 연결 옵션으로 전용 연결 또는 VPN을 선택합니다. 모든 워크로드 요구 사항(암호화, 대역폭 및 트래픽 요구 사항)을 기반으로 AWS Direct Connect 또는 AWS VPN(또는 둘 다)을 선택할 수 있습니다. 다음 다이어그램은 적절한 연결 유형을 선택하는 데 도움이 됩니다.
  - AWS Direct Connect에서는 전용 연결 또는 호스팅된 연결을 사용하여 50Mbps에서 최대 100Gbps까지 AWS 환경에 대한 전용 연결을 제공합니다. 이렇게 하면 지연 시간을 관리 및 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, Link Aggregation Group(LAG) 또는 BGP Equal-Cost Multipath(ECMP)를 사용하여 확장된 Direct Connect 대역폭을 제공합니다.

  - AWS Site-to-Site VPN에서는 인터넷 프로토콜 보안(IPsec) 프로토콜을 지원하는 관리형 VPN 서비스를 제공합니다. VPN 연결이 생성되면 각 VPN 연결에는 고가용성을 위해 두 개의 터널이 포함됩니다.
- AWS 설명서에 따라 적절한 연결 옵션을 선택합니다.
  - AWS Direct Connect를 사용하기로 결정한 경우 연결에 적합한 대역폭을 선택합니다.
  - 여러 위치에서 AWS Site-to-Site VPN을 사용하여 AWS 리전에 연결하는 경우에는 가속화된 Site-to-Site VPN 연결을 사용해 네트워크 성능을 개선하세요.
  - 네트워크 설계가 AWS Direct Connect에서 IPSec VPN 연결로 구성된 경우 보안을 강화하고 세분화를 달성하기 위해 프라이빗 IP VPN 사용을 고려하세요. AWS Site-to-Site 프라이빗 IP VPN은 전송 가상 인터페이스(VIF) 위에 배포됩니다.
  - AWS Direct Connect SiteLink를 사용하면 AWS Direct Connect 위치 사이에서 AWS 리전을 우회하는 가장 빠른 경로로 데이터를 전송해 전 세계 데이터 센터 간에 지연 시간이 짧고 중복 연결을 생성합니다.
- 프로덕션에 배포하기 전에 연결 설정을 확인하세요. 보안 및 성능 테스트를 수행하여 대역폭, 신뢰성, 지연 시간 및 규정 준수 요구 사항을 충족하는지 확인하세요.
- 연결 성능 및 사용량을 정기적으로 모니터링하고 필요한 경우 최적화하세요.

결정론적 성능 흐름도

리소스

관련 문서:

- AWS의 네트워킹 제품
- AWS Transit Gateway
- VPC Endpoints
- 확장 가능하고 안전한 다중 VPC AWS 네트워크 인프라 구축
- Client VPN

관련 비디오:

- AWS re:Invent 2023 – Building hybrid network connectivity with AWS

- AWS re:Invent 2023 – Secure remote connectivity to AWS
- AWS re:Invent 2022 – Optimizing performance with Amazon CloudFront
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2020 – AWS Transit Gateway Connect

관련 예제:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 워크숍

PERF04-BP04 로드 밸런싱을 사용하여 여러 리소스에 트래픽 분산

클라우드의 탄력성을 워크로드에 활용할 수 있도록 여러 리소스 또는 서비스에 트래픽을 분산합니다. 로드 밸런싱을 사용하여 암호화 종료를 오프로드하면 성능과 신뢰성을 개선하고 트래픽을 효율적으로 관리 및 라우팅할 수 있습니다.

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.
- 기존 로드 밸런서를 통해 모든 인터넷 트래픽을 라우팅합니다.
- 일반 TCP 로드 밸런싱을 사용하고 각 컴퓨팅 노드에서 SSL 암호화를 처리하도록 합니다.

이 모범 사례 확립의 이점: 로드 밸런서는 단일 가용 영역 또는 여러 가용 영역에서 애플리케이션 트래픽의 다양한 부하를 처리하고 고가용성, 자동 규모 조정, 워크로드 활용도 향상을 지원합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

로드 밸런서는 워크로드의 진입점 역할을 하며 여기에서 트래픽을 컴퓨팅 인스턴스나 컨테이너와 같은 백엔드 대상으로 분산하여 사용률을 개선합니다.

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), 애플

리케이션 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.

AWS는 애플리케이션이 로드 밸런싱을 사용할 수 있도록 여러 모델을 제공합니다. Application Load Balancer는 HTTP 및 HTTPS 트래픽을 로드 밸런싱하는 데 가장 적합하며, 마이크로서비스 및 컨테이너를 비롯한 최신 애플리케이션 아키텍처를 제공할 때 사용할 수 있는 고급 요청 라우팅 기능을 제공합니다.

Network Load Balancer는 성능이 매우 우수해야 하는 TCP 트래픽 로드 밸런싱을 수행하려는 경우에 사용하면 가장 효율적입니다. 또한 지연 시간을 매우 짧게 유지하면서 초당 수백만 개의 요청을 처리할 수 있으며, 예상치 못한 휘발성 트래픽 패턴도 처리할 수 있도록 최적화되어 있습니다.

Elastic Load Balancing이 제공하는 통합 인증서 관리 및 SSL/TLS 복호화를 활용하면 로드 밸런서의 SSL 설정을 중앙에서 유연하게 관리하고 CPU 집약적인 작업을 워크로드에서 오프로드할 수 있습니다.

적절한 로드 밸런서를 선택한 후 해당 기능을 활용하면 백엔드가 트래픽을 처리하는 데 필요한 노력을 줄일 수 있습니다.

예를 들어 Application Load Balancer(ALB) 및 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 확보하며 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.

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 또는 AWS Outposts에서 Application Load Balancer를 활용하여 워크로드를 고객에게 더 가까이 가져오기로 결정할 수 있습니다.

지연 시간에 민감한 워크로드에 대한 또 다른 대응책은 크로스 영역 로드 밸런싱입니다. 크로스 영역 로드 밸런싱을 활용하면 각 로드 밸런서 노드를 사용하도록 설정된 모든 가용 영역의 등록된 대상에 트래픽을 분산합니다.

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. 로드 밸런서는 컨테이너식 워크로드를 위해 Amazon ECS 및 Amazon EKS와 통합할 수도 있습니다.

- Amazon ECS - 서비스 로드 밸런싱
- Amazon EKS에서의 애플리케이션 로드 밸런싱
- Amazon EKS의 네트워크 로드 밸런싱

구현 단계

- 트래픽 볼륨, 가용성 및 애플리케이션 확장성을 포함한 로드 밸런싱 요구 사항을 정의합니다.
- 애플리케이션에 적합한 로드 밸런서 유형을 선택하세요.
  - HTTP/HTTPS 워크로드에 대해 Application Load Balancer를 사용합니다.
  - TCP 또는 UDP에서 실행되는 비HTTP 워크로드에 대해 Network Load Balancer를 사용합니다.
  - 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 AWS PrivateLink에 HTTP 워크로드를 노출하려는 경우 이를 수행할 수 있습니다.
  - 로드 밸런서의 전체 비교를 위해서 ELB 제품 비교를 참조하세요.
- 가능하면 SSL/TLS 오프로딩을 사용하세요.
  - AWS Certificate Manager와 통합된 Application Load Balancer 및 Network Load Balancer 모두에서 HTTPS/TLS 리스너를 구성합니다.
  - 일부 워크로드는 규정 준수상의 이유로 엔드 투 엔드 암호화가 필요할 수 있습니다. 이 경우 대상에서 암호화를 사용하도록 설정해야 합니다.
  - 보안 모범 사례는 SEC09-BP02 전송 중 암호화 적용을 참조하세요.
- 적절한 라우팅 알고리즘(ALB만)을 선택합니다.
  - 라우팅 알고리즘에 따라 백엔드 대상에서의 활용도 및 성능에 미치는 영향에 차이를 만들 수 있습니다. 예를 들어 ALB는 라우팅 알고리즘에 대한 두 가지 옵션을 제공합니다.
  - 미해결 요청 최소화: 애플리케이션에 대한 요청의 복잡성이 다양하거나 대상의 처리 능력이 다양한 경우 백엔드 대상에 로드를 더 효율적으로 분산하기 위해 사용합니다.
  - 라운드 로빈: 요청과 대상이 유사하거나 대상 간에 요청을 균등하게 분산해야 하는 경우에 사용합니다.
- 크로스 영역 또는 영역 격리를 고려합니다.

  - 지연 시간 개선 및 영역 장애 도메인을 위해 크로스 영역 끔(영역 격리)을 사용합니다. 이 기능은 NLB에서 기본적으로 꺼져 있으며 ALB의 경우 대상 그룹별로 해제할 수 있습니다.
  - 가용성과 유연성 향상을 위해 크로스 영역을 사용합니다. 기본적으로 크로스 영역은 ALB에서 활성화되어 있으며 NLB의 경우 대상 그룹별로 활성화할 수 있습니다.
- HTTP 워크로드(ALB만)에 대해 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.
- 로드 밸런서에 대한 모니터링을 켜세요.
  - Application Load Balancer 및 Network Load Balancer의 액세스 로그를 활성화합니다.
  - ALB에 대해 고려해야 할 주요 필드는 `request_processing_time`, `request_processing_time`, `response_processing_time`입니다.
  - NLB에 대해 고려해야 할 주요 필드는 `connection_time` 및 `tls_handshake_time`입니다.
  - 필요할 때 로그 쿼리를 준비합니다. Amazon Athena를 사용하여 ALB 로그와 NLB 로그를 모두 쿼리할 수 있습니다.
  - ALB의 `TargetResponseTime`과 같은 성능 관련 지표에 대한 경보를 생성합니다.

리소스

관련 문서:

- ELB 제품 비교
- AWS 글로벌 인프라
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Application Load Balancer 로그 쿼리
- Monitor your Application Load Balancers
- Monitor your Network Load Balancer
- Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group

관련 비디오:

- AWS re:Invent 2023: What can networking do for your application?

- AWS re:Inforce 20: How to use Elastic Load Balancing to enhance your security posture at scale
- AWS re:Invent 2018: Elastic Load Balancing: Deep Dive and Best Practices
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads

관련 예제:

- Gateway Load Balancer
- CDK and AWS CloudFormation samples for Log Analysis with Amazon Athena

PERF04-BP05 성능을 개선할 수 있는 네트워크 프로토콜 선택

워크로드 성능에 미치는 영향을 기준으로 시스템과 네트워크 간의 통신에 사용할 프로토콜을 결정합니다.

원하는 처리량을 달성하려면 지연 시간과 대역폭 간의 관계를 고려해야 합니다. 파일 전송이 전송 제어 프로토콜(TCP)을 사용하는 경우 지연 시간이 길수록 전체 처리량이 줄어들 가능성이 높습니다. 이 문제는 TCP 튜닝 및 최적화된 전송 프로토콜을 사용하여 해결되지만 한 가지 해결 방법은 사용자 데이터그램 프로토콜(UDP)을 사용하는 것입니다.

일반적인 안티 패턴:

- 성능 요구 사항과 관계없이 모든 워크로드에 TCP를 사용합니다.

이 모범 사례 확립의 이점: 사용자와 워크로드 구성 요소 간의 통신에 적절한 프로토콜이 사용되는지 확인하면 애플리케이션의 전반적인 사용자 경험을 개선하는 데 도움이 됩니다. 예를 들어, 연결 없는 UDP는 빠른 속도를 허용하지만 재전송 기능 또는 높은 신뢰성을 제공하지 않습니다. TCP는 모든 기능을 갖춘 프로토콜이지만 패킷 처리에 더 많은 오버헤드가 필요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

애플리케이션에 맞는 다양한 프로토콜을 선택할 수 있고 이 분야에 대한 전문 지식이 있다면 다른 프로토콜을 사용하여 애플리케이션과 최종 사용자 경험을 최적화하세요. 이 접근 방식은 상당히 어려우므로 먼저 다른 방법으로 애플리케이션을 최적화한 경우에만 시도해야 합니다.

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 구성 요소 간 통신하기 위한 TCP 소켓을 열어줍니다. 이메일 및 파일 데이터 전송은 TCP를 사용하는 일반적인 애플리케이션입니다. TCP는 애플리케이션 구성 요소 간의 간단하고 안정적인 전송 메커니즘이기 때문입니다. TCP와 함께 TLS를 사용하면 통신에 약간의 오버헤드가 추가되어 대기 시간이 증가하고 처리량이 감소할 수 있지만 보안상의 이점이 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 연결이 필요 없는 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. 데이터그램 전송 계층 보안(DTLS)은 전송 계층 보안 전송 계층 보안(TLS)과 동일한 UDP입니다. 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

SRD(Scalable Reliable Datagram)는 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

- 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.
- 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, 연결 설정 시간을 제거하도록 동적 콘텐츠에 대한 각 원본에 영구 연결을 설정하도록 Amazon CloudFront를 사용하는 것이 좋습니다. 그렇지 않으면 각 클라이언트 요청이 느려집니다.

- TCP 또는 UDP와 함께 TLS를 사용하면 암호화 및 복호화의 영향으로 인해 지연 시간이 증가하고 워크로드에 대한 처리량이 감소합니다. 이러한 워크로드의 경우, 백엔드 인스턴스 대신, 로드 밸런서가 SSL/TLS 암호화 및 복호화 프로세스를 처리하도록 허용하여 워크로드 성능을 개선하기 위해 Elastic Load Balancing에서 SSL/TLS 오프로딩을 고려하세요. 이를 통해 백엔드 인스턴스의 CPU 사용률을 줄여 성능을 향상시키고 용량을 늘릴 수 있습니다.
- Network Load Balancer(NLB)를 사용하여 인증 및 권한 부여, 로깅, DNS, IoT, 스트리밍 미디어 등 UDP 프로토콜에 의존하는 서비스를 배포하세요. NLB는 여러 대상에 걸쳐 수신되는 UDP 트래픽을 배포하여 워크로드를 수평 확장하고 용량을 늘리고 단일 대상의 오버헤드를 줄일 수 있습니다.
- 고성능 컴퓨팅(HPC) 워크로드의 경우 Elastic Network Adapter(ENA) Express 기능 사용을 고려하세요. 이 기능은 SRD 프로토콜을 사용하여 EC2 인스턴스 간의 네트워크 트래픽에 대해 더 높은 단일 흐름 대역폭(25Gbps) 및 더 짧은 테일 지연 시간(99.9 백분위수)을 제공함으로서 네트워크 성능을 개선합니다.
- Application Load Balancer(ALB)를 사용하여 워크로드 구성 요소 간에 또는 gRPC 지원 클라이언트와 서비스 간에 gRPC(원격 프로시저 호출) 트래픽을 라우팅하고 로드를 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:

- How to route UDP traffic into Kubernetes
- Application Load Balancer
- Linux 기반 EC2의 향상된 네트워킹
- Windows 기반 EC2의 향상된 네트워킹
- EC2 배치 그룹
- Linux 인스턴스에서 Elastic Network Adapter(ENA)로 향상된 네트워킹 활성화
- Network Load Balancer
- AWS의 네트워킹 제품

- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC 엔드포인트

관련 비디오:

- AWS re:Invent 2022 – Scaling network performance on next-gen Amazon Elastic Compute Cloud instances
- AWS re:Invent 2022 – Application networking foundations

관련 예제:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking 워크숍

PERF04-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.

일반적인 안티 패턴:

- 모든 워크로드 리소스를 하나의 지리적 위치로 통합합니다.
- 워크로드 최종 사용자가 아니라 본인과 가장 가까운 리전을 선택했습니다.

이 모범 사례 확립의 이점: 사용자 경험은 사용자와 애플리케이션 간의 지연 시간에 크게 영향을 받습니다. 적절한 AWS 리전 및 AWS 프라이빗 글로벌 네트워크를 사용하면 대기 시간을 줄이고 원격 사용자에게 더 나은 경험을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Amazon EC2 인스턴스와 같은 리소스는 AWS 리전의 가용 영역, 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 서비스를 사용하여 엣지 로케이션의 콘텐츠를 캐싱하거나 AWS 글로벌 네트워크를 통해 워크로드에 대한 최적의 경로를 사용자에게 제공함으로써 네트워크 성능을 향상시킬 수 있습니다.

Amazon EC2에서는 네트워킹용 배치 그룹을 제공합니다. 배치 그룹은 지연 시간을 줄이기 위한 인스턴스의 논리적 그룹입니다. 지원되는 인스턴스 유형이 포함된 배치 그룹과 Elastic Network Adapter(ENA)를 사용하면 지연 시간이 짧고 지터가 감소된 25Gbps 네트워크에 워크로드를 연결할 수 있습니다. 네트워크 지연 시간이 짧거나 처리량이 높은 경우 또는 두 조건을 모두 충족하는 경우 성능이 개선되는 워크로드에는 배치 그룹을 사용하는 것이 좋습니다.

지연 시간에 민감한 서비스는 Amazon CloudFront와 같은 AWS 글로벌 네트워크를 사용하여 엣지 로케이션에서 제공됩니다. 이러한 엣지 로케이션에서는 보통 콘텐츠 전송 네트워크(CDN) 및 도메인 이름 시스템(DNS)과 같은 서비스를 제공합니다. 엣지에서 이러한 서비스를 사용함으로써 워크로드 지연 시간을 짧게 유지하면서 콘텐츠 또는 DNS 확인 요청에 응답할 수 있습니다. 이러한 서비스는 지리적 콘텐츠 타게팅(최종 사용자의 위치를 기준으로 각기 다른 콘텐츠 제공) 등의 지리적 서비스나 최종 사용자를 가장 가까운 리전으로 라우팅하는 지연 시간 기반 라우팅(지연 시간이 최소화됨)도 제공합니다.

지연 시간을 줄이고 콘텐츠 캐싱을 활성화하려면 엣지 서비스를 사용합니다. 이러한 방식의 이점을 최대한 활용하려면 DNS 및 HTTP/HTTPS용으로 캐시 제어를 올바르게 구성하세요.

구현 단계

- 네트워크 인터페이스를 오가는 IP 트래픽에 대한 정보를 캡처합니다.
  - VPC 흐름 로그를 사용하여 IP 트래픽 로깅
  - How the client IP address is preserved in AWS Global Accelerator
- 워크로드의 네트워크 액세스 패턴을 분석하여 사용자가 애플리케이션을 사용하는 방법을 식별합니다.
  - Amazon CloudWatch 및 AWS CloudTrail과 같은 모니터링 도구를 사용하여 네트워크 활동에 대한 데이터를 수집합니다.
  - 데이터를 분석하여 네트워크 액세스 패턴을 식별합니다.
- 다음과 같은 주요 요소를 토대로 하여 워크로드 배포용 리전을 선택합니다.
  - 데이터 위치: 데이터를 많이 사용하는 애플리케이션의 경우(예: 빅 데이터 및 기계 학습) 애플리케이션 코드는 최대한 데이터와 가까운 위치에서 실행되어야 합니다.
  - 사용자 위치: 사용자가 직접 사용하는 애플리케이션의 경우 워크로드의 사용자와 가까운 하나 이상의 리전을 선택합니다.
  - 기타 제약 조건: What to Consider when Selecting a Region for your Workloads에 나와 있는 비용 및 규정 준수 등 제약 요건을 고려합니다.
- AWS 로컬 영역을 사용하여 비디오 렌더링과 같은 워크로드를 실행합니다. 로컬 영역에서는 최종 사용자와 가까운 위치에 컴퓨팅 및 스토리지 리소스를 배치함으로써 이점을 얻을 수 있습니다.

- 온프레미스에 남아 있어야 하는 워크로드 및 해당 워크로드를 AWS의 나머지 워크로드와 함께 원활하게 실행하려는 워크로드에 대해서는 AWS Outposts를 사용합니다.
- 고해상도 라이브 비디오 스트리밍, 고음질 오디오 및 증강 현실 또는 가상 현실(AR/VR)과 같은 5G 디바이스용 애플리케이션은 지연 시간이 매우 짧아야 합니다. 이러한 애플리케이션에서는 AWS Wavelength를 고려합니다. AWS Wavelength에서는 5G 네트워크 내에 AWS 컴퓨팅 및 스토리지 서비스를 포함하여 지연 시간이 짧은 애플리케이션을 개발, 배포 및 확장하기 위한 모바일 엣지 컴퓨팅 인프라를 제공합니다.
- 자주 사용하는 자산에 로컬 캐싱 또는 AWS Caching Solutions를 사용하여 성능을 개선하고, 데이터 이동을 줄이며, 환경에 미치는 영향을 줄입니다.

Service	사용해야 하는 경우
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.

Service	사용해야 하는 경우
Lambda@Edge	객체가 캐시에 없는 경우 시작되는 컴퓨팅 집약적 작업에 사용합니다.
Amazon CloudFront Functions	HTTP(s) 요청 또는 응답 조작 등과 같이 단기 기능으로 실행할 수 있는 간단한 사용 사례에 사용합니다.
AWS IoT Greengrass	커넥티드 디바이스를 위한 로컬 컴퓨팅, 메시징 및 데이터 캐시를 실행하는 데 사용합니다.

- 일부 애플리케이션은 첫 번째 바이트까지의 지연 시간과 지터를 줄이고 처리량을 늘려 고정 진입 지점 또는 그 이상의 성능이 필요합니다. 이러한 애플리케이션은 엣지 로케이션에서 정적 애

니캐스트 IP 주소 및 TCP 종료를 제공하는 네트워킹 서비스를 활용할 수 있습니다. AWS Global Accelerator는 애플리케이션의 성능을 최대 60%까지 향상시키고 다중 리전 아키텍처에 빠른 장애 조치를 제공합니다. AWS Global Accelerator에서는 하나 이상의 AWS 리전에 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 가장 가까운 AWS 엣지 로케이션 간에 TCP 연결을 설정하여 초기 연결 설정 시간을 줄입니다. AWS Global Accelerator의 사용을 검토하여 TCP/UDP 워크로드의 성능을 향상시키고 다중 리전 아키텍처에 빠른 장애 조치를 제공합니다.

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 비즈니스 요구 사항과 지속 가능성 목표를 기준으로 리전 선택
- SUS02-BP04 네트워킹 요구 사항에 따라 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
- 배치 그룹
- AWS 로컬 영역
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN

- Amazon Route 53

관련 비디오:

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

관련 예제:

- AWS Global Accelerator Custom Routing 워크숍
- Handling Rewrites and Redirects using Edge Functions

PERF04-BP07 지표를 기준으로 네트워크 구성 최적화

수집 및 분석된 데이터가 제공하는 정보를 사용하여 네트워크 구성 최적화를 결정합니다.

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

VPC, 서브넷 또는 네트워크 인터페이스를 오가는 트래픽을 모니터링하는 것은 AWS 네트워크 리소스를 활용하는 방법과 네트워크 구성을 최적화하는 방법을 이해하는 데 중요합니다. 다음 AWS 네트워킹 도구를 사용하면 트래픽 사용, 네트워크 액세스 및 로그에 대한 정보를 추가로 검사할 수 있습니다.

구현 단계

- 수집할 대기 시간 또는 패킷 손실과 같은 핵심 성과 지표를 식별합니다. AWS는 이러한 지표를 수집하는 데 도움이 되는 몇 가지 도구를 제공합니다. 다음 도구를 사용하면 트래픽 사용, 네트워크 액세스 및 로그에 대한 정보를 추가로 검사할 수 있습니다.

AWS 도구	사용 장소
Amazon VPC IP 주소 관리자.	IPAM을 사용하여 AWS 및 온프레미스 워크로드를 위해 IP 주소를 계획, 추적 및 모니터링할 수 있습니다. 이는 IP 주소 사용 및 할당을 최적화하는 모범 사례입니다.
VPC 흐름 로그	VPC 흐름 로그를 사용하여 VPC의 네트워크 인터페이스에서 전송되고 수신되는 IP 트래픽에 대한 정보를 캡처합니다. VPC 흐름 로그를 사용하면 지나치게 제한적이거나 허용되는 보안 그룹 규칙을 진단하고 네트워크 인터페이스와의 트래픽 방향을 결정할 수 있습니다.
AWS Transit Gateway 흐름 로그	AWS Transit Gateway 흐름 로그를 사용하여 전송 게이트웨이를 오가는 IP 트래픽에 대한 정보를 캡처하세요.
DNS 쿼리 로깅	Route 53에서 수신하는 퍼블릭 또는 프라이빗 DNS 쿼리에 대한 정보를 기록합니다. DNS 로그를 사용하면 요청된 도메인 또는 하위 도메인 또는 DNS 쿼리에 응답한 Route 53 엣지 로케이션을 이해하여 DNS 구성을 최적화할 수 있습니다.
Reachability Analyzer	Reachability Analyzer를 사용하여 네트워크 도달 가능성을 분석하고 디버깅합니다.

AWS 도구	사용 장소
	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.
Network Access Analyzer	Network Access Analyzer를 사용하면 리소스에 대한 네트워크 액세스를 파악할 수 있습니다. 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.
Amazon CloudWatch	Amazon CloudWatch를 사용하고 네트워크 옵션에 적합한 지표를 켭니다. 워크로드에 적합한 네트워크 지표를 선택해야 합니다. 예를 들어 VPC 네트워크 주소 사용, 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.

AWS 도구	사용 장소
AWS Network Manager	AWS Network Manager를 사용하면 운영 및 계획 목적으로 AWS 글로벌 네트워크의 실시간 및 과거 성능을 모니터링할 수 있습니다. Network Manager에서는 AWS 리전 및 가용 영역 간 그리고 각 가용 영역 내에서 집계된 네트워크 지연 시간을 제공하므로 애플리케이션 성능이 기본 AWS 네트워크의 성능과 어떻게 연관되는지 더 잘 이해할 수 있습니다.
Amazon CloudWatch RUM	Amazon CloudWatch RUM을 사용하여 사용자 경험을 식별, 이해 및 개선하는 데 도움이 되는 인사이트를 제공하는 지표를 수집하세요.

- VPC 및 AWS Transit Gateway 흐름 로그를 사용하여 상위 발화자와 애플리케이션 트래픽 패턴을 식별합니다.
- VPC, 서브넷, 라우팅을 포함한 현재 네트워크 아키텍처를 평가하고 최적화합니다. 예를 들어, 다양한 VPC 피어링 또는 AWS Transit Gateway가 아키텍처의 네트워킹을 개선하는 데 어떻게 도움이 되는지 평가할 수 있습니다.
- 네트워크의 라우팅 경로를 평가하여 대상 간 최단 경로가 항상 사용되는지 확인하세요. Network Access Analyzer는 이를 수행하는 데 도움이 됩니다.

리소스

관련 문서:

- Public DNS query logging
- IPAM이란?
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- VPC의 CloudWatch 지표
- Apache Parquet 형식의 VPC 흐름 로그를 사용하여 네트워크 분석 성능을 최적화하고 비용을 절감합니다.
- Monitoring your global and core networks with Amazon CloudWatch metrics
- Continuously monitor network traffic and resources

관련 비디오:

- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2020 – Networking best practices and tips with the AWS Well-Architected Framework
- AWS re:Invent 2020 – Monitoring and troubleshooting network traffic

관련 예제:

- AWS Networking 워크숍
- AWS Network Monitoring
- Observing and diagnosing your network on AWS
- Finding and addressing network misconfigurations on AWS

프로세스 및 문화

Questions

- PERF 5. 조직의 관행과 문화가 워크로드의 성과 효율성에 어떻게 기여하나요?

PERF 5. 조직의 관행과 문화가 워크로드의 성과 효율성에 어떻게 기여하나요?

워크로드를 설계할 때는 효율적인 고성능 클라우드 워크로드를 더 잘 실행하는 데 도움이 되도록 채택할 수 있는 원칙과 관행이 있습니다. 클라우드 워크로드의 성능 효율성을 촉진하는 문화를 채택하려면 다음과 같은 주요 원칙과 관행을 고려합니다.

모범 사례

- PERF05-BP01 워크로드 상태 및 성능을 측정하기 위한 핵심 성과 지표(KPI) 수립
- PERF05-BP02 모니터링 솔루션을 사용하여 성능이 가장 중요한 영역 파악
- PERF05-BP03 워크로드 성능 개선을 위한 프로세스 정의
- PERF05-BP04 워크로드 로드 테스트
- PERF05-BP05 자동화를 사용하여 성능 관련 문제 사전 해결

- PERF05-BP06 워크로드 및 서비스 최신 상태 유지
- PERF05-BP07 정기적으로 지표 검토

PERF05-BP01 워크로드 상태 및 성능을 측정하기 위한 핵심 성과 지표(KPI) 수립

워크로드 성능을 양적 및 질적으로 측정하는 KPI를 식별합니다. KPI는 비즈니스 목표와 관련된 워크로드의 상태와 성과를 측정하는 데 도움이 됩니다.

일반적인 안티 패턴:

- 시스템 수준 지표만 모니터링하여 워크로드에 대한 인사이트를 얻고, 해당 지표에 대한 비즈니스 영향을 이해하지 못합니다.
- KPI가 이미 표준 지표 데이터로 게시 및 공유되고 있다고 가정합니다.
- 정량적이고 측정 가능한 KPI를 정의하지 않습니다.
- KPI를 비즈니스 목표나 전략에 맞추지 않습니다.

이 모범 사례 확립의 이점: 워크로드 상태 및 성능을 나타내는 특정 KPI를 식별하면 팀의 우선순위를 조정하고 성공적인 비즈니스 결과를 정의하는 데 도움이 됩니다. 이러한 지표를 모든 부서와 공유하면 임곗값, 기대치 및 비즈니스에 미치는 영향을 파악하고, 이에 따른 조정이 가능해집니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

KPI를 통해 비즈니스 및 엔지니어링 팀은 목표 측정과 전략 그리고 이러한 요인을 조합하여 비즈니스 성과를 도출하는 방법에 대해 협의할 수 있습니다. 예를 들어, 웹 사이트 워크로드에는 전체 성능을 나타내는 지표로 페이지 로드 시간을 사용할 수 있습니다. 이 지표는 사용자 경험을 측정하는 여러 데이터 포인트 중 하나입니다. 페이지 로드 시간 임곗값을 파악하는 것 말고도 이상적인 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 임곗값을 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 모두 고려해서 KPI를 설정해야 합니다.

팀은 참조용으로 실시간 세분화된 데이터와 기록 데이터를 사용하여 워크로드 KPI를 평가하고, KPI 데이터에 대한 지표 산술을 수행하여 운영 및 사용률 인사이트를 도출하는 대시보드를 만들어야 합니다. KPI는 문서화되어야 하며 비즈니스 목표와 전략을 지원하는 임곗값을 포함해야 하고 모니터링 중인 지

표에 매핑되어야 합니다. 비즈니스 목표, 전략 또는 최종 사용자 요구 사항이 변경되면 KPI를 다시 검토해야 합니다.

구현 단계

- 이해관계자 식별: 개발 및 운영 팀을 비롯한 주요 비즈니스 이해관계자를 식별하고 문서화합니다.
- 목표 정의: 이러한 이해관계자들과 협력하여 워크로드의 목표를 정의하고 문서화합니다. 처리량, 응답 시간, 비용 등 워크로드 성능의 중요한 측면과 사용자 만족도와 같은 비즈니스 목표를 고려합니다.
- 업계 모범 사례 검토: 업계 모범 사례를 검토하여 워크로드 목표에 부합하는 관련 KPI를 파악합니다.
- 지표 식별: 워크로드 목표에 부합하며 성능 및 비즈니스 목표를 측정하는 데 도움이 될 수 있는 지표를 식별합니다. 이러한 지표를 기반으로 KPI를 설정합니다. 지표의 예로는 평균 응답 시간이나 동시 사용자 수와 같은 측정값이 있습니다.
- KPI 정의 및 문서화: 업계 모범 사례와 워크로드 목표를 사용하여 워크로드 KPI의 목표를 설정합니다. 이 정보를 사용하여 심각도 또는 경보 수준에 대한 KPI 임곗값을 설정합니다. KPI가 충족되지 않을 경우의 위험과 영향을 식별하고 문서화합니다.
- 모니터링 구현: Amazon CloudWatch 또는 AWS Config와 같은 모니터링 도구를 사용하여 지표를 수집하고 KPI를 측정합니다.
- KPI 시각적 전달: KPI를 시각화하고 이해관계자에게 전달하기 위한 Amazon QuickSight와 같은 대시보드 도구를 사용합니다.
- 분석 및 최적화: 정기적으로 KPI를 검토하고 분석하여 개선이 필요한 워크로드 영역을 파악합니다. 이해관계자와 협력하여 이러한 개선 사항을 구현합니다.
- 보관 및 개선: 지표와 KPI를 정기적으로 검토하여 효과를 평가합니다. 이는 특히 비즈니스 목표나 워크로드 성능이 변경되는 경우에는 더욱 중요합니다.

리소스

관련 문서:

- CloudWatch 설명서
- 모니터링, 로깅 및 성능 AWS Partner
- AWS observability tools
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations
- How to track your cost optimization KPIs with the KPI Dashboard
- X-Ray 설명서

- Amazon CloudWatch 대시보드 사용
- QuickSight KPIs

관련 비디오:

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health
- AWS re:Invent 2023 - Performance & efficiency at Pinterest: Optimizing the latest instances
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Creating an Effective Metrics Strategy for Your Business | AWS Events

관련 예제:

- Creating a dashboard with QuickSight

PERF05-BP02 모니터링 솔루션을 사용하여 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.

일반적인 안티 패턴:

- CPU 사용률 또는 메모리 압력과 같은 표준 컴퓨팅 지표로 성능 문제를 파악하기에 충분하다고 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 위험 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.

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 조사한 후에는 이러한 성능 문제를 해결하기 위해 어떤 솔루션을 배포할 수 있는지 살펴보세요.

구현 단계

- 엔드 투 엔드 모니터링을 설정하여 모든 워크로드 구성 요소 및 지표를 캡처합니다. 다음은 AWS에 대한 모니터링 솔루션의 예입니다.

Service	사용 장소
Amazon CloudWatch 실제 사용자 모니터링 (RUM)	실제 사용자 클라이언트 측 및 프론트엔드 세션에서 애플리케이션 성과 지표를 포착합니다.
AWS X-Ray	애플리케이션 계층을 통해 트래픽을 추적하고 구성 요소와 종속성 간의 지연 시간을 식별합니다. X-Ray 서비스 맵을 사용하여 워크로드 구성 요소 간 관계 및 지연 시간을 확인합니다.
Amazon Relational Database Service 성능 개선 도우미	데이터베이스 성과 지표를 보고 성능 향상을 식별합니다.
Amazon RDS 향상된 모니터링	데이터베이스 OS 성과 지표를 확인합니다.
Amazon DevOps Guru	비정상적인 운영 패턴을 감지하여 고객에게 영향을 미치기 전에 운영 문제를 식별할 수 있습니다.

- 지표를 생성하고 트래픽 패턴, 병목 현상 및 중요한 성능 영역을 파악하기 위한 테스트를 수행합니다. 다음은 테스트를 수행하는 방법에 대한 몇 가지 예입니다.

  - 시간이 지남에 따라 일관된 지표를 생성하기 위해 Linux 크론 작업 또는 rate 표현식을 사용하여 프로그래밍 방식으로 브라우저 기반 사용자 활동을 모방하도록 CloudWatch Synthetic Canary를 설정합니다.
  - AWS 분산 로드 테스트 솔루션을 사용하여 피크 트래픽을 생성하거나 예상되는 증가율로 워크로드를 테스트합니다.
- 지표 및 원격 측정을 평가하여 중요한 성능 영역을 파악합니다. 팀과 함께 이러한 영역을 검토하여 병목 현상을 방지할 수 있는 모니터링 및 솔루션을 논의합니다.
- 성능 개선을 실험하고 데이터로 이러한 변경 사항을 측정합니다. 예를 들어 워크로드에 대한 새로운 개선 사항과 성능 영향을 테스트하기 위해 CloudWatch Evidently를 사용할 수 있습니다.

리소스

관련 문서:

- What's new in AWS Observability at re:Invent 2023
- Amazon Builders' Library
- X-Ray 설명서
- Amazon CloudWatch RUM
- Amazon DevOps Guru

관련 비디오:

- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 years of Amazon operational excellence
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

관련 예제:

- Measure page load time with Amazon CloudWatch Synthetics

- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Python
- Distributed Load Testing on AWS

PERF05-BP03 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.

일반적인 안티 패턴:

- 시간이 지나면 현재 아키텍처가 정적 아키텍처가 되고 업데이트되지 않는다고 가정합니다.
- 시간이 지나면 타당한 지표 없이 아키텍처 변경을 도입합니다.

이 모범 사례 확립의 이점: 아키텍처 변경을 위한 프로세스를 정의하면 수집된 데이터를 사용하여 워크로드 설계에 지속적으로 영향을 줄 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

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 서비스나 기술을 습득할 때 이 정보를 사용하여 제약 조건이나 병목 현상을 완화하는 방법을 파악합니다.

워크로드에 대한 주요 성능 제약을 파악합니다. 워크로드의 성능 제약을 문서화하면 어떤 종류의 혁신이 워크로드의 성능을 개선할 수 있는지 알 수 있습니다.

구현 단계

- KPI 식별: PERF05-BP01 워크로드 상태 및 성능을 측정하기 위한 핵심 성과 지표(KPI) 수립에 설명된 대로 워크로드 성능 KPI를 식별하여 워크로드의 기준을 설정합니다.
- 모니터링 구현: AWS 관찰성 도구를 사용하여 성과 지표를 수집하고 KPI를 측정합니다.
- 분석 수행: 심층 분석을 수행하여 PERF05-BP02 모니터링 솔루션을 사용하여 성능이 가장 중요한 영역 파악에 설명된 대로 워크로드에서 성능이 저조한 영역(예: 구성 및 애플리케이션 코드)을 식별합니다. 분석 및 성능 도구를 사용하여 성능 개선 전략을 식별합니다.
- 개선 사항 검증: 샌드박스 또는 사전 프로덕션 환경을 사용하여 개선 전략의 유효성을 검증합니다.

- 변경 사항 구현: 프로덕션 환경에서 변경 사항을 구현하고 워크로드의 성능을 지속적으로 모니터링합니다. 개선 사항을 문서화하고 변경 사항을 이해관계자에게 전달합니다.
- 보관 및 개선: 성과 개선 프로세스를 정기적으로 검토하여 개선이 필요한 영역을 식별합니다.

리소스

관련 문서:

- AWS 블로그
- AWS의 새로운 소식
- AWS Skill Builder

관련 비디오:

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance

관련 예제:

- AWS Github

PERF05-BP04 워크로드 로드 테스트

워크로드 로드 테스트를 통해 프로덕션 로드를 처리할 수 있는지 확인하고 성능 병목 현상을 파악할 수 있습니다.

일반적인 안티 패턴:

- 전체 워크로드가 아니라 워크로드의 개별 부분을 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.
- Amazon EC2 테스트 정책을 참조하거나 시뮬레이션된 이벤트 제출 양식을 제출하지 않고도 로드 테스트를 수행할 수 있습니다. 이는 서비스 거부 이벤트처럼 보이기 때문에 테스트 실행이 실패하게 됩니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

클라우드에서의 로드 테스트는 예상되는 사용자 부하가 있는 실제 조건에서 클라우드 워크로드의 성능을 측정하는 프로세스입니다. 이 프로세스에는 프로덕션과 유사한 클라우드 환경을 프로비저닝하고, 로드 테스트 도구를 사용하여 로드를 생성하며, 지표를 분석하여 실제 워크로드 처리 능력을 평가하는 작업이 포함됩니다. 로드 테스트는 프로덕션 데이터의 통합 또는 제거 버전(민감한 정보 또는 식별 정보 제거)을 사용하여 실행되어야 합니다. 전송 파이프라인의 일부로 로드 테스트를 자동으로 수행하고 결과를 미리 정의된 KPI 및 임곗값과 비교합니다. 이 프로세스를 통해 필요한 성능을 계속해서 달성할 수 있습니다.

구현 단계

- 테스트 목표 정의: 평가하려는 워크로드의 성능 측면(예: 처리량 및 응답 시간)을 식별합니다.
- 테스트 도구 선택: 워크로드에 적합한 로드 테스트 도구를 선택하고 구성합니다.
- 환경 설정: 프로덕션 환경에 따라 테스트 환경을 설정합니다. AWS 서비스를 사용하면 프로덕션 규모의 환경을 실행하여 아키텍처를 테스트할 수 있습니다.
- 모니터링 구현: Amazon CloudWatch와 같은 모니터링 도구를 사용하여 아키텍처의 리소스 전반에서 지표를 수집합니다. 사용자 지정 지표를 수집하고 게시할 수도 있습니다.
- 시나리오 정의: 로드 테스트 시나리오 및 파라미터(테스트 기간 및 사용자 수 등)를 정의합니다.
- 로드 테스트 수행: 대규모로 테스트 시나리오를 수행합니다. AWS 클라우드를 활용하여 워크로드를 테스트하여 확장에 실패하거나 비선형적인 방식으로 확장되는 부분을 발견하세요. 예를 들어, 스팟 인스턴스를 사용하여 저렴한 비용으로 로드를 생성하고 프로덕션 환경에서 발생하기 전에 병목 현상을 발견할 수 있습니다.
- 테스트 결과 분석: 결과를 분석하여 성능 병목 현상과 개선이 필요한 영역을 파악합니다.
- 조사 결과 문서화 및 공유: 조사 결과 및 권장 사항을 문서화하고 보고합니다. 이해관계자와 이 정보를 공유하면 이해관계자가 성능 최적화 전략과 관련하여 정보에 입각한 결정을 내리는 데 도움이 됩니다.
- 지속적인 반복: 로드 테스트는 정기적으로, 특히 시스템 변경 또는 업데이트 후에 수행해야 합니다.

리소스

관련 문서:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

관련 비디오:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing
- AWS re:Invent 2022 - Scaling on AWS for your first 10 million users
- Solving with AWS Solutions: Distributed Load Testing

- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

관련 예제:

- Distributed Load Testing on AWS

PERF05-BP05 자동화를 사용하여 성능 관련 문제 사전 해결

핵심 성과 지표(KPI)를 모니터링 및 경보 시스템과 함께 사용하여 성능 관련 문제를 선제적으로 해결합니다.

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 확립의 이점: 경보 작업을 사전에 해결하면 지원 직원이 자동으로 실행할 수 없는 항목에 집중할 수 있습니다. 이를 통해 운영 담당자는 모든 알람을 처리하는 데 부담을 느끼지 않고 중요한 경보에만 집중할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

경보를 사용하여 가능한 경우 문제를 해결하는 자동화 작업을 트리거합니다. 자동 대응이 불가능한 경우 대응을 수행할 수 있는 담당자에게 경보를 에스컬레이션합니다. 예를 들어 필요한 핵심 성과 지표(KPI) 값을 예측하고 해당 값이 특정 임곗값을 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.

구현 단계

- 수정 워크플로 식별: 자동으로 해결할 수 있는 성능 문제를 식별하고 이해합니다. Amazon CloudWatch 또는 AWS X-Ray와 같은 AWS 모니터링 솔루션을 사용하여 문제의 근본 원인을 더 잘 파악할 수 있습니다.
- 자동화 프로세스 정의: 문제를 자동으로 해결하는 데 사용할 수 있는 단계별 수정 프로세스를 만듭니다.
- 시작 이벤트 구성: 수정 프로세스를 자동으로 시작하도록 이벤트를 구성합니다. 예를 들어 CPU 사용률이 특정 임곗값에 도달하면 인스턴스를 자동으로 다시 시작하도록 트리거를 정의할 수 있습니다.
- 수정 자동화: AWS 서비스 및 기술을 사용하여 수정 프로세스를 자동화합니다. 예를 들어, AWS Systems Manager Automation에서는 수정 프로세스를 자동화할 수 있는 안전하고 확장 가능한 방법을 제공합니다. 문제를 성공적으로 해결하지 못한 경우 자체 복구 논리를 사용하여 변경 내용을 되돌립니다.
- 워크플로 테스트: 사전 프로덕션 환경에서 자동화된 수정 프로세스를 테스트합니다.
- 워크플로우 구현: 프로덕션 환경에서 자동화된 문제 해결을 구현합니다.
- 플레이북 개발: 시작 이벤트, 수정 논리 및 실행된 조치를 포함하여 수정 계획의 단계를 설명하는 플레이북을 개발하고 문서화합니다. 이해관계자가 자동화된 수정 이벤트에 효과적으로 대응할 수 있도록 교육해야 합니다.
- 검토 및 개선: 자동화된 수정 워크플로의 효과를 정기적으로 평가합니다. 필요한 경우 시작 이벤트 및 수정 논리를 조정합니다.

리소스

관련 문서:

- CloudWatch 설명서

- 모니터링, 로깅 및 성능 AWS Partner Network 파트너
- X-Ray 설명서
- Using Alarms and Alarm Actions in CloudWatch
- Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- Automate your Amazon Redshift performance tuning with automatic table optimization

관련 비디오:

- AWS re:Invent 2023 - Strategies for automated scaling, remediation, and smart self-healing
- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2021 - Intelligently automating cloud operations
- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - Automating patch management and compliance using AWS
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS
- AWS re:Invent 2021 -{New Launch} Automatically detect and resolve issues with Amazon DevOps Guru
- AWS re:Invent 2023 - Centralize your operations

관련 예제:

- CloudWatch Logs Customize Alarms

PERF05-BP06 워크로드 및 서비스 최신 상태 유지

새로운 클라우드 서비스 및 기능에 대한 최신 정보를 파악하여 효율적인 기능을 채택하고 문제를 제거하며 워크로드의 전반적인 성능 효율성을 개선할 수 있습니다.

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: 새로운 서비스 및 제품에 대한 최신 정보를 파악하는 프로세스를 구축하면 새로운 기능을 채택하고 문제를 해결하며 워크로드 성능을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

새로운 서비스, 설계 패턴 및 제품 기능이 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.

구현 단계

- 인벤토리 작업 수행: 워크로드 소프트웨어 및 아키텍처를 조사하여 업데이트하는 데 필요한 구성 요소를 식별합니다.
- 업데이트 소스 식별: 워크로드 구성 요소 관련 뉴스 및 업데이트 소스를 파악합니다. 예를 들어, AWS의 새로운 소식 블로그 구독하여 워크로드 구성 요소와 일치하는 제품을 구입할 수 있습니다. RSS 피드를 구독하거나 이메일 구독을 관리할 수 있습니다.
- 업데이트 일정 정의: 워크로드에 대한 새로운 서비스 및 기능을 평가할 일정을 정의할 수 있습니다.
  - AWS Systems Manager Inventory를 사용하여 Amazon EC2 인스턴스에서 운영 체제(OS), 애플리케이션, 인스턴스 메타데이터를 수집하고 소프트웨어를 실행 중인 인스턴스, 소프트웨어 정책에 필요한 구성, 업데이트해야 할 인스턴스를 신속하게 파악합니다.
- 새 업데이트 평가: 워크로드 구성 요소의 업데이트 방법을 파악합니다. 클라우드의 민첩성을 활용하여 새로운 기능이 워크로드를 어떻게 개선하여 성능 효율성을 높일 수 있는지 빠르게 테스트합니다.
- 자동화 사용: 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.
- 프로세스 문서화: 업데이트 및 새로운 서비스를 평가하기 위한 프로세스를 문서화합니다. 업데이트 및 새로운 서비스를 연구, 테스트, 실험 및 검증하는 데 필요한 시간과 공간을 담당자에게 제공합니다. 문서화된 비즈니스 요구 사항 및 KPI를 다시 참조하여 비즈니스에 긍정적인 영향을 줄 업데이트에 대한 우선순위를 지정할 수 있습니다.

리소스

관련 문서:

- AWS 블로그
- AWS의 새로운 소식
- Implementing up-to-date images with automated EC2 Image Builder pipelines

관련 비디오:

- AWS re:Inforce 2022 - Automating patch management and compliance using AWS
- All Things Patch: AWS Systems Manager | AWS Events

관련 예제:

- Inventory and Patch Management
- One Observability 워크숍

PERF05-BP07 정기적으로 지표 검토

정기적인 유지 관리의 일환으로 또는 이벤트나 인시던트 대응 과정에서 수집된 지표를 검토합니다. 이러한 검토를 수행하면 문제를 해결하는 데 반드시 필요했던 지표와 문제를 식별, 해결 또는 방지하는 데 도움이 되었던 지표(추적한 경우)를 추가로 파악할 수 있습니다.

일반적인 안티 패턴:

- 지표가 장기간 경보 상태로 유지되는 것을 허용합니다.
- 자동화 시스템으로 수행할 수 없는 경보를 생성합니다.

이 모범 사례 확립의 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 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

지표 수집 및 모니터링을 지속적으로 개선합니다. 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.

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.

구현 단계

- 지표 정의: 응답 시간 및 리소스 사용률과 같은 지표 등 워크로드 목표에 맞춰 모니터링할 중요한 성과 지표를 정의합니다.
- 기준선 설정: 각 지표에 대한 기준과 원하는 값을 설정합니다. 기준은 편차 또는 이상을 식별하기 위한 기준을 제공해야 합니다.
- 주기 설정: 주기(예: 주별 또는 월별)를 설정하여 중요 지표를 검토합니다.
- 성능 문제 식별: 각 검토 과정에서 추세와 기준값과의 편차를 평가합니다. 성능 병목 현상 또는 이상 징후가 있는지 찾아봅니다. 식별된 문제의 경우 심층적인 근본 원인 분석을 수행하여 문제의 주요 원인을 파악합니다.
- 수정 조치 식별: 분석을 사용하여 수정 조치를 식별합니다. 여기에는 파라미터 조정, 버그 수정, 리소스 규모 조정이 포함될 수 있습니다.
- 조사 결과 문서화: 식별된 문제, 근본 원인 및 시정 조치를 포함하여 조사 결과를 문서화합니다.
- 반복 및 개선: 지표 검토 프로세스를 지속적으로 평가하고 개선합니다. 이전 검토에서 배운 내용을 활용하여 시간이 지남에 따라 프로세스를 개선합니다.

리소스

관련 문서:

- CloudWatch 설명서
- CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집
- CloudWatch Metrics Insights를 사용하는 지표 쿼리
- 모니터링, 로깅 및 성능 AWS Partner Network 파트너
- X-Ray 설명서

관련 비디오:

- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS

관련 예제:

- Creating a dashboard with QuickSight
- CloudWatch Dashboards

비용 최적화

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

COST 1. 클라우드 재무 관리를 어떻게 구현하나요?

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 소유권 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 소유권 설정

조직 전체에서 비용 인식의 확립과 유지를 담당하는 팀(클라우드 비즈니스 오피스 또는 클라우드 혁신 센터 또는 FinOps 팀)을 구성합니다. 비용 최적화 책임자는 전체 조직 및 클라우드 관련 재무를 이해하는 개인 또는 팀(재무, 기술 및 비즈니스 팀 인력 필요)이 될 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

여기에서는 클라우드 컴퓨팅에서 비용 인식 문화의 확립과 유지를 담당하는 클라우드 비즈니스 오피스(CBO) 또는 클라우드 혁신 센터(CCoE) 부서 또는 팀을 소개합니다. 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 바탕으로 다방면에 걸친 접근을 필요로 합니다. 이 역할은 다음 세 가지 부문 내에서 비용 최적화를 실행하여 워크로드의 효율성을 개선할 수 있습니다.

- 중앙화: 고객은 FinOps 팀, 클라우드 재무 관리(CFM) 팀, 클라우드 비즈니스 오피스(CBO) 또는 클라우드 혁신 센터(CCoE)와 같은 지정된 팀을 통해 거버넌스 메커니즘을 설계 및 구현하고 전사적으로 모범 사례를 추진할 수 있습니다.
- 탈집중화: 기술 팀이 비용 최적화를 실행하도록 영향을 줍니다.
- 하이브리드: 중앙화 팀과 탈중앙화 팀이 함께 비용 최적화를 실행하기 위해 협력할 수 있습니다.

이 역할의 성과는 비용 최적화 목표(예: 워크로드 효율성 지표)를 실행하고 제공할 수 있는 능력을 기준으로 측정할 수 있습니다.

이 역할에 대한 경영진 차원의 지원을 아끼지 않는 것이 성공의 핵심 요소입니다. 스폰서는 비용 효율적인 클라우드 사용의 옹호자로 간주되며, 팀에 에스컬레이션 지원을 제공하여 비용 최적화 활동이 조직이 정의한 우선순위 수준에 따라 처리될 수 있도록 합니다. 그렇지 않으면 지침이 무시되고 비용 절감 기회를 우선시하지 않게 됩니다. 스폰서와 팀은 조직이 클라우드를 효율적으로 사용하고 비즈니스 가치를 전달할 수 있도록 함께 돕습니다.

Business, Enterprise-On-Ramp 또는 Enterprise 지원 플랜을 사용 중이며 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 부서가 비용 관리에 기여하고 관심을 가져야 합니다. 조직 내의 공통 팀으로는 일반적으로 재무, 애플리케이션 또는 제품 책임자, 관리, 기술 팀(DevOps) 등이 있습니다. 일부(재무 또는 기술)는 정규직이며 일부는 필요에 따라 주기적으로 근무합니다. CFM을 수행하는 개인 또는 팀에는 다음과 같은 기술이 필요합니다.
  - 소프트웨어 개발: 스크립트 및 자동화가 구축되는 경우에 해당합니다.
  - 인프라 엔지니어링: 스크립트를 배포하고 프로세스를 자동화하고 서비스 또는 리소스 프로비저닝 방법을 파악하려는 경우에 해당합니다.
  - 운영 감각: CFM은 클라우드의 효율적인 사용을 측정, 모니터링, 수정, 계획 및 조정하여 클라우드를 효율적으로 운영할 수 있는 솔루션입니다.
- 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 프로그램 생성 및 실행, 비용 최적화를 위한 모니터링 및 보고와 같은 조직 차원의 표준 개발, 최적화에 대한 워크로드 목표 설정 등이 있습니다. 또한 이 역할은 조직의 비용 최적화 역량에 대해 조직에 정기적으로 보고해야 합니다.

  가치 또는 비용 기반 핵심 성과 지표(KPI)를 정의할 수 있습니다. KPI를 정의할 때 효율성과 예상되는 비즈니스의 성과 측면에서 예상 비용을 계산할 수 있습니다. 가치 기반 KPI는 비용 및 사용량 지표를 비즈니스 가치 동인과 연계하고 AWS 지출 변화를 합리화하는 데 도움이 됩니다. 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. 그 후에는 주요 워크로드가 보다 자세히 보고됩니다.

이러한 정기 검토 중에 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 KPI를 식별하는 데 도움이 될 수 있습니다.

리소스

관련 문서:

- AWS CCOE 블로그
- Creating Cloud Business Office
- CCOE - Cloud Center of Excellence

관련 비디오:

- Vanguard CCOE Success Story

관련 예제:

- Using a Cloud Center of Excellence (CCOE) to Transform the Entire Enterprise
- Building a CCOE to transform the entire enterprise
- 7 Pitfalls to Avoid When Building CCOE

COST01-BP02 재무 팀과 기술 팀 간의 파트너십 수립

클라우드 여정의 모든 단계에서 비용 및 사용량 논의에 재무 및 기술 팀을 참여시킵니다. 팀은 정기적으로 만나 조직의 목적과 목표, 비용 및 사용량의 현재 상태, 재무 및 회계 실무와 같은 주제에 대해 논의합니다.

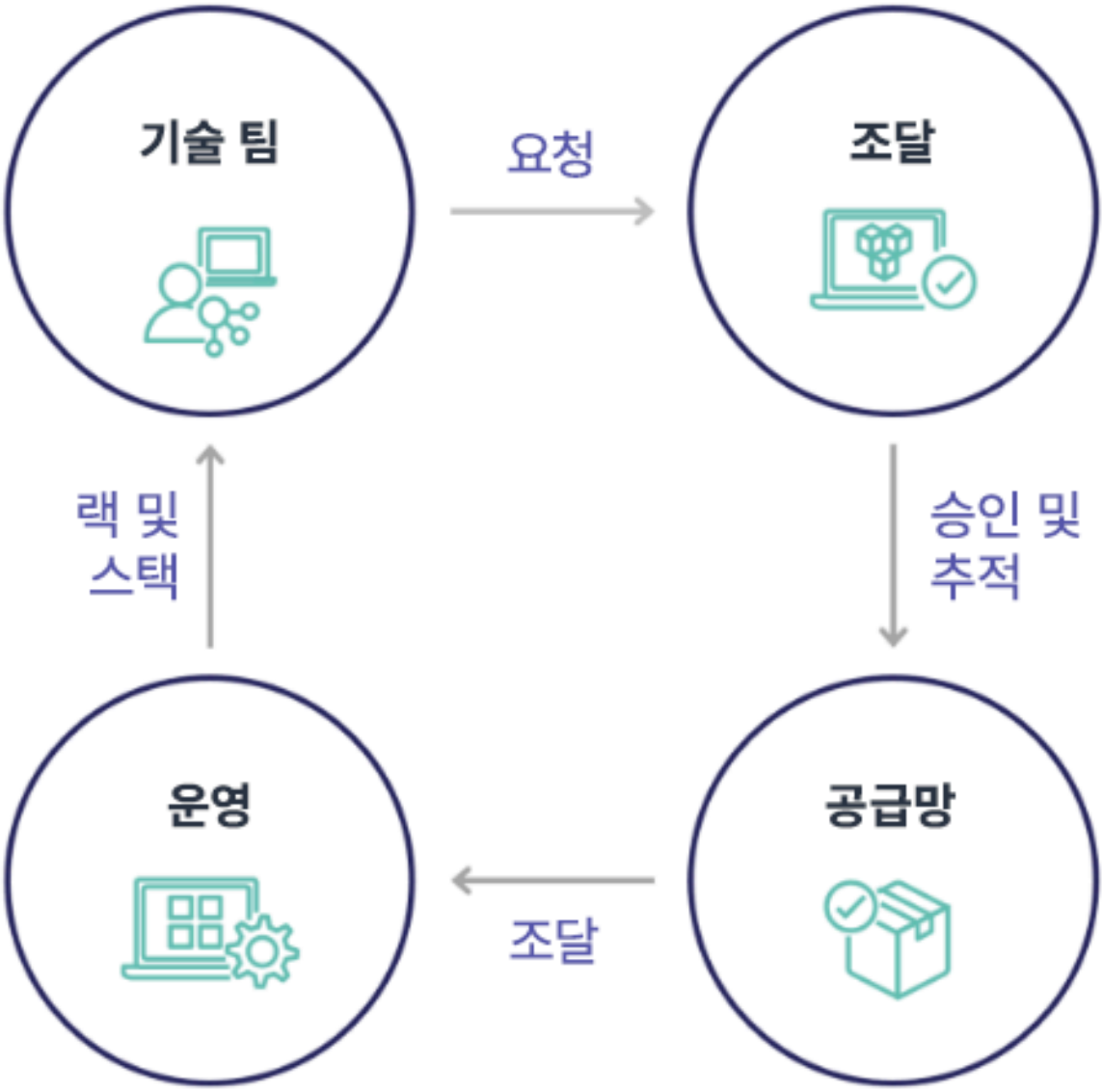
이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

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 계정과 올바른 권한 집합만 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 관점에서 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, 사용량에 따른 요금, 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 수준에 관한 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: 예약형 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 예약형 인스턴스 또는 절감형 플랜 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 투자 수익률(ROI)을 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 구성원이 파트너십에 참여하는지 확인하세요. 재무 팀의 관련 구성원은 클라우드 청구서와 관련하여 업무를 수행하는 사람들입니다. 대개 CFO, 재무 관리자, 재무 계획자, 비즈니스 분석가, 구매, 소싱 담당자가 여기에 해당합니다. 기술 구성원은 대개 제품 및 애플리케이션 소유자, 기술 관리자 및 클라우드에 구축된 모든 팀의 담당자가 됩니다. 다른 구성원으로는 제품 사용에 영향을 주는 마케팅과 같은 사업부 소유자와 목표 및 메커니즘에 부합하도록 하고 보고를 지원하는 컨설턴트와 같은 서드파티 등이 있을 수 있습니다.
- 토롤 주제 정의: 팀 간에 공통된 주제나 공동의 이해가 필요한 주제를 정의합니다. 발생한 시점부터 청구서가 지불될 때까지 비용을 추적합니다. 관련된 모든 구성원과 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 지침

기존의 온프레미스 IT 환경에서 고객은 고정 비용을 계획하는 데 어려움을 겪는 경우가 많습니다. 고정 비용은 드물게 변동하며 보통은 피크 수요를 충족하기 위해 새 IT 하드웨어와 서비스를 구입하는 경우

입니다. 이와 대조적으로, AWS 클라우드는 고객이 실제 IT 및 비즈니스 요구 사항에 따라 사용하는 리소스에 대해 비용을 지불하는 다른 접근 방식을 채택합니다. 클라우드 환경에서 수요는 월별, 일별 또는 시간별로 변동될 수 있습니다.

클라우드를 사용하면 효율성, 속도, 민첩성이 향상되어 비용 및 사용 패턴의 변동성이 커집니다. 워크로드 효율성 향상이나 새로운 워크로드 및 기능의 배포에 따라 비용이 감소하거나 때때로 증가할 수 있습니다. 확장되는 고객 기반에 맞춰 워크로드가 확장됨에 따라 리소스 접근성이 증대되어 클라우드 사용량과 비용도 함께 증가합니다. 클라우드 서비스의 이러한 유연성은 비용 및 예측으로 확장되어 어느 정도의 탄력성을 제공합니다.

이처럼 변화하는 비즈니스 요구 사항 및 수요 동인과 긴밀하게 연계하고 가능한 가장 정확한 계획을 세우는 것이 중요합니다. 기존의 조직 예산 프로세스는 이러한 변동성을 수용하도록 조정되어야 합니다.

새 워크로드의 비용을 예측하면서 비용 모델링을 고려하세요. 비용 모델링은 예상 클라우드 비용에 대한 기본 이해를 생성하여 총 소유 비용(TCO), 투자 수익률(ROI) 및 기타 재무 분석을 수행하고, 이해관계자와 함께 타겟 및 기대치를 설정하고, 비용 최적화 기회를 식별하는 데 도움이 됩니다.

조직은 비용 정의와 수락된 그룹화를 이해해야 합니다. 얼마나 세부적으로 예측할 것인지는 조직의 구조와 내부 워크플로에 따라 달라질 수 있습니다. 특정 요구 사항 및 조직 설정에 맞는 세부 수준을 선택하세요. 예측이 어느 수준에서 수행되는지 이해하는 것이 중요합니다.

- 관리 계정 또는 AWS Organizations 수준: 관리 계정은 AWS Organizations를 생성할 때 사용한 계정을 말합니다. 조직에는 기본적으로 관리 계정이 하나만 있습니다.
- 연결 계정 또는 구성원 계정: Organizations의 계정은 AWS 리소스를 포함하는 표준 AWS 계정으로, 이러한 리소스에 액세스할 수 있는 자격 증명입니다.
- 환경: 환경은 애플리케이션 버전을 실행 중인 AWS 리소스 모음입니다. 연결된 계정 또는 구성원 계정을 여러 개 사용하여 하나의 환경을 만들 수 있습니다.
- 프로젝트: 프로젝트는 정해진 기간에 달성해야 하는 정해진 목표 또는 작업의 조합입니다. 예측 과정에서 프로젝트 수명 주기를 고려하는 것이 중요합니다.
- AWS 서비스: 예측을 위해 AWS 서비스를 그룹화할 수 있는 컴퓨팅 또는 스토리지 서비스와 같은 그룹 또는 범주입니다.
- 사용자 지정 그룹화: 사업부, 비용 센터, 팀, 비용 할당 태그, 비용 범주, 연결 계정 또는 이들의 조합 등 조직의 요구 사항에 따라 사용자 지정 그룹을 생성할 수 있습니다.

사용 비용에 영향을 미칠 수 있는 비즈니스 동인을 식별하고 각 동인을 개별적으로 예측하여 예상 사용량을 미리 계산하세요. 일부 동인은 조직 내 IT 및 제품 팀과 연관된 것일 수 있습니다. 영업, 마케팅 및

비즈니스 리더는 마케팅 이벤트, 프로모션, 지리적 확장, 합병 및 인수 등의 다른 비즈니스 동인을 잘 알고 있으므로 이들과 협업하고 이러한 수요 동인을 모두 고려하는 것도 중요합니다.

AWS Cost Explorer를 사용하면 정의된 미래의 시간 범위에서 과거 지출을 기준으로 추세 기반 예측을 수행할 수 있습니다. AWS Cost Explorer의 예측 엔진은 비용 유형(예: 예약형 인스턴스)을 기준으로 과거 데이터를 구분하고 기계 학습과 규칙 기반 모델을 결합하여 모든 비용 유형 전반에서 비용을 개별적으로 예측합니다.

예측 프로세스를 수립하고 모델을 구축한 후에는 AWS Budgets를 사용하여 기간, 반복 주기 또는 금액(고정 또는 가변)을 지정하고 서비스, AWS 리전 및 태그 등과 같은 필터를 추가해 사용자 지정 예산을 세부 수준에서 설정할 수 있습니다. 예산은 일반적으로 1년 단위로 준비되고 고정되어 있으며, 관련된 모든 사람이 엄격하게 준수해야 합니다. 반면 예측은 더 유연하여 연중 재조정이 가능하며 1년, 2년 또는 3년 기간에 걸쳐 동적으로 추정할 수 있습니다. 예산과 예측은 모두 다양한 기술 및 비즈니스 이해관계자 간의 재정적 기대치를 설정하는 데 중요한 역할을 합니다. 정확한 예측 및 구현을 통해 처음부터 비용 프로비저닝을 직접 담당하는 이해관계자가 책임을 지게 되며 아울러 전반적인 비용 인식도 높일 수 있습니다.

기존 예산의 성과에 대한 최신 정보를 확보하기 위해 AWS Budgets 보고서를 생성하고 자신과 이해관계자에게 해당 보고서가 이메일로 정기적으로 전송되도록 예약할 수 있습니다. 또한 사후 대응적인 실제 비용 또는 예상 비용을 기반으로 AWS Budgets 알림을 생성하여 잠재적 비용 초과에 대한 완화 조치를 구현할 시간을 확보할 수 있습니다. 비용이나 사용량이 실제로 특정 수준을 초과하거나 예산 금액을 초과할 것으로 예상되는 경우 알림을 받을 수 있습니다.

추세 기반 알고리즘(예: 기간별 비용을 입력으로 사용)을 사용하거나, 동적이고 가변적인 지출 환경에 적합한 동인 기반 알고리즘(예: 신제품 출시, 리전별 확장 또는 워크로드를 위한 새로운 환경)을 사용하여 기존의 예산 및 예측 프로세스를 보다 동적으로 조정하세요. Cost Explorer 또는 다른 도구를 사용하여 추세 기반 예측을 결정한 후에는 AWS Pricing Calculator를 사용하여 예상 사용량(트래픽, 초당 요청 또는 필요한 Amazon EC2 인스턴스)을 기반으로 AWS 사용 사례와 향후 비용을 추정할 수 있습니다.

이러한 예측 계산 및 추정을 기반으로 예산을 수립해야 하므로 해당 예측의 정확성을 추적합니다. 통합 클라우드 비용 예측의 정확성과 효과를 모니터링합니다. 예측과 비교하여 실제 지출을 정기적으로 검토하고 필요에 따라 조정하여 예측의 정확도를 개선하세요. 예측 편차를 추적하고 보고된 변동에 대한 근본 원인 분석을 수행하여 예측을 실행하고 조정합니다.

COST01-BP02 재무 팀과 기술 팀 간의 파트너십 수립에 언급된 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 늘려 보다 신속하게 변화에 대응할 수 있습니다.

구현 단계

- 조직 내 비용 언어 정의: 조직 내에서 다차원 및 그룹화를 사용하여 공통 AWS 비용 언어를 생성합니다. 이해관계자가 예측 세부 수준, 가격 책정 모델, 비용 예측 수준을 이해하도록 하세요.
- 추세 기반 예측 분석: AWS Cost Explorer 및 Amazon Forecast와 같은 추세 기반 예측 도구를 사용합니다. 서비스, 계정, 태그 및 비용 범주와 같은 여러 차원에서 사용 비용을 분석할 수 있습니다.
- 동인 기반 예측 분석: 비즈니스 동인이 클라우드 사용량에 미치는 영향을 파악하고 각 동인에 대해 개별적으로 예측하여 예상 사용 비용을 미리 계산합니다. 사업부 책임자 및 이해관계자와 긴밀히 협력하여 새로운 동인에 미치는 영향을 이해하고 예상되는 비용 변경을 계산하여 정확한 예산을 정의하세요.
- 기존 예측 및 예산 프로세스 업데이트: 추세 기반, 비즈니스 동인 기반 또는 이 두 예측 방법의 조합 등 채택된 예측 방법을 기반으로 예측 및 예산 프로세스를 정의합니다. 예산은 현실적이고 예측에 기반하여 계산되어야 합니다.
- 경고 및 알림 구성: AWS Budgets 알림 및 비용 이상 탐지를 사용하여 경고 및 알림을 받을 수 있습니다.
- 주요 이해관계자와 함께 정기 검토 수행: 예를 들어 IT, 재무, 플랫폼 팀 및 기타 비즈니스 영역의 이해관계자와 함께 비즈니스 방향과 사용량의 변화에 맞춰 조정합니다.

리소스

관련 문서:

- AWS Cost Explorer
- AWS Cost and Usage Report
- Forecasting with Cost Explorer
- QuickSight Forecasting
- AWS Budgets

관련 비디오:

- How can I use AWS Budgets to track my spending and usage
- AWS Cost Optimization Series: AWS Budgets

관련 예제:

- Understand and build driver-based forecasting
- How to establish and drive a forecasting culture
- How to improve your cloud cost forecasting
- Using the right tools for your cloud cost forecasting

COST01-BP04 조직의 프로세스에서 비용 인식 구현

비용 인식을 구현하고 사용량에 영향을 미치는 신규 또는 기존 프로세스에 투명성과 책임을 더하며 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 가속화합니다. 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 Budgets, AWS Budgets 보고서)를 활용합니다.

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 조직 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 금융 관리

관련 예제:

- Strategy for Efficient Cloud Cost Management
- Cost Control Blog Series #3: How to Handle Cost Shock
- A Beginner’s Guide to AWS Cost Management

COST01-BP05 비용 최적화 보고 및 알림

클라우드 예산을 설정하고 사용 중 이상을 감지하는 메커니즘을 구성하세요. 미리 정의된 목표에 대해 비용 및 사용량 알림을 받을 수 있도록 관련 도구를 구성하고 사용량이 해당 목표를 초과할 경우 알림

을 받을 수 있습니다. 정기 회의를 통해 워크로드의 비용 효율성을 분석하고 비용 인식을 제고할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

조직 내의 비용 및 사용량 최적화를 정기적으로 보고해야 합니다. 비용 성과를 논의하기 위한 전용 세션을 구현하거나 워크로드에 대한 정기적인 운영 보고 주기에 비용 최적화를 포함할 수 있습니다. 서비스와 도구를 사용하여 비용 성과를 정기적으로 모니터링하고 비용 절감 기회를 모색할 수 있습니다.

AWS Cost Explorer를 사용하여 여러 필터 및 세분화로 비용 및 사용량을 볼 수 있습니다. 여기에서는 서비스별 또는 계정별 비용, 일일 비용 또는 마켓플레이스 비용과 같은 대시보드 및 보고서를 제공합니다. AWS Budgets 보고서를 통해 구성된 예산을 기준으로 비용 및 사용량의 진행 상태를 추적합니다.

AWS Budgets를 통해 사용자 지정 예산을 설정하여 임곗값을 초과한 경우 비용 및 사용량을 추적하고 이메일로 받은 알림 또는 Amazon Simple Notification Service(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 구현합니다. AWS Cost Anomaly Detection를 사용하면 갑작스럽게 발생하는 비용을 줄이고 혁신 속도를 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.

QuickSight와 AWS Cost and Usage Report(CUR) 데이터를 함께 사용하면 세분화된 데이터로 사용자 정의 기반의 보고를 제공할 수 있습니다. QuickSight에서는 보고서 일정을 수립하고 과거 비용 및 사용량 또는 비용 절감 기회에 대한 비용 보고서를 이메일을 통해 정기적으로 받을 수 있습니다. QuickSight에 구축된 Cost Intelligence Dashboard(CID) 솔루션을 확인해 보세요. 이 솔루션은 고급 가시성을 제공합니다.

AWS Trusted Advisor를 사용합니다. 이는 프로비저닝된 리소스가 비용 최적화를 위한 AWS 모범 사례에 부합하는지 여부를 확인하는 지침을 제공합니다.

세부적인 비용 및 사용량과 비교한 시각적 그래프를 통해 절감형 플랜 권장 사항을 확인해 보세요. 시간 단위 그래프는 온디맨드 지출을 권장 절감형 플랜 약정과 함께 보여주며 예상 절감액, 절감형 플랜

적용 범위 및 절감형 플랜 사용률에 대한 인사이트를 제공합니다. 이를 통해 조직은 지출을 분석하기 위한 모델을 구축하는 데 시간과 리소스를 투자하지 않고도 절감형 플랜이 각 지출 시간에 적용되는 방식을 이해할 수 있습니다.

안정 상태 워크로드, 유휴 리소스 및 사용률이 낮은 리소스와 관련된 비용의 절감을 위해 AWS Cost Explorer에서 절감형 플랜, 예약형 인스턴스 및 Amazon EC2의 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 낭비를 줄이거나 계정 팀 및 파트너에게 도움을 요청하여 클라우드 비용을 최적화 및 절약합니다.

더 나은 리소스 구매 옵션을 선택할 수 있도록 정기적으로 보고서를 생성하여 워크로드에 대한 단위 비용을 절감하세요. 절감형 플랜, 예약형 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 Budgets를 구성합니다. 태그를 사용하여 전체 계정 지출에 대한 예산과 워크로드에 대한 예산을 설정합니다.
  - Well-Architected Labs: Cost and Governance Usage
- 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 클라우드 지출과 관련된 애플리케이션 팀 및 책임자, 재무 책임자 및 주요 의사 결정권자가 포함되어야 합니다.

리소스

관련 문서:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Cost and Usage Report

- AWS Budgets 모범 사례
- Amazon S3 분석

관련 예제:

- Key ways to start optimizing your AWS cloud costs

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: 중간

구현 지침

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.

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 특정 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(AWS Solutions Architects가 구축한 Amazon QuickSight 솔루션)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 활성화: AWS Budgets를 사용하면 알림을 구성할 수 있어 예산 유형이 미리

구성된 임곗값을 벗어난 경우 알림을 받을 수 있습니다. AWS Budgets을 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 Detection에서는 고급 기계 학습 기술을 사용하여 비정상적인 지출과 근본 원인을 식별하므로 빠르게 조치를 취할 수 있습니다. 또한 평가하고 싶은 지출 세그먼트(예: 개별 AWS 서비스, 구성원 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 Explorer를 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 Dashboard는 사용자 지정 및 액세스 가능한 대시보드로, 고유한 비용 관리 및 최적화 도구의 기반을 만드는 데 유용합니다.

리소스

관련 문서:

- AWS Budgets
- AWS Cost Explorer
- Daily Cost and Usage Budgets
- AWS Cost Anomaly Detection

관련 예제:

- AWS Cost Anomaly Detection Alert with Slack

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: 중간

구현 가이드

AWS는 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. AWS 비용 관리, AWS 뉴스 블로그, AWS 비용 관리 블로그 및 AWS의 새로운 소식을 주기적으로 검토하여 새로운 서비스 및 기능 릴리스에 대한 정보를 확인하세요. 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 가입할 수 있습니다.
- AWS 뉴스 구독: 정기적으로 AWS 뉴스 블로그와 AWS의 새로운 소식을 검토하여 새로운 서비스 및 기능 출시에 대한 정보를 확인하세요. RSS 피드를 구독하거나 이메일을 사용하여 발표 및 릴리스 소식을 팔로우할 수 있습니다.
- AWS 요금 인하 준수: 스케일 조정에서 얻은 경제적 효율성을 고객에게 되돌려 드리기 위해 AWS에서 실시하는 모든 서비스의 정기적인 가격 인하는 표준으로 자리를 잡았습니다. 2023년 9월 20일 기준으로 AWS는 2006년 이후 134회 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 포함한 이전의 가격 인하 노력에 대한 내용은 price-reduction category of the AWS News Blog에서 확인할 수 있습니다.
- AWS 이벤트 및 회의: 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 팀과 회의: 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 예제:

- Amazon EC2 – 15 Years of Optimizing and Saving Your IT Costs
- 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 지침

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 조정할 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 알림: 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 스토리 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 회의: 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 예제:

- AWS 클라우드 금융 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

비즈니스 가치를 정량화한다는 것은 기업이 취하는 조치와 결정을 통해 얻는 이점을 측정하는 것을 의미합니다. 비즈니스 가치는 유형(예: 비용 절감 또는 수익 증대)일 수도 있고 무형(예: 브랜드 평판 향상 또는 고객 만족도 향상)일 수도 있습니다.

비용 최적화를 통해 비즈니스 가치를 정량화한다는 것은 더 효율적으로 지출하기 위한 노력을 통해 얻을 수 있는 가치나 혜택을 파악하는 것을 의미합니다. 예를 들어 회사에서 AWS에 워크로드를 배포하는 데 10만 USD를 지출하고 나중에 최적화한다면 품질이나 출력에 영향을 주지 않으면서 새로 드는 비용이 8만 USD에 불과합니다. 이 시나리오에서는 비용 최적화를 통해 비즈니스 가치를 정량화하면 2만 USD를 절감할 수 있습니다. 그러나 비용 절감뿐만 아니라 납품 시간 단축, 고객 만족도 향상 또는 비용 최적화 노력으로 인한 기타 지표 측면에서 가치를 정량화할 수도 있습니다. 이해관계자들은 비용 최적화의 잠재적 가치, 워크로드 최적화 비용, 수익 가치를 판단해야 합니다.

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 비용을 줄이고 워크로드 결과 수준을 유지할 수 있는 절감형 플랜을 구매하는 경우 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 제거될 때 AWS에서 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

- 비즈니스 이점 평가: 지출 비용의 1 USD마다 얻을 수 있는 이점을 극대화하는 방식으로 AWS 클라우드 클라우드 비용을 분석하고 조정하는 프로세스입니다. 비즈니스 가치에 대한 고려 없이 비용을 절감하는 데에만 초점을 맞추지 말고 비용 최적화의 비즈니스 혜택과 투자 수익률을 고려하세요. 그러면 지출한 비용으로 더 많은 가치를 창출할 수 있습니다. 현명하게 지출하고 수익률이 가장 높은 분야에 투자와 지출을 하는 것이 관건입니다.
- 예상되는 AWS 비용 분석: 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 클라우드 경제학
- AWS 블로그
- AWS 비용 관리
- AWS 뉴스 블로그
- Well-Architected 신뢰성 원칙 백서
- AWS Cost Explorer

관련 비디오:

- Unlock Business Value with Windows on AWS

관련 예제:

- Measuring and Maximizing the Business Value of Customer 360
- The Business Value of Adopting Amazon Web Services Managed Databases
- The Business Value of Amazon Web Services for Independent Software Vendors
- Business Value of Cloud Modernization
- The Business Value of Migration to Amazon Web Services

지출 및 사용량 인식

Questions

- COST 2. 사용량을 어떻게 관리하나요?
- COST 3. 비용과 사용량을 어떻게 모니터링하나요?
- COST 4. 리소스를 어떻게 폐기하나요?

COST 2. 사용량을 어떻게 관리하나요?

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 구현
- 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 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 간

에 오해가 없음). 또한 당사의 메커니즘과 같이 정기적으로 점검하고 고객의 비즈니스 상황 또는 우선순위가 바뀌면 정책이 상황에 맞지 않을 수 있으므로 업데이트해야 합니다.

사용할 지리적 리전, 리소스를 실행해야 하는 하루 중 시간 등과 같이 넓은 범위의 상위 수준 정책부터 시작합니다. 그런 다음 다양한 조직 단위 및 워크로드에 대한 정책을 점진적으로 구체화합니다. 가장 많이 사용하는 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 이해관계자의 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 예약형 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 서드파티 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

- 이해관계자와 만남: 정책을 개발하려면 조직 내 이해관계자(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 구성원뿐만 아니라 보안 및 재무 팀과 같은 지원 그룹이 포함됩니다.
- 확인 받기: 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 생성: 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: 국가, 국가 내 지역을 포함하여 워크로드가 운영되는 곳을 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.
- 기능별 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 및 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: 워크로드 및 조직 요구 사항은 시간이 지남에 따라 변경될 수 있습니다. 조직의 우선순위와 일치하도록 워크로드 검토 일정을 정의합니다.
- 정책 문서화: 정의된 정책에 조직의 필요에 따라 액세스할 수 있는지 확인합니다. 이러한 정책은 환경의 액세스를 구현, 유지 관리 및 감사하는 데 사용됩니다.

리소스

관련 문서:

- Change Management in the Cloud
- 직무 기능에 대한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- Actions, Resources, and Condition Keys for AWS Services
- AWS의 관리 및 거버넌스
- Control access to AWS 리전 using IAM policies
- 글로벌 인프라 리전 및 AZ

관련 비디오:

- AWS Management and Governance at Scale

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 않을 경우 노출되는 위험 수준: 높음

구현 지침

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 특정 서비스의 채택 등이 포함될 수 있습니다. 측정 가능하고 달성 가능한 목표를 설정하여 비즈니스 운영에 중요한 효율성 개선을 측정할 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(구체성, 측정 가능성, 할당성, 현실성, 적시성(SMART) 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. 올바른 지표를 사용하고 조직에 대해 계산된 KPI를 설정하세요. 기본 KPI로 시작하여 나중에 비즈니스 요구 사항에 따라 발전시켜 나갈 수 있습니다.

비용 최적화의 목표는 워크로드 효율성을 높이는 것이며, 이는 즉 시간이 지남에 따라 워크로드의 비즈니스 결과당 비용을 줄이는 것입니다. 이 목표를 모든 워크로드에 적용하고 6개월에서 1년마다 효율성을 5% 높이는 것과 같은 타겟을 설정하세요. 클라우드에서는 비용 최적화 기능과 새로운 서비스 및 기능 릴리스를 통해 이를 달성할 수 있습니다.

타겟은 목표 달성을 위해 도달하고자 하는 정량화할 수 있는 벤치마크이며, 벤치마크는 실제 결과를 타겟과 비교합니다. KPI로 컴퓨팅 서비스(예: 스팟 채택, Graviton 채택, 최신 인스턴스 유형, 온디맨드 적용 범위), 스토리지 서비스(예: EBS GP3 채택, 오래된 EBS 스냅샷, Amazon S3 Standard 스토리지) 또는 데이터베이스 서비스 사용(예: RDS 오픈 소스 엔진, Graviton 채택, 온디맨드 적용 범위)의 단위당 비용에 대한 벤치마크를 설정합니다. 이러한 벤치마크와 KPI는 가장 비용 효과적인 방식으로 AWS 서비스를 사용하고 있는지 확인하는 데 도움이 될 수 있습니다.

다음 표에는 참조용 표준 AWS 지표 목록이 나와 있습니다. 각 조직은 이러한 KPI에 대해 서로 다른 타겟 값을 가질 수 있습니다.

범주	KPI(%)	설명
컴퓨팅	EC2 사용 범위	EC2 인스턴스의 전체 비용(또는 시간)을 기준으로 SP+RI+스팟을 사용하는 EC2 인스턴스 비용(또는 시간) 비교
컴퓨팅	컴퓨팅 SP/RI 사용률	총 가용 SP 또는 RI 시간을 기준으로 SP 또는 RI 사용 시간 비교
컴퓨팅	EC2/시간당 비용	EC2 비용을 해당 시간에 실행 중인 EC2 인스턴스 수로 나눈 값
컴퓨팅	vCPU 비용	모든 인스턴스의 vCPU당 비용
컴퓨팅	최신 인스턴스 세대	Graviton(또는 기타 최신 세대 인스턴스 유형)의 인스턴스 비율
데이터베이스	RDS 적용 범위	RDS 인스턴스의 총 비용(또는 시간)을 기준으로 RI를 사용한 RDS 인스턴스 비용(또는 시간) 비교
데이터베이스	RDS 사용률	총 가용 RI 시간을 기준으로 RI 사용 시간 비교
데이터베이스	RDS 가동 시간	RDS 비용을 해당 시간에 실행 중인 RDS 인스턴스 수로 나눈 값
데이터베이스	최신 인스턴스 세대	Graviton(또는 기타 최신 인스턴스 유형)의 인스턴스 비율
스토리지	스토리지 사용률	최적화된 스토리지 비용(예: Glacier, Deep Archive 또는

범주	KPI(%)	설명
		Infrequent Access)을 총 스토리지 비용으로 나눈 값
태그 지정	태그가 지정되지 않은 리소스	Cost Explorer: 1. 크레딧, 할인, 세금, 환불, 마켓플레이스를 필터링하고 최근 월별 비용을 복사합니다. 2. Cost Explorer에서 태그 없는 리소스만 표시를 선택합니다. 3. 태그 없는 리소스의 양을 월별 비용으로 나눕니다.

이 표를 사용하여 조직의 목표를 기반으로 계산해야 하는 타겟 값 또는 벤치마크 값을 포함합니다. 정확하고 현실적인 KPI를 정의하려면 비즈니스에 대한 특정 지표를 측정하고 해당 워크로드의 비즈니스 성과를 이해해야 합니다. 조직 내에서 성과 지표를 평가할 때는 각기 다른 용도에 맞는 여러 유형의 지표를 구별해야 합니다. 이러한 지표는 전반적인 비즈니스 영향을 직접 측정하기보다는 주로 기술 인프라의 성능과 효율성을 측정합니다. 예를 들어 서버 응답 시간, 네트워크 지연 시간 또는 시스템 가동 시간을 추적할 수 있습니다. 이러한 지표는 인프라가 조직의 기술 운영을 얼마나 잘 지원하는지 평가하는 데 중요합니다. 그러나 고객 만족, 수익 성장 또는 시장 점유율과 같은 광범위한 비즈니스 목표에 대한 직접적인 인사이트를 제공하지는 않습니다. 비즈니스 성과를 포괄적으로 이해하려면 이러한 효율성 지표를 비즈니스 성과와 직접 연관되는 전략적 비즈니스 지표로 보완하세요.

KPI 및 관련 비용 절감 기회를 거의 실시간으로 파악하고 시간 경과에 따른 진행 상황을 추적합니다. KPI 목표 정의 및 추적을 시작하려면 Cloud Intelligence Dashboards(CID)의 KPI 대시보드를 권장합니다. 비용 및 사용량 보고서(CUR)의 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 이해관계자가 이러한 방법을 채택하도록 해야 합니다.

구현 단계

- 예상 사용량 수준 정의: 먼저 사용량 수준에 초점을 맞춥니다. 애플리케이션 소유자, 마케팅 팀, 규모가 큰 비즈니스 팀과 협력하여 워크로드의 예상 사용량 수준을 파악합니다. 시간이 지남에 따라 고객 수요는 어떻게 변화할 수 있으며 시즌성 증가 또는 마케팅 캠페인으로 인해 어떤 변화가 있을 수 있나요?
- 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. 각 주요 지점의 비용을 지정하고 사용량이 변경될 때 비용 변동을 예측합니다.
- 비즈니스 목표 정의: 예상 사용량과 비용 변화의 결과를 예상되는 기술 변화나 실행 중인 모든 프로그램과 결합하고 워크로드에 대한 목표를 설정합니다. 목표는 사용량과 비용 그리고 둘 사이의 관계를 다루어야 합니다. 목표는 간단하고 간략하며, 기업에서 어떤 결과를 원하는지 이해하는 데 도움이 되어야 합니다(예: 미사용 리소스는 특정 비용 수준 미만으로 유지). 각각의 사용하지 않은 리소스 유형에 대해 목표를 정의하거나 목표와 타겟에 손실을 초래하는 비용을 정의할 필요는 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, 과다 공급으로 인한 낭비를 최소화하겠다는 목표를 설정할 수 있습니다. 이 목표에서는 프로덕션 워크로드의 첫 번째 계층에서 컴퓨팅 오버프로비저닝으로 인한 낭비가 계층 컴퓨팅 비용의 10%를 초과하지 않도록 하는 것을 타겟으로 삼을 수 있습니다. 또한 두 번째 타겟은 프로덕션 워크로드의 두 번째 계층에서 컴퓨팅 오버프로비저닝으로 인한 낭비가 계층 컴퓨팅 비용의 5%를 초과하지 않아야 한다는 것일 수 있습니다.

리소스

관련 문서:

- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- Control access to AWS 리전 using IAM policies
- S.M.A.R.T. Goals
- How to track your cost optimization KPIs with the CID KPI Dashboard

관련 비디오:

- Well-Architected Labs: Goals and Targets (Level 100)

관련 예제:

- What is a unit metric?
- Selecting a unit metric to support your business
- Unit metrics in practice – lessons learned
- How unit metrics help create alignment between business functions

COST02-BP03 계정 구조 구현

조직에 적합한 계정 구조를 구현합니다. 이를 통해 조직 전체에서 비용을 쉽게 할당하고 관리할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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. 그런 다음 관리 계정 모범 사례 및 구성원 계정 모범 사례를 따라 설계된 계정 구조를 기반으로 새 AWS 계정 계정을 생성하거나 기존 계정을 구성원 계정으로 선택할 수 있습니다.

이때 조직의 규모나 사용량에 관계없이 하나의 구성원 계정이 연결된 하나 이상의 관리 계정을 항상 보유하는 것이 좋습니다. 모든 워크로드 리소스는 구성원 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 계정을 생성합니다.

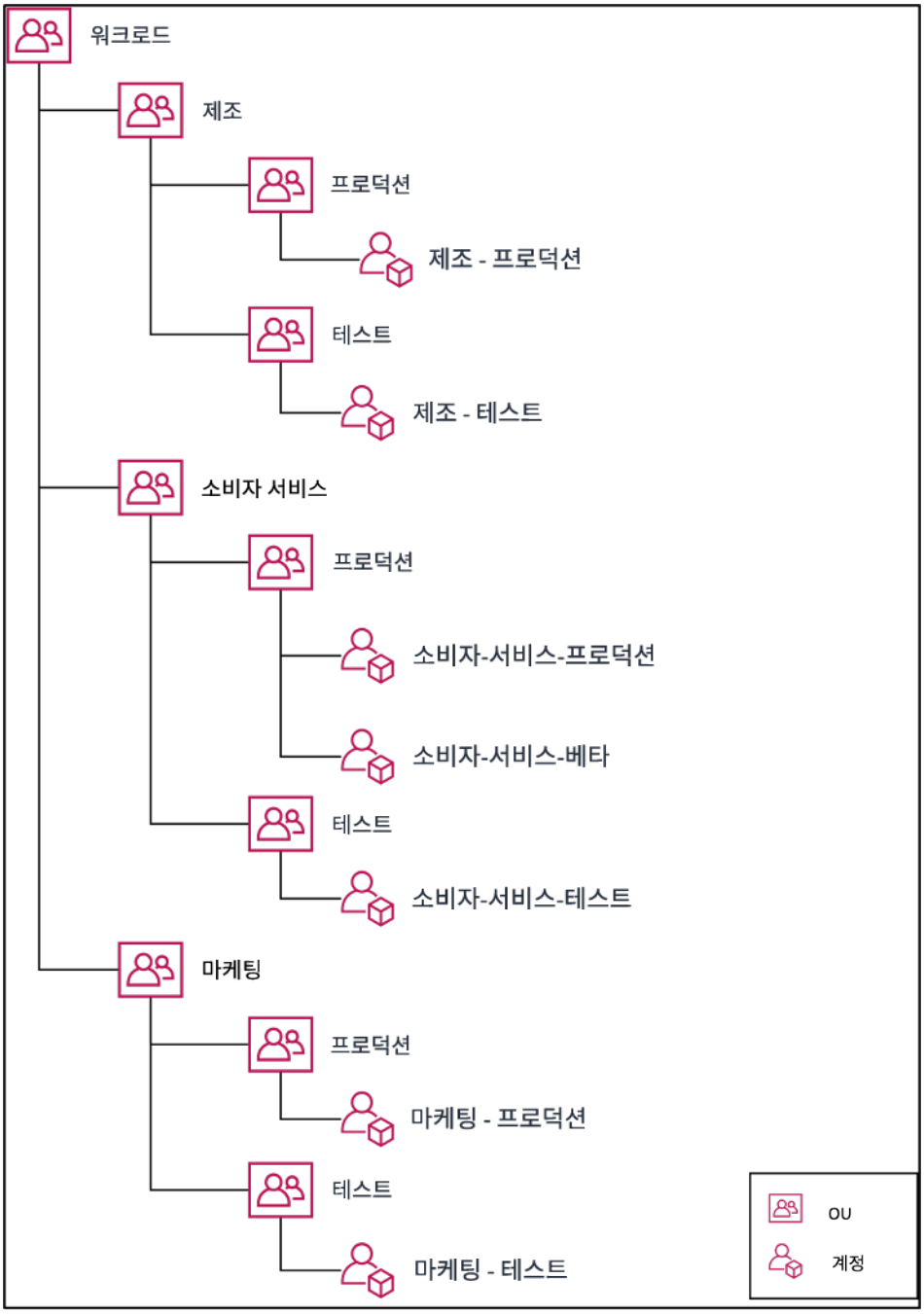
- 조직 단위, 비용 센터 또는 특정 워크로드 간에 관리 또는 재무 및 결제 작업을 분리해야 합니다.
- 특정 워크로드별로 AWS 서비스 한도가 설정됩니다.
- 워크로드와 리소스 간에 격리 및 분리에 대한 요구 사항이 있습니다.

AWS Organizations에서 통합 결제를 사용하는 경우 하나 이상의 구성원 계정과 관리 계정 간의 구조가 생성됩니다. 구성원 계정을 사용하면 비용과 사용량을 그룹으로 분리하고 구별할 수 있습니다. 각

조직 단위(재무, 마케팅, 영업 등), 각 환경 수명 주기(개발, 테스트, 프로덕션 등) 또는 각 워크로드(워크로드 a, b, c)용으로 별도의 구성원 계정을 생성한 다음 통합 결제를 사용하여 이러한 연결 계정을 집계하는 방식이 흔히 사용됩니다.

통합 결제에서는 여러 구성원 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 할인율을 극대화하고 약정 할인(절감형 플랜 및 예약형 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 여러 AWS 계정을 그룹화하는 예.

AWS Control Tower를 사용하면 여러 AWS 계정을 빠르게 설정하고 구성하여 조직의 요구 사항에 부합하는 거버넌스를 시행할 수 있습니다.

구현 단계

- 분리 요구 사항 정의: 분리 요구 사항은 보안, 신뢰성, 재무 구조와 같은 여러 요인을 결합한 것입니다. 각 요인을 순서대로 살펴보고 워크로드 또는 워크로드 환경이 다른 워크로드와 분리되어야 하는

지 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 정기적으로 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 이해관계자에 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 유지 관리합니다. 이러한 계정은 구성원 또는 연결된 계정입니다. 이러한 구성원 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 구성원 계정에 결제 데이터의 개별 보기를 제공할 수 있습니다. 구성원 계정의 사용 내역 또는 결제 데이터가 다른 계정에 표시되지 않아야 하거나 AWS와 별도의 청구서가 필요한 경우 여러 관리 또는 지급인 계정을 정의합니다. 이 경우 구성원 계정마다 고유의 관리 또는 지급인 계정이 있습니다. 리소스는 항상 구성원 또는 연결 계정에 배치되어야 합니다. 관리 또는 지급인 계정은 관리에만 사용해야 합니다.

리소스

관련 문서:

- 비용 할당 태그 사용
- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- Control access to AWS 리전 using IAM policies
- AWS Control Tower
- AWS Organizations
- 관리 계정 및 구성원 계정의 모범 사례
- Organizing Your AWS Environment Using Multiple Accounts
- Turning on shared reserved instances and Savings Plans discounts
- Consolidated billing
- Consolidated billing

관련 예제:

- Splitting the CUR and Sharing Access

관련 비디오:

- Introducing AWS Organizations
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

관련 예제:

- Defining an AWS Multi-Account Strategy for telecommunications companies
- Best Practices for Optimizing AWS 계정
- Best Practices for Organizational Units with AWS Organizations

COST02-BP04 그룹 및 역할 구현

정책에 따라 그룹과 역할을 구현하고 각 그룹에서 인스턴스와 리소스를 생성, 수정 또는 폐기할 수 있는 사용자를 제어합니다. 예를 들어 개발, 테스트 및 프로덕션 그룹을 구현합니다. 이는 AWS 서비스와 서드파티 솔루션에 적용됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

사용자 역할 및 그룹은 안전하고 효율적인 시스템을 설계하고 구현하는 데 필요한 기본 구성 요소입니다. 역할 및 그룹은 조직이 통제의 필요성과 유연성 및 생산성에 대한 요구 사항의 균형을 맞추어 궁극적으로 조직의 목표를 달성하고 사용자 요구를 충족하는 데 도움이 됩니다. AWS Well-Architected Framework 보안 원칙의 ID 및 액세스 관리 섹션에서 권장하는 바와 같이, 올바른 조건에서 올바른 사용자에게 올바른 리소스에 대한 액세스를 제공하려면 강력한 ID 관리 및 권한이 필요합니다. 사용자는 작업을 완료하는 데 필요한 액세스 권한만 받습니다. 이렇게 하면 무단 액세스 또는 오용과 관련된 위험을 최소화할 수 있습니다.

정책을 개발한 후에는 조직 내에서 논리적 그룹과 사용자 역할을 만들 수 있습니다. 이를 통해 권한을 할당하고, 사용을 제어하며, 강력한 액세스 제어 메커니즘을 구현하여 민감한 정보에 대한 무단 액세스를 방지할 수 있습니다. 개괄적인 수준에서 사람을 그룹화하는 작업부터 시작합니다. 일반적으로 이렇게 구성되는 그룹은 조직 단위 및 직무 역할(예: IT 부서의 시스템 관리자 또는 재무 관리자, 비즈니스 분석가)과 일치합니다. 유사한 작업을 수행하고 유사한 접근 권한이 필요한 사람이 각각의 그룹으로 분

류됩니다. 역할은 그룹이 수행해야 할 작업을 정의합니다. 개별 사용자보다 그룹 및 역할에 대한 권한을 관리하는 것이 더 쉽습니다. 역할과 그룹을 사용하면 모든 사용자에게 일관되고 체계적으로 권한을 할당하여 오류와 불일치를 예방할 수 있습니다.

사용자의 역할이 변경되었을 때, 관리자는 개별 사용자 계정을 재구성하는 대신 역할 또는 그룹 수준에서 액세스를 조정할 수 있습니다. 예를 들어 IT의 시스템 관리자는 모든 리소스를 생성할 수 있는 접근 권한이 필요하지만 분석 팀원은 분석 리소스만 생성하면 됩니다.

구현 단계

- 그룹 구현: 조직 정책에 정의된 사용자 그룹을 사용하여 필요한 경우 해당 그룹을 만듭니다. 사용자, 그룹 및 인증에 대한 모범 사례는 AWS Well-Architected 프레임워크의 보안 원칙을 참조하세요.
- 역할 및 정책 구현: 조직 정책에 정의된 작업을 사용하여 필요한 역할과 액세스 정책을 생성합니다. 역할 및 정책에 대한 모범 사례는 AWS Well-Architected Framework의 보안 원칙을 참조하세요.

리소스

관련 문서:

- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- AWS Well-Architected Framework 보안 원칙
- AWS Identity and Access Management (IAM)
- AWS Identity and Access Management 정책

관련 비디오:

- Why use Identity and Access Management

관련 예제:

- Control access to AWS 리전 using IAM policies
- Starting your Cloud Financial Management journey: Cloud cost operations

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 Budgets를 사용하면 AWS 비용, 사용량 및 약정 할인(절감형 플랜 및 예약형 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 Budgets를 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 임곗값을 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 AWS에서 거버넌스 정책을 시행합니다. IAM은 AWS 서비스 및 리소스에 대한 접근을 안전하게 관리할 수 있는 기능을 제공합니다. 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 구성원 계정에 대한 작업을 거부하거나 허용하도록 SCP를 구성할 수 있습니다. 액세스 관리 구현에 대한 자세한 내용은 Well-Architected 보안 원칙 백서를 참조하세요.

AWS Service Quotas의 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 생

성 및 폐기)를 파악하고 할당량 변경을 구현할 수 있는 속도를 고려해야 합니다. 필요한 경우 Service Quotas를 사용하여 할당량을 늘릴 수 있습니다.

구현 단계

- 지출에 대한 알림 구현: 정의된 조직 정책으로 AWS Budgets를 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 작업을 사전 구성할 수 있습니다. 예산 작업은 자동으로 시작하거나 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 감지되었을 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:

- 직무에 관한 AWS 관리형 정책
- AWS 다중 계정 결제 전략
- Control access to AWS 리전 using IAM policies
- AWS Budgets
- AWS Cost Anomaly Detection

- AWS 비용 관리

관련 비디오:

- How can I use AWS Budgets to track my spending and usage

관련 예제:

- IAM 액세스 관리 정책 예
- Example service control policies
- AWS 예산 작업
- 태그로 EC2 리소스 액세스를 제어하는 IAM 정책 생성
- 특정 Amazon EC2 리소스에 대한 IAM Identity 액세스 제한
- 채팅 애플리케이션 내 Amazon Q Developer를 사용한 비용 이상 탐지를 위한 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 막으세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

조직은 프로젝트 수명 주기를 효과적으로 추적함으로써 향상된 계획, 관리 및 리소스 최적화를 통해 더 나은 비용 관리를 실현할 수 있습니다. 추적을 통해 얻은 인사이트는 비용 효과성과 프로젝트의 전반적인 성공에 기여하는 정보 기반 의사 결정에 매우 중요합니다.

워크로드의 전체 수명 주기를 추적하면 워크로드 또는 워크로드 구성 요소가 더 이상 필요하지 않은 시기를 파악할 수 있습니다. 기존 워크로드와 구성 요소가 사용 중인 것처럼 보일 수 있지만 AWS에서 새 서비스나 기능이 출시되면 폐기되거나 채택될 수 있습니다. 워크로드의 이전 단계를 확인합니다. 워크로드가 프로덕션 단계로 전환된 후에는 이전 환경을 폐기하거나 다시 필요할 때까지 환경의 용량을 크게 줄일 수 있습니다.

리소스에 타임프레임 또는 알림을 태그하여 워크로드가 검토된 시간을 지정할 수 있습니다. 예를 들어 개발 환경을 몇 달 전에 마지막으로 검토했다면 다시 검토하여 새로운 서비스를 채택할 수 있는지 또는 환경이 사용 중인지 알아보는 것이 좋습니다. AWS의 myApplications를 사용하여 애플리케이션을 그룹화하고 태그를 지정하여 중요도, 환경, 최근 검토 날짜, 비용 센터와 같은 메타데이터를 관리하고 추

적할 수 있습니다. 워크로드의 수명 주기를 추적하고 애플리케이션의 비용, 상태, 보안 태세 및 성능을 모니터링하고 관리할 수 있습니다.

AWS는 엔터티 수명 주기 추적에 사용할 수 있는 다양한 관리 및 거버넌스 서비스를 제공합니다. AWS Config 또는 AWS Systems Manager를 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 지표와 현재 시스템을 통합하면 중요한 수명 주기 이벤트 보기를 만들고 리소스를 사전에 관리하여 불필요한 비용을 줄일 수 있습니다.

애플리케이션 수명 주기 관리(ALM)와 마찬가지로 프로젝트 수명 주기 추적에는 설계 및 개발, 테스트, 프로덕션, 지원, 워크로드 이중화 등 여러 프로세스, 도구 및 팀이 함께 작업해야 합니다.

프로젝트 수명 주기의 각 단계를 주의 깊게 모니터링함으로써 중요한 인사이트를 얻고 통제를 강화하여 성공적인 프로젝트 계획, 구현 및 완료를 촉진합니다. 이렇게 세심한 감독을 거치면 프로젝트가 품질 표준을 충족할 뿐만 아니라 제시간에 예산 범위 내에서 납품되어 전반적인 비용 효율성을 높인다는 것을 확인할 수 있습니다.

엔터티 수명 주기 추적에 대한 자세한 내용은 AWS Well-Architected 운영 우수성 원칙 백서를 참조하세요.

구현 단계

- 프로젝트 수명 주기 모니터링 프로세스 수립: Cloud Center of Excellence 팀은 프로젝트 수명 주기 모니터링 프로세스를 수립해야 합니다. 프로젝트의 통제, 가시성 및 성과를 개선하기 위해 워크로드 모니터링에 대한 구조적이고 체계적인 접근 방식을 수립합니다. 모니터링 프로세스를 투명하고 협력적이며 지속적인 개선에 집중하도록 만들어 효과와 가치를 극대화합니다.
- 워크로드 검토 수행: 조직 정책에 정의된 대로 일정한 주기를 설정하여 기존 프로젝트를 감사하고 워크로드 검토를 수행합니다. 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:

- Guidance for Tagging on AWS

- 애플리케이션 수명 주기 관리(ALM)란 무엇인가요?
- 직무에 관한 AWS 관리형 정책

관련 예제:

- Control access to AWS 리전 using IAM policies

관련 도구

- AWS Config
- AWS Systems Manager
- AWS Budgets
- AWS Organizations
- AWS CloudFormation

COST 3. 비용과 사용량을 어떻게 모니터링하나요?

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

비용 및 사용량 데이터의 분석과 투명성을 향상하기 위한 비용 관리 및 보고 도구를 설정합니다. 비용과 사용량을 추적하고 분리하는 데 도움이 되는 로그 항목을 생성하도록 워크로드를 구성합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

비용 관리 도구의 시간 단위 세분화와 같은 상세한 청구 정보를 통해 조직은 추가 세부 정보와 함께 소비량을 추적하고 비용 증가 원인을 파악할 수 있습니다. 이러한 데이터 소스는 전체 조직의 비용 및 사용량을 가장 정확하게 보여줍니다.

AWS Data Exports를 사용하여 AWS Cost and Usage Report(CUR) 2.0의 내보내기를 생성할 수 있습니다. 이것은 AWS로부터 상세한 비용 및 사용량 데이터를 받을 수 있는 새로운 방법이자 권장되는 방법입니다. 일부 개선 사항과 함께 모든 유료 AWS 서비스의 일별 또는 시간별 사용 세부 수준, 요금, 비용 및 사용 속성(CUR과 동일한 정보)을 제공합니다. 태그 지정, 위치, 리소스 속성 및 계정 ID를 포함하여 가능한 모든 차원을 CUR에서 사용할 수 있습니다.

생성하려는 내보내기 유형에 따라 표준 데이터 내보내기, QuickSight 통합을 통한 Cost and Usage Dashboard로 내보내기 또는 기존 데이터 내보내기의 세 가지 내보내기 유형이 있습니다.

- 표준 데이터 내보내기: Amazon S3에 반복적으로 전송되는 테이블의 사용자 지정 내보내기입니다.
- Cost and Usage Dashboard: QuickSight로 내보내고 통합하여 사전 구축된 Cost and Usage Dashboard를 배포합니다.
- 레거시 데이터 내보내기: 레거시 AWS Cost and Usage Report(CUR) 내보내기입니다.

다음과 같은 사용자 지정을 사용하여 데이터 내보내기를 생성할 수 있습니다.

- 리소스 ID 포함
- 분할 비용 할당 데이터
- 시간 단위 세분화
- 버전 관리
- 압축 유형 및 파일 형식

Amazon ECS 또는 Amazon EKS에서 컨테이너를 실행하는 워크로드의 경우, 분할 비용 할당 데이터를 활성화하면 컨테이너 워크로드가 공유 컴퓨팅 및 메모리 리소스를 사용하는 방식에 따라 컨테이너 비용을 개별 사업부 및 팀에 할당할 수 있습니다. 분할 비용 할당 데이터를 통해 AWS Cost and Usage Report에서 새로운 컨테이너 수준 리소스에 대한 비용 및 사용 데이터를 확인할 수 있습니다. 분할 비용 할당 데이터는 클러스터에서 실행되는 개별 ECS 서비스 및 작업의 비용을 기준으로 계산됩니다.

Cost and Usage Dashboard는 Cost and Usage Dashboard 테이블을 정기적으로 S3 버킷에 내보내고 사전 구축된 Cost and Usage Dashboard를 QuickSight에 배포합니다. 사용자 지정 기능 없이 비용 및 사용 데이터의 대시보드를 빠르게 배포하려는 경우 이 옵션을 사용합니다.

원하는 경우 레거시 모드에서 CUR을 계속 내보낼 수 있습니다. 이때 AWS Glue와 같은 기타 처리 서비스를 통합하여 분석용으로 데이터를 준비하고 Amazon Athena를 통해 SQL로 데이터를 쿼리하여 데이터 분석을 수행할 수 있습니다.

구현 단계

- 데이터 내보내기 생성: 원하는 데이터로 사용자 지정 내보내기를 생성하고 내보내기 스키마를 제어합니다. 기본 SQL을 사용하여 결제 및 비용 관리 데이터 내보내기를 생성하고 QuickSight와 통합하여 결제 및 비용 관리 데이터를 시각화합니다. 표준 모드에서 데이터를 내보내 Amazon Athena와 같은 기타 처리 도구를 사용하여 데이터를 분석할 수도 있습니다.
- 비용 및 사용 보고서 구성: 결제 콘솔을 사용하여 하나 이상의 비용 및 사용 보고서를 구성합니다. 모든 식별자 및 리소스 ID를 포함하는 시간 단위의 세분화로 보고서를 구성합니다. 다양한 세분화의 다른 보고서를 생성하여 더 개략적인 요약 정보를 제공할 수도 있습니다.
- Cost Explorer에서 시간별 세부 수준 구성: 지난 14일 동안의 시간별 세부 수준으로 비용 및 사용 데이터에 액세스하려면 결제 콘솔에서 시간별 및 리소스 수준 데이터를 활성화하는 방법을 고려합니다.
- 애플리케이션 로깅 구성: 애플리케이션에서 제공하는 각 비즈니스 성과를 추적하고 측정할 수 있도록 이에 대한 로깅을 확인합니다. 비용 및 사용량 데이터와 일치하도록 이 데이터의 세분화가 시간 단위 이상인지 확인합니다. 로깅 및 모니터링에 대한 자세한 내용은 Well-Architected 운영 우수성 원칙을 참조하세요.

리소스

관련 문서:

- AWS Data Exports
- AWS Glue
- QuickSight
- AWS 비용 관리 요금
- AWS 리소스에 태그 지정
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- Well-Architected 운영 우수성 요소

관련 예제:

- AWS Account Setup
- Data Exports for AWS Billing and Cost Management
- AWS Cost Explorer Common Use Cases

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 AWS에서 비용 및 사용 보고서에 이 정보를 추가합니다. 태그가 지정된 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 태그 편집기를 사용하면 여러 리소스의 태그를 추가, 삭제 및 관리할 수 있습니다. Tag Editor에서는 태그를 관리할 리소스를 검색한 후 검색 결과에 나온 리소스에서 바로 태그를 관리합니다.

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 분류하도록 범주 규칙을 정의합니다. 태그 지정에 더해 또 다른 수준의 관리 기능을 제공할 수 있습니다. 특정 계정과 태그를 여러 프로젝트에 매핑할 수도 있습니다.

구현 단계

- 태그 지정 체계 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.
- 리소스 태그 지정: 정의된 비용 속성 범주를 사용하여 범주에 따라 워크로드의 모든 리소스에 태그를 지정합니다. CLI, 태그 편집기, AWS Systems Manager 등의 도구를 사용하여 효율성을 높입니다.
- AWS Cost Categories 구현: 태그 지정을 구현하지 않고 비용 범주를 생성할 수 있습니다. 비용 범주는 기존의 비용 및 사용량 규모를 사용합니다. 스키마에서 범주 규칙을 생성하고, 비용 범주에 구현합니다.
- 태그 지정 자동화: 모든 리소스에서 많은 태그 지정을 유지하려면 리소스가 생성될 때 자동으로 태그가 지정되도록 태그 지정을 자동화합니다. 리소스가 생성될 때 리소스에 태그가 지정되었는지 확인하기 위해 AWS CloudFormation과 같은 서비스를 사용합니다. 또한 워크로드를 주기적으로 스캔하고 태그가 지정되지 않은 리소스를 제거하는 마이크로서비스를 사용하거나 Lambda 함수를 사용해 자동으로 태그를 지정하는 사용자 지정 솔루션을 생성할 수 있습니다. 이는 테스트 및 개발 환경에서 매우 유용합니다.
- 태그 지정 시 모니터링 및 보고: 조직 전체에서 높은 수준의 태그 지정을 유지하려면 워크로드 전체의 태그를 보고하고 모니터링합니다. AWS Cost Explorer를 사용하여 태그가 지정되거나 태그가 지정되지 않은 리소스의 비용을 보거나 태그 편집기와 같은 서비스를 사용할 수 있습니다. 태그가 지정되지 않은 리소스의 수를 정기적으로 검토하고 원하는 태그 지정 수준에 도달할 때까지 태그를 추가하는 작업을 수행합니다.

리소스

관련 문서:

- Tagging Best Practices
- AWS CloudFormation 리소스 태그
- AWS Cost Categories
- AWS 리소스에 태그 지정
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

관련 비디오:

- How can I tag my AWS resources to divide up my bill by cost center or project
- Tagging AWS Resources

COST03-BP03 비용 귀속 범주 식별

조직 내 비용을 내부 소비 주체에 할당하는 데 활용할 수 있는 조직 범주(예: 사업부, 부서, 프로젝트)를 파악합니다. 이러한 범주를 사용하여 지출에 대한 책임을 강화하고, 비용 인식을 제고하고, 효과적인 소비 행동을 유도하세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

비용 범주화 프로세스는 예산 책정, 회계, 재무 보고, 의사 결정, 벤치마킹, 프로젝트 관리에 매우 중요합니다. 지출을 분류하고 범주화하면 팀이 클라우드 여정 전반에서 발생하는 비용 유형을 더 잘 이해할 수 있어 정보에 입각한 결정을 내리고 효과적으로 예산을 관리할 수 있습니다.

클라우드 지출 책임은 엄격한 수요 및 비용 관리를 위한 강력한 인센티브를 제공합니다. 그 결과 클라우드 지출의 대부분을 소비 사업부 또는 팀에 할당하는 조직에서 클라우드 비용을 크게 절감할 수 있습니다. 또한 클라우드 지출을 할당하면 조직이 중앙화된 클라우드 거버넌스의 모범 사례를 더 많이 채택하는 데 도움이 됩니다.

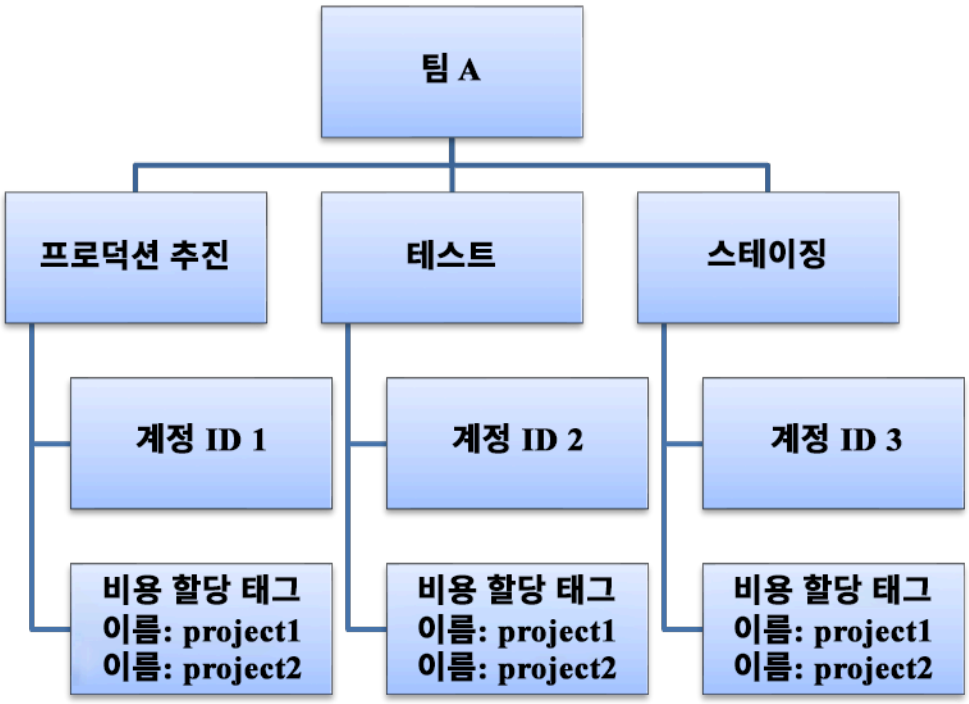
재무 팀 및 기타 이해관계자와 협력하여 정기적인 주기의 회의 중에 조직 내 비용 할당 방식 관련 요구사항을 파악합니다. 워크로드 비용은 개발, 테스트, 프로덕션 및 폐기를 포함한 전체 수명 주기에 걸쳐 할당되어야 합니다. 조직에서 학습, 직원 개발 및 아이디어 창출에 대해 발생한 비용의 귀속 방법을 파악합니다. 그러면 이 목적으로 사용되는 계정을 일반적인 IT 비용 예산 대신 교육 및 개발 예산에 올바르게 할당할 수 있습니다.

조직 내 이해관계자와 함께 비용 귀속 범주를 정의한 후 AWS Cost Categories를 사용하여 AWS 클라우드에서 비용 및 사용 정보를 특정 프로젝트 비용 또는 부서나 사업부에 대한 AWS 계정과 같은 유의미한 범주로 그룹화합니다. 계정, 태그, 서비스 또는 요금 유형과 같은 다양한 차원을 사용하여 정의한 규칙에 따라 사용자 지정 범주를 생성하고 비용과 사용량 정보를 이러한 범주에 매핑할 수 있습니다. 비용 범주가 설정되면 범주별로 비용과 사용량 정보를 확인할 수 있어 조직이 더 나은 전략 결정과 구매 결정을 내릴 수 있습니다. 이들 범주는 AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report에도 표시됩니다.

예를 들어, 사업부(DevOps 팀)에 대한 비용 범주를 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 대

한 규칙)을 생성합니다. 비용 범주를 사용하면 규칙 기반 엔진을 통해 비용을 정리할 수 있습니다. 사용자가 구성한 규칙이 비용을 범주로 정리합니다. 이들 규칙 내에서 각 범주에 대한 여러 차원(예: 특정 AWS 계정, AWS 서비스, 요금 유형)을 사용하여 필터링할 수 있습니다. 이후 AWS 결제 및 비용 관리 및 비용 관리 콘솔에서 여러 제품에 걸쳐 이러한 범주를 사용할 수 있습니다. 여기에 AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report 및 AWS Cost Anomaly Detection도 추가되었습니다.

예를 들어, 다음 다이어그램은 여러 팀(비용 범주), 여러 환경(규칙), 및 여러 리소스 또는 자산(차원)을 가진 각 환경을 구성하여 조직의 비용과 사용량 정보를 그룹화하는 방법을 보여줍니다.



비용 및 사용 조직도

비용 점주를 사용하여 비용 그룹을 만들 수도 있습니다. 비용 범주를 생성하면(사용 레코드에 대한 비용 범주를 생성한 후 값을 업데이트하기 위해 최대 24시간 허용) AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, AWS Cost Anomaly Detection에 나타납니다. AWS Cost Explorer 및 AWS Budgets에서 비용 범주가 추가적 청구 차원으로 표시됩니다. 이를 사용하여 특정 비용 범주 값을 필터링하거나 비용 범주를 기준으로 그룹화할 수 있습니다.

구현 단계

- 조직 범주 정의: 내부 이해관계자 및 사업부와 회의하여 조직의 구조 및 요구 사항을 반영하는 범주를 정의합니다. 이러한 범주는 사업부, 예산, 비용 센터 또는 부서와 같은 기존 재무 범주의 구조에 직

접 매핑되어야 합니다. 교육과 같이 클라우드가 비즈니스에 제공하는 성과를 살펴보세요. 이들은 조직 범주이기도 합니다.

- 직무 범주 정의: 내부 이해관계자 및 사업부와 회의하여 비즈니스 내에서 맡은 역할을 반영하는 범주를 정의합니다. 워크로드 또는 애플리케이션 이름과 프로덕션, 테스트, 개발 등의 환경 유형이 이에 해당될 수 있습니다.
- AWS 비용 범주 정의: 비용 범주를 생성하고 AWS Cost Categories를 사용하여 비용 및 사용 정보를 체계적으로 정리하고 AWS 비용 및 사용을 유의미한 범주에 매핑합니다. 한 리소스에 여러 범주를 할당할 수 있으며 리소스는 서로 다른 여러 범주에 있을 수 있으므로 범주화된 구조 내에서 비용을 관리할 수 있도록 필요한 만큼 범주를 정의합니다(AWS Cost Categories 사용).

리소스

관련 문서:

- AWS 리소스에 태그 지정
- 비용 할당 태그 사용
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- AWS Cost Categories
- Managing your costs with AWS Cost Categories
- Creating cost categories
- Tagging cost categories
- Splitting charges within cost categories
- AWS Cost Categories Features

관련 예제:

- Organize your cost and usage data with AWS Cost Categories
- Managing your costs with AWS Cost Categories

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 우선순위를 지정하거나 각 구성 요소에 대한 지표를 정의하고 추적할 수 있습니다. 팀과 협력하여 사용할 지표를 파악합니다. 이 단위는 워크로드의 효율성 또는 각 비즈니스 결과의 비용을 파악하는 데 사용됩니다.

구현 단계

- 워크로드 성과 정의: 비즈니스 이해관계자를 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.
- 워크크로드 구성 요소 성과 정의: 선택적으로 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 리소스에 태그 지정
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

COST03-BP05 결제 및 비용 관리 도구 구성

조직의 정책에 따라 비용 관리 도구를 구성하여 클라우드 지출을 관리하고 최적화합니다. 여기에는 비용 및 사용 데이터를 구성 및 추적하고, 통합 과금 및 액세스 권한을 통해 제어 기능을 강화하며, 예산

책정 및 예측을 통해 계획을 개선하고, 알림 또는 경고를 수신하며, 리소스 및 가격 최적화를 통해 비용을 절감하는 서비스, 툴 및 리소스가 포함됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

강한 책임감을 갖게 하려면 비용 배분 전략의 일환으로 계정 전략을 먼저 고려합니다. 이 단계를 제대로 해내면 더 이상의 조치는 필요 없을 수도 있습니다. 그렇지 않다면 인식이 결여되고 문제가 더 발생할 수 있습니다.

클라우드 지출에 대한 책임을 장려하려면 비용 및 사용량을 파악할 수 있는 도구에 대한 액세스 권한을 사용자에게 부여하세요. AWS에서는 다음과 같은 목적으로 모든 워크로드와 팀을 구성하는 것이 좋습니다.

- 구성: 자체 태그 전략 및 분류법을 사용하여 비용 할당 및 거버넌스 기준을 설정합니다. AWS Control Tower 또는 AWS 조직과 같은 도구를 사용하여 여러 AWS 계정을 생성합니다. 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 사업부 계정을 해당 비용 센터로 그룹화하면 사업 단위 책임자가 한 곳에서 여러 계정의 소비 내역을 볼 수 있습니다.
- 액세스: 통합 결제를 통해 조직 전체의 청구 정보를 추적합니다. 올바른 이해관계자 및 비즈니스 소유자가 액세스할 수 있는지 확인합니다.
- 제어: 적절한 가드레일로 효과적인 거버넌스 메커니즘을 구축하여 서비스 제어 정책(SCP), 태그 정책, IAM 정책 및 예산 알림을 사용할 때 예상치 못한 시나리오를 방지합니다. 예를 들어 효과적인 제어 메커니즘을 사용하여 팀이 선호 지역에서만 특정 리소스를 생성하도록 허용하고 특정 태그(예: 비용 센터) 없이 리소스가 생성되지 않도록 할 수 있습니다.
- 현재 상태: 현재 비용 및 사용량 수준을 보여주는 대시보드를 구성합니다. 대시보드는 운영 대시보드와 같이 작업 환경 내의 가시성이 높은 위치에서 사용할 수 있어야 합니다. 데이터를 내보내고 AWS Cost Optimization Hub 또는 지원되는 모든 제품에서 비용 및 사용 대시보드를 사용하여 이러한 가시성을 만들 수 있습니다. 다양한 페르소나에 대해 서로 다른 대시보드를 만들어야 할 수도 있습니다. 예를 들어 관리자 대시보드는 엔지니어링 대시보드와 다를 수 있습니다.
- 알림: 비용 또는 사용량이 정의된 한도를 초과하고 AWS Budgets 또는 AWS Cost Anomaly Detection에서 이상 징후가 발생할 경우 알림을 제공합니다.
- 보고서: 모든 비용 및 사용량 정보를 요약합니다. 상세하고 귀속 가능한 비용 데이터를 통해 클라우드 지출에 대한 인식과 책임성을 높입니다. 보고서를 사용하는 팀과 관련성이 높고 권장 사항이 포함된 보고서를 만듭니다.
- 추적: 구성된 목표 또는 목표를 기준으로 현재 비용 및 사용량을 보여줍니다.

- 분석: 팀원이 다양한 필터(리소스, 계정, 태그 등)를 사용하여 시간별, 일별 또는 월별 세부 수준까지 사용자 지정 및 심층 분석을 수행할 수 있습니다.
- 검사: 리소스 배포 및 비용 최적화 기회에 대한 최신 정보를 얻습니다. Amazon CloudWatch, Amazon SNS 또는 Amazon SES를 사용하여 조직 수준에서 리소스 배포에 대한 알림을 받을 수 있습니다. AWS Trusted Advisor 또는 AWS Compute Optimizer를 사용하여 비용 최적화 권장 사항을 검토하세요.
- 추세 보고서: 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.

AWS Cost Optimization Hub를 사용하면 중앙 위치에서 통합된 잠재적 비용 절감 기회를 파악하고 Amazon Athena 통합을 위한 데이터 내보내기를 생성할 수 있습니다. 또한 AWS Cost Optimization Hub를 사용하여 대화형 비용 분석 및 안전한 비용 인사이트 공유에 QuickSight를 활용할 수 있는 Cost and Usage Dashboard를 배포할 수 있습니다.

조직에 필수적인 기술 또는 역량이 없다면 AWS ProServ, AWS Managed Services(AMS) 또는 AWS 파트너와 협력할 수 있습니다. 서드파티 도구를 사용할 수도 있지만 반드시 가치 제안을 확인해야 합니다.

구현 단계

- 팀 기반 도구 액세스 허용: 계정을 구성하고, 도구 사용에 필수적인 비용 및 사용 보고서에 대한 액세스 권한이 있는 그룹을 만들어, AWS Identity and Access Management를 통해 AWS Cost Explorer와 같은 도구에 대한 액세스를 관리하는 데 사용합니다. 이 그룹에는 애플리케이션을 소유하거나 관리하는 모든 팀의 담당자가 포함되어야 합니다. 그래야 모든 팀이 비용 및 사용량 정보에 액세스하여 사용량을 추적할 수 있습니다.
- 비용 태그 및 범주 구성: 팀, 사업부, 애플리케이션, 환경, 프로젝트 전반의 비용을 체계적으로 정리합니다. 리소스 태그를 사용하여 비용 할당 태그별로 비용을 정리할 수 있습니다. 태그, 계정, 서비스 등을 사용하여 크기를 기반으로 비용 범주를 생성하여 비용을 매핑합니다.
- AWS Budgets 구성: 워크로드의 모든 계정에서 AWS Budgets를 구성합니다. 태그 및 비용 범주를 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 Anomaly Detection 구성: 생성한 계정, 핵심 서비스 또는 비용 범주에 대해 AWS Cost Anomaly Detection을 사용하여 비용과 사용량을 모니터링하고 비정상적인 지출을 탐지할 수 있습니

다. 집계된 보고서에서 개별적으로 알림을 수신하고, 이메일이나 Amazon SNS 주제로 알림을 수신하면 이상이 발생한 근본 원인을 분석하여 알아내고 비용을 높이는 요소를 찾아낼 수 있습니다.

- 비용 분석 도구 사용: 앞으로의 분석에 대한 비용 데이터를 시각화합기 위해 워크로드 및 계정에 대해 AWS Cost Explorer를 구성합니다. 전체 지출, 워크로드의 주요 사용량 지표 및 과거 비용 데이터를 기반으로 미래의 비용을 예측하는 워크로드 대시보드를 만듭니다.
- 비용 절감 분석 도구 사용: AWS Cost Optimization Hub를 사용하여 미사용 리소스 삭제, 적정 규모 조정, 절감형 플랜, 예약, 컴퓨팅 최적화 도구 권장 사항 등 맞춤형 권장 사항을 통해 비용 절감 기회를 식별할 수 있습니다.
- 고급 도구 구성: 선택적으로 시각적 객체를 생성하여 대화형 분석 및 비용 인사이트 공유를 쉽게 수행할 수 있습니다. AWS Cost Optimization Hub의 데이터 내보내기를 사용하면 QuickSight를 기반으로 조직에 필요한 Cost and Usage Dashboard를 생성하여 추가 세부 정보와 세부 수준을 제공할 수 있습니다. 또한 고급 쿼리에 대해 Amazon Athena의 데이터 내보내기를 사용하여 고급 분석 기능을 구현하고 QuickSight에서 대시보드를 만들 수 있습니다. AWS 파트너와 협력하여 통합 클라우드 청구서 모니터링 및 최적화를 위한 클라우드 관리 솔루션을 채택합니다.

리소스

관련 문서:

- What is AWS 결제 및 비용 관리 and Cost Management?
- 모범 사례 AWS 환경 설정
- Best Practices for Tagging AWS Resources
- AWS 리소스에 태그 지정
- AWS Cost Categories
- Analyzing your costs with AWS Budgets
- Analyzing your costs with AWS Cost Explorer
- What is AWS Data Exports?

관련 비디오:

- Deploying Cloud Intelligence Dashboards
- Get Alerts on any FinOps or Cost Optimization Metric or KPI

관련 예제:

- Cost and Usage Dashboard powered by QuickSight
- AWS Cost and Usage Governance 워크숍

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 않을 경우 노출되는 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 모니터링합니다. 이 데이터를 워크로드 비용과 결합합니다. 워크로드 비용은 특정 태그 값 또는 계정 ID에 연결된 비용을 조회하여 확인할 수 있습니다. 시간별 분석을 수행합니다. 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.

Amazon Elastic Container Service(Amazon ECS) 및 Amazon API Gateway의 컨테이너식 애플리케이션과 같은 리소스에 비해, 공유 리소스에 대한 워크로드 지표를 생성하는 작업은 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 애플리케이션 비용을 다시 할당할 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 리소스에 태그 지정
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

관련 예제:

- AWS 분할 비용 할당 데이터를 사용하여 Amazon ECS 및 AWS Batch의 비용 가시성 향상

COST 4. 리소스를 어떻게 폐기하나요?

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

모범 사례

- COST04-BP01 수명 주기 동안 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 주기 동안 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 실행

할 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 예로, 폐기를 위해 특정 시간이나 기간을 정의하기 위해 삭제할 항목의 태그 키 이름 및 값과 같은 리소스에 대한 `LifeSpan` 또는 `TTL`을 사용할 수 있습니다.

구현 단계

- 태그 지정 체계 구현: 리소스가 속한 워크로드를 식별하는 태그 지정 체계를 구현하여 워크로드 내의 모든 리소스에 태그가 적절히 지정되도록 합니다. 태그 지정을 사용하면 목적, 팀, 환경 또는 비즈니스에 맞는 다른 기준에 따라 리소스를 분류할 수 있습니다. 태그 지정 사용 사례, 전략 및 기술에 대한 자세한 내용은 AWS Tagging Best Practices를 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. 미사용 리소스 또는 활용률이 낮은 리소스에 대한 자세한 내용은 AWS Trusted Advisor Cost Optimization checks를 참조하세요.
- AWS 리소스 그룹화: AWS 리소스에 대한 그룹을 생성합니다. AWS Resource Groups를 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. 태그 편집기를 사용하여 지원되는 리소스에 태그를 대량으로 지정합니다. AWS Service Catalog를 사용하여 승인된 제품으로 구성된 포트폴리오를 생성 및 관리하고 최종 사용자에게 배포하며 제품 수명 주기를 관리하는 방법을 고려하세요.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor Cost Optimization Checks
- AWS 리소스에 태그 지정
- 사용자 지정 지표 게시

관련 비디오:

- How to optimize costs using AWS Trusted Advisor

관련 예제:

- AWS 리소스 구성
- AWS Trusted Advisor를 사용하여 비용 최적화

COST04-BP02 폐기 프로세스 구현

미사용 리소스를 식별하고 폐기하는 프로세스를 구현합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 시작되었음을 알립니다.

  다음 폐기 단계를 사용하여 프로세스의 일부로 무엇을 확인해야 하는지 알 수 있습니다.

  - 폐기할 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 커뮤니케이션: 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: 필요한 경우 리소스를 재배포할 수 있도록 리소스가 AWS CloudFormation, Terraform, AWS 클라우드 개발 키트 (AWS CDK) 또는 기타 코드형 인프라 배포 도구를 통해 배포되었는지를 확인합니다.
  - 액세스 방지: 일정 기간 제한적인 제어를 적용하여 리소스가 필요한지 확인하는 동안 리소스 사용을 방지합니다. 필요한 경우 리소스 환경을 원래 상태로 되돌릴 수 있는지 확인합니다.

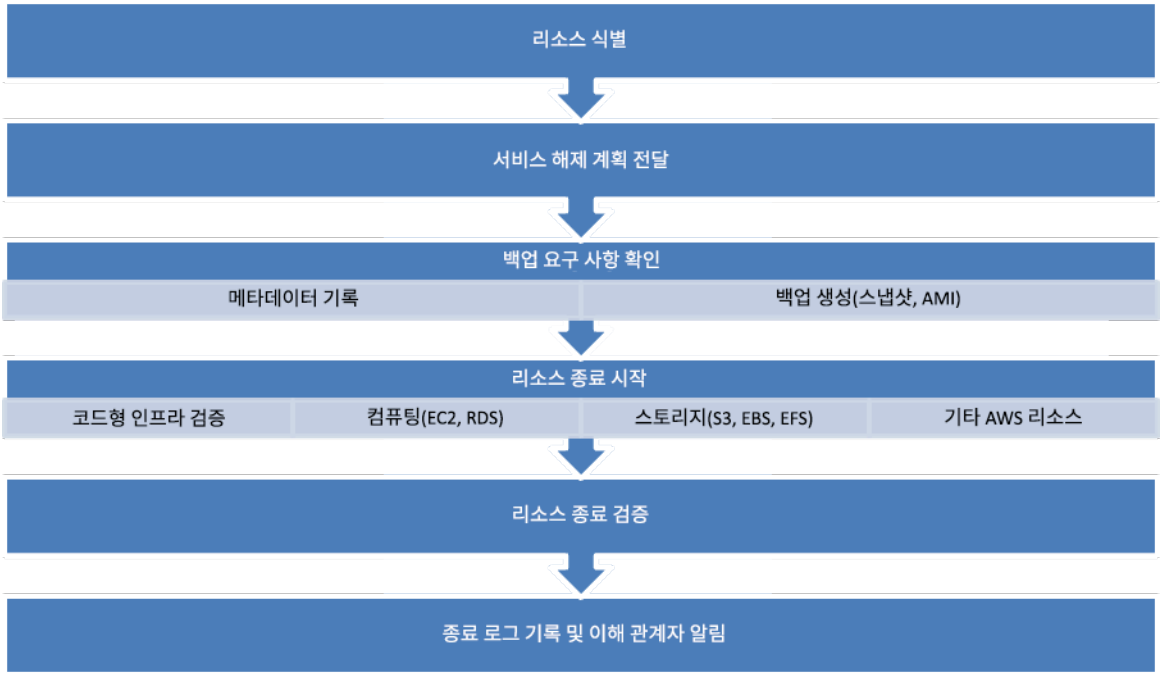
- 내부 폐기 프로세스 준수: 조직의 관리 작업 및 폐기 프로세스를 따릅니다. 여기에는 조직 도메인에서 리소스 제거, DNS 레코드 제거, 구성 관리 도구, 모니터링 도구, 자동화 도구 및 보안 도구에서 리소스 제거 등이 포함됩니다.

리소스가 Amazon EC2 인스턴스인 경우 다음 목록을 참조합니다. Amazon EC2 리소스를 삭제하거나 종료하려면 어떻게 해야 하나요?

- 모든 Amazon EC2 인스턴스 및 로드 밸런서를 중지하거나 종료합니다. Amazon EC2 인스턴스는 종료된 후에 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 지속 시간은 사용된 스토리지 클래스에 따라 다릅니다. 각 스토리지 클래스의 최소 스토리지 기간에 대한 요약은 Amazon S3 스토리지 클래스 전반에 걸친 성능을 참조하세요. 조기 삭제 요금 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 비디오:

- Delete CloudFormation stack but retain some resources
- Amazon EC2 인스턴스를 시작한 사용자 식별

관련 예제:

- Amazon EC2 리소스 삭제 또는 종료
- Amazon EC2 인스턴스를 시작한 사용자 식별

COST04-BP03 리소스 폐기

정기적 감사 또는 사용량 변화와 같은 이벤트에 의해 시작된 리소스를 폐기합니다. 폐기는 일반적으로 정기적으로 수행되며 수동 또는 자동으로 수행할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

미사용 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 시작할 수 있습니다. 검색 및 폐기 이벤트는 시장 상황의 변화나 제품 종료와 같은 외부 이벤트에 의해 시작될 수도 있습니다.

구현 단계

- 리소스 폐기: 더 이상 필요하지 않거나 라이선스 계약이 종료된 AWS 리소스의 사용 중지 단계에 해당합니다. 스냅샷 생성 또는 백업과 같은 원치 않는 중단이 발생하지 않도록, 폐기 단계 및 리소스 폐기로 진행하기 전에 최종 확인을 완료해야 합니다. 폐기 프로세스를 사용하여, 미사용 리소스로 식별된 각 리소스를 폐기합니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Trusted Advisor

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

자동화를 사용하여 폐기 프로세스와 관련된 비용을 줄이거나 제거합니다. 자동 폐기를 수행하도록 워크로드를 설계하면 수명 주기 동안 전체 워크로드 비용을 절감할 수 있습니다. Amazon EC2 Auto Scaling 또는 Application Auto Scaling을 사용하여 폐기 프로세스를 수행할 수 있습니다. API 또는 SDK를 사용하여 워크로드 리소스를 자동으로 폐기하는 사용자 지정 코드를 구현할 수도 있습니다.

최신 애플리케이션은 서버리스 서비스 채택을 우선시하는 전략인 서버리스 우선으로 구축됩니다. AWS에서는 스택의 세 계층(컴퓨팅, 통합, 데이터 저장소) 모두를 위한 서버리스 서비스를 개발했습니다. 서버리스 아키텍처를 사용하면 자동 스케일 업 및 스케일 다운을 통해 트래픽이 적은 기간에 비용을 절감할 수 있습니다.

구현 단계

- Amazon EC2 Auto Scaling 또는 Application Auto Scaling 구현: 지원되는 리소스의 경우 Amazon EC2 Auto Scaling 또는 Application Auto Scaling으로 구성합니다. 이러한 서비스는 AWS 서비스를 이용할 때 사용률과 비용 효율성을 최적화하는 데 도움이 될 수 있습니다. 수요가 낮아지면 이러한 서비스는 과도한 지출을 방지할 수 있도록 모든 초과 리소스 용량을 자동으로 제거합니다.
- 인스턴스를 종료하도록 CloudWatch 구성: CloudWatch 경보를 사용하여 종료하도록 인스턴스를 구성할 수 있습니다. 폐기 프로세스의 지표를 사용하여 Amazon Elastic Compute Cloud 작업으로 경보를 구현합니다. 롤아웃하기 전에 비프로덕션 환경에서 작업을 확인합니다.
- 워크로드 내에 코드 구현: AWS SDK 또는 AWS CLI를 사용하거나 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 AWS에서 서버리스 아키텍처 및 이벤트 기반 아키텍처 구축을 우선시합니다. AWS에서는 자동으로 최적화된 리소스 사용률 및 자동화된 폐기(스케일 인 및 스케일 아웃) 기능을 기본적으로 제공하는 여러 서버리스 기술 서비스를 제공합니다. 서버리스 애플리케이션을 통해 리소스 사용률이 자동으로 최적화되고 과다 프로비저닝에 대한 비용을 지불할 필요가 없습니다.

리소스

관련 문서:

- Amazon EC2 Auto Scaling
- Getting Started with Amazon EC2 Auto Scaling
- Application Auto Scaling
- AWS Trusted Advisor
- AWS의 서버리스
- 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성
- Amazon CloudWatch 경보에 종료 작업 추가하기

관련 예제:

- Scheduling automatic deletion of AWS CloudFormation stacks

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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. API 또는 SDK를 사용하여 사용자 지정 코드를 구현해 자동으로 삭제되는 객체에 대한 수명 주기 정책 및 정책 규칙을 생성할 수도 있습니다.

구현 단계

- Amazon Data Lifecycle Manager 사용: Amazon Data Lifecycle Manager의 수명 주기 정책을 사용하여 Amazon EBS 스냅샷 및 Amazon EBS 지원 AMI의 삭제를 자동화합니다.
- 버킷에서 수명 주기 구성 설정: 버킷에서 Amazon S3 수명 주기 구성을 사용하여 비즈니스 요구 사항에 따라 객체 수명 주기 동안 Amazon S3에서 수행할 작업과 객체 수명 주기 종료 시 삭제를 정의합니다.

리소스

관련 문서:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- Amazon S3 버킷에서 수명 주기 구성 설정

관련 비디오:

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager

- Empty an Amazon S3 bucket using a lifecycle configuration rule

관련 예제:

- Empty an Amazon S3 bucket using a lifecycle configuration rule

비용 효율적인 리소스

Questions

- COST 5. 서비스를 선택할 때 비용을 어떻게 평가하나요?
- COST 6. 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성하나요?
- COST 7. 비용 절감을 위해 요금 모델을 어떻게 사용하나요?
- COST 8. 데이터 전송 요금을 위한 계획은 어떻게 되나요?

COST 5. 서비스를 선택할 때 비용을 어떻게 평가하나요?

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, 애플리케이션 및 비즈니스 관련 활동에 집중할 수 있습니다.

모범 사례

- COST05-BP01 조직의 비용 요구 사항 파악
- COST05-BP02 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, 신뢰성) 사이의 균형을 정의합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

대부분의 조직에서 정보 기술(IT) 부서는 여러 소규모 팀으로 구성되어 있으며, 팀마다 팀원의 전문성과 기술을 반영하여 고유한 과제와 중점 영역을 가지고 있습니다. 조직의 전반적인 목표, 우선순위, 세부 목표 및 각 부서 또는 프로젝트가 이러한 전반적인 목표에 어떻게 기여하는지 이해해야 합니다. 인력, 장비, 기술, 자재 및 외부 서비스를 포함한 모든 필수 리소스를 분류하는 것은 조직 목표를 달성하고 종합적인 예산 계획을 수립하는 데 매우 중요합니다. 비용 식별 및 이해를 위해 이러한 체계적인 접근 방식을 채택하는 것은 조직에서 현실적이고 강력한 비용 계획을 수립하는 데 필수적입니다.

워크로드에 사용할 서비스를 선택할 때는 조직의 우선순위를 이해하는 것이 중요합니다. 비용 최적화와 기타 AWS Well-Architected Framework 원칙(예: 성능 및 신뢰성) 사이에서 균형을 맞춥니다. 이 프로세스는 조직의 목표, 시장 상황 및 운영 역학의 변화를 반영하기 위해 체계적이고 정기적으로 수행되어야 합니다. 완전한 비용 최적화 워크로드는 조직의 요구 사항과 가장 일치하는 솔루션을 의미하며 꼭 비용이 가장 낮은 솔루션이 아닐 수 있습니다. 조직 내 모든 팀(예: 제품, 비즈니스, 기술, 재무)과 회의를 통해 정보를 수집합니다. 주력할 영역을 결정하거나 수행할 조치를 선택할 때 정보를 토대로 결정을 내릴 수 있도록 상충하는 이해관계나 대안 사이에서 장단점의 영향을 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.

구현 단계

- 조직의 비용 요구 사항 파악: 제품 관리, 애플리케이션 소유자, 개발 및 운영 팀, 관리 및 재무 역할 등 조직의 다양한 팀원을 만납니다. 이 워크로드와 그 구성 요소에 대해 Well-Architected 원칙의 우선순위를 정합니다. 원칙을 순서대로 나열해야 합니다. 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.
- 기술적 부채 해결 및 문서화: 워크로드 검토 중에 기술 부채를 해결합니다. 백로그 항목을 문서화하여 향후 워크로드를 재검토합니다. 워크로드를 리팩터링 또는 다시 설계하여 한층 더 최적화하는 것을 목표로 합니다. 장단점을 다른 이해관계자에게 명확하게 전달하는 것이 중요합니다.

리소스

관련 모범 사례:

- REL11-BP07 가용성 목표 및 가동 시간 서비스 수준에 관한 계약(SLA)을 충족하도록 제품 설계
- OPS01-BP06 장단점 평가

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- Amazon S3 storage classes
- 클라우드 제품

COST05-BP02 워크로드의 모든 구성 요소 분석

현재 크기나 비용을 막론하고 모든 워크로드 구성 요소가 분석되도록 해야 합니다. 검토 작업은 현재 비용과 예상 비용 등의 잠재적인 이점을 반영해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

조직에 비즈니스 가치를 제공하도록 설계된 워크로드 구성 요소에는 다양한 서비스가 포함될 수 있습니다. 각 구성 요소에 대해 비즈니스 요구 사항을 해결하기 위해 특정 AWS 클라우드 서비스를 선택할 수도 있습니다. 이 선택은 이러한 서비스에 대한 친숙도나 사용 경험 등 여러 요인의 영향을 받을 수 있습니다.

COST05-BP01 조직의 비용 요구 사항 파악에서 언급한 대로, 조직의 요구 사항을 파악한 후 워크로드의 모든 구성 요소를 철저히 분석합니다. 현재 및 예상 비용과 크기를 고려하여 각 구성 요소를 분석합니다. 수명 주기 동안 발생할 수 있는 워크로드 절감 효과와 비교하여 분석 비용을 고려합니다. 이 워크로드의 모든 구성 요소를 분석하는 데 드는 노력은 구성 요소를 최적화함으로써 예상되는 잠재적 절감 또는 개선 효과에 상응해야 합니다. 예를 들어 제안된 리소스의 비용이 월 10 USD이고 과소 예측된 로드가 월 15 USD를 초과하지 않는 경우, 비용을 50%(월 5 USD) 절감하기 위해 들여야 하는 하루분의 노력이 시스템 수명 동안 얻을 수 있는 잠재적 이점보다 더 클 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. 예를 들어 처음에는 EMR 서버리스를 사용하는 것이 적절할 수 있습니다. 그러나 이 서비스의 소비가 증가함에 따라 EMR on EC2로 전환하면 워크로드의 해당 구성 요소에 대한 비용을 줄일 수 있습니다.

AWS Cost Explorer 및 AWS Cost and Usage Report(CUR)를 사용하여 개념 증명(PoC) 또는 환경 실행 비용을 분석할 수 있습니다. AWS Pricing Calculator를 사용하여 워크로드 비용을 추정할 수도 있습니다.

기술 팀이 워크로드를 검토하기 위해 따라야 할 워크플로를 작성하세요. 이 워크플로를 단순하게 유지하되, 팀이 워크로드의 각 구성 요소와 가격을 이해할 수 있도록 필요한 모든 단계를 다루어야 합니다. 그러면 조직은 각 팀의 특정 요구 사항에 따라 이 워크플로를 따르고 사용자 지정할 수 있습니다.

1. 워크로드에 사용 중인 각 서비스 나열: 이는 좋은 출발점입니다. 현재 사용 중인 모든 서비스와 비용의 출처를 파악하세요.
2. 해당 서비스의 요금 책정 방식 이해: 각 서비스의 요금 모델을 이해합니다. AWS 서비스마다 사용량, 데이터 전송, 기능별 요금 등의 요소에 따라 가격 책정 모델이 다릅니다.
3. 예상치 못한 워크로드 비용이 발생하고 예상 사용량 및 비즈니스 성과와 일치하지 않는 서비스에 집중: AWS Cost Explorer 또는 AWS Cost and Usage Report를 사용하여 비용이 가치나 사용량에 비례하지 않는 이상치 또는 서비스를 식별합니다. 비용을 비즈니스 성과와 연관시켜 최적화 작업의 우선순위를 정하는 것이 중요합니다.
4. AWS Cost Explorer, CloudWatch Logs, VPC 흐름 로그 및 Amazon S3 Storage Lens를 통해 이러한 고비용의 근본 원인 파악: 이러한 도구는 고비용을 진단하는 데 중요한 역할을 합니다. 각 서비스는 사용량과 비용을 확인하고 분석하기 위한 다양한 렌즈를 제공합니다. 예를 들어 Cost Explorer에서는 전체 비용 추세를 파악할 수 있고, CloudWatch Logs는 운영 인사이트를 제공하며, VPC 흐름 로그는 IP 트래픽을 표시하고, Amazon S3 Storage Lens는 스토리지 분석에 유용합니다.
5. AWS Budgets를 사용하여 서비스 또는 계정의 일정 금액에 대한 예산 설정: 예산을 설정하면 비용을 사전에 관리할 수 있습니다. AWS Budgets를 사용하여 사용자 지정 예산 임곗값을 설정하고 비용이 임곗값을 초과할 경우 알림을 받습니다.
6. Amazon CloudWatch 경보를 구성하여 청구 및 사용 알림 전송: 비용 및 사용량 지표에 대한 모니터링 및 알림을 설정합니다. CloudWatch 경보를 통해 특정 임곗값 위반 시 이를 알릴 수 있으므로 개입 응답 시간이 향상됩니다.

현재 속성에 관계없이 모든 워크로드 구성 요소에 대한 전략적 검토를 통해 시간이 지남에 따라 눈에 띄는 개선과 비용 절감이 이루어지도록 합니다. 이 검토 프로세스에 투자하는 노력은 실현될 수 있는 잠재적 이점을 주의 깊게 고려하여 신중하게 이루어져야 합니다.

구현 단계

- 워크로드 구성 요소 나열: 워크로드 구성 요소 목록을 작성합니다. 이 목록을 사용하여 각 구성요소가 분석되었는지 확인할 수 있습니다. 여기에 드는 노력은 조직의 우선순위에 정의된 워크로드에 대한 중요도를 반영해야 합니다. 리소스를 그룹화하면 프로덕션 데이터베이스 스토리지의 경우처럼 데이터베이스가 여러 개인 경우 효율성이 향상됩니다.

- 구성 요소 목록의 우선순위 지정: 구성 요소 목록을 가져와서 작업량순으로 우선순위를 지정합니다. 일반적으로 구성 요소의 비용(가장 비싼 것부터 가장 싼 것까지) 또는 조직 우선순위에 정의된 중요도의 순서를 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 storage classes
- AWS 클라우드 제품

관련 비디오:

- AWS Cost Optimization Series: CloudWatch

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 사례가 확립되지 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 앤 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(RDS) 등과 같은 서비스형 데이터베이스 플랫폼으로 마이그레이션하거나 AWS Elastic Beanstalk과 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 클러스터나 Amazon Redshift 클러스터의 노드 수를 늘리거나 줄여서 스케일 아웃 또는 스케일 인할 수 있습니다.

또한 AWS 리소스의 여러 인스턴스를 압축하여 리소스 사용 밀도를 높일 수도 있습니다. 예를 들어 단일 Amazon Relational Database Service(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, Amazon ElastiCache에서는 관리형 데이터베이스 서비스를 제공합니다. Amazon Athena, Amazon EMR, 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 EC2 기반 Apache Spark를 실행하는 것에 비해 Amazon RDS 옵션(AWS 관리형 서비스) 또는 Amazon EMR을 사용하는 경우를 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 storage classes
- AWS 클라우드 제품
- AWS Shared Responsibility Model

관련 비디오:

- Why move to a managed database?
- What is Amazon EMR and how can I use it for processing data?

관련 예제:

- Why to move to a managed database
- Consolidate data from identical SQL Server databases into a single Amazon RDS for SQL Server database using AWS DMS
- Deliver data at scale to Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Migrate an ASP.NET web application to AWS Elastic Beanstalk

COST05-BP04 비용 효율적인 라이선스가 포함된 소프트웨어 선택

오픈 소스 소프트웨어에는 워크로드 비용에서 상당한 부분을 차지할 수 있는 소프트웨어 라이선스 비용이 없습니다. 라이선스가 부여된 소프트웨어가 필요한 경우 CPU와 같은 임의의 속성에 바인딩된 라이선스를 피하고 결과 또는 성과에 바인딩된 라이선스를 찾습니다. 이러한 라이선스의 비용은 해당 라이선스가 제공하는 혜택에 더 근접하게 조정됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

오픈 소스는 소프트웨어가 특정 무료 배포 기준을 준수함을 알리는 소프트웨어 개발이라는 맥락에서 시작되었습니다. 오픈 소스 소프트웨어는 누구나 검사, 수정, 개선할 수 있는 소스 코드로 구성됩니다. 라이선스 비용을 최소화하기 위해 비즈니스 요구 사항, 엔지니어의 역량, 예상 사용량 또는 기타 기술 종속성에 따라 AWS에서 오픈 소스 소프트웨어를 사용하는 것을 고려할 수 있습니다. 즉, 오픈 소스 소프트웨어를 사용하면 소프트웨어 라이선스 비용을 줄일 수 있습니다. 라이선스 비용은 워크로드 규모가 확장됨에 따라 워크로드 비용에 상당한 영향을 미칠 수 있습니다.

총 비용 대비 라이선스 소프트웨어의 이점을 측정하여 워크로드를 최적화하세요. 라이선스 변경 사항과 이러한 변경 사항이 워크로드 비용에 미치는 영향을 모델링합니다. 공급업체가 데이터베이스 라이선스 비용을 변경하는 경우 이 비용이 워크로드의 전반적인 효율성에 어떤 영향을 미치는지 조사합니다. 공급업체의 기간별 요금 발표를 고려하여 제품 전반의 라이선스 변경 추세를 파악합니다. 또한 라이선스 비용은 하드웨어에 따라 확장되는 라이선스(CPU 바인딩 라이선스)와 같이 처리량이나 사용량에 관계없이 확장될 수 있습니다. 이러한 라이선스는 해당하는 결과 없이 비용이 빠르게 증가할 수 있으므로 피해야 합니다.

예를 들어 Linux 운영 체제로 us-east-1에서 Amazon EC2 인스턴스를 운영하면 Windows에서 실행되는 다른 Amazon EC2 인스턴스를 실행하는 것에 비해 비용을 약 45% 절감할 수 있습니다.

AWS Pricing Calculator는 Amazon RDS 인스턴스 및 다양한 데이터베이스 엔진과 같은 다양한 라이선스 옵션을 사용하여 다양한 리소스의 비용을 비교할 수 있는 포괄적인 방법을 제공합니다. 또한 AWS Cost Explorer는 기존 워크로드, 특히 다른 라이선스와 함께 제공되는 워크로드의 비용에 대해 매우 유용한 관점을 제공합니다. 라이선스 관리를 위해 AWS License Manager에서는 소프트웨어 라이선스를 감독하고 처리할 수 있는 간소화된 방법을 제공합니다. AWS 클라우드에서 선호하는 오픈 소스 소프트웨어를 배포하고 운영할 수 있습니다.

구현 단계

- 라이선스 옵션 분석: 사용 가능한 소프트웨어의 라이선스 약관을 검토합니다. 필요한 기능을 갖춘 오픈 소스 버전을 찾고 라이선스가 부여된 소프트웨어의 혜택이 비용보다 큰지 확인합니다. 소프트웨어에서 제공하는 이점과 소프트웨어의 비용이 일치하는 것이 유리한 약관입니다.
- 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 찾습니다. 또한 공급업체가 감사를 수행하는 방법과 부과될 수 있는 페널티를 알아봅니다.

리소스

관련 문서:

- Open Source at AWS
- AWS 총 소유 비용(TCO) 계산기
- Amazon S3 storage classes
- 클라우드 제품

관련 예제:

- Open Source Blogs
- AWS Open Source Blogs
- Optimization and Licensing Assessment

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

모든 구성 요소를 선택할 때 서비스 및 옵션 비용을 고려합니다. 이 과정에서 Amazon Relational Database Service(RDS), Amazon DynamoDB, Amazon Simple Notification Service(SNS), 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: 정적 웹 사이트의 경우 AWS Lambda 및 Amazon Simple Storage Service(S3)). 가능하면 애플리케이션을 컨테이너화하고 Amazon Elastic Container Service(Amazon ECS) 또는 Amazon Elastic Kubernetes Service(Amazon EKS)와 같은 AWS 관리형 컨테이너를 사용합니다.

오픈 소스 소프트웨어 또는 라이선스 요금이 없는 소프트웨어를 사용하여 라이선스 비용을 최소화합니다(예: 컴퓨팅 워크로드용 Amazon Linux 또는 Amazon Aurora로 데이터베이스 마이그레이션).

Lambda, Amazon Simple Queue Service(Amazon SQS), Amazon SNS, Amazon SES와 같은 서버리스 또는 애플리케이션 수준 서비스를 사용할 수 있습니다. 이러한 서비스를 사용하면 리소스를 관리할 필요가 없으며 코드 실행, 대기열 서비스 및 메시지 전송 기능이 제공됩니다. 또 다른 이점은 사용량에 따라 성능과 비용이 조정되므로 효율적인 비용 할당 및 귀속이 가능하다는 것입니다.

이벤트 기반 아키텍처 사용은 서버리스 서비스에서도 가능합니다. 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 폴링 용량, SSL/TLS 핸드셰이크가 줄어듭니다.

서버리스에 대한 자세한 내용은 Well-Architected Serverless Application Lens 백서를 참조하세요.

구현 단계

- 각 서비스를 선택하여 비용 최적화: 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 검토해야 하는 경우 인스턴스 크기를 늘리거나 Amazon ElastiCache 서비스(Redis 또는 Memcached)를 사용하여 데이터베이스에 캐시된 메커니즘을 제공하는 것을 고려하세요.
- 이벤트 기반 아키텍처 평가: 서버리스 아키텍처를 사용하면 분산된 마이크로서비스 기반 애플리케이션을 위한 이벤트 기반 아키텍처를 구축할 수도 있습니다. 이러한 아키텍처는 확장 가능하고 복원력이 있으며 민첩하고 비용 효과적인 솔루션을 구축하는 데 도움이 됩니다.

리소스

관련 문서:

- AWS 총 소유 비용(TCO) 계산기
- AWS Serverless
- 이벤트 기반 아키텍처(EDA)란 무엇인가요?
- Amazon S3 storage classes
- 클라우드 제품
- Amazon ElastiCache (Redis OSS)

관련 예제:

- Getting started with event-driven architecture
- 이벤트 중심 아키텍처
- How Statsig runs 100x more cost-effectively using Amazon ElastiCache (Redis OSS)
- 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 다시 시작해야 합니다. 사용량의 큰 변화는 대체 서비스가 더 적합하다는 의미일 수 있습니다.

데이터를 AWS 클라우드로 이전해야 하는 경우 데이터세트가 파일, 데이터베이스, 머신 이미지, 블록 볼륨 또는 테이프 백업이든 상관없이 AWS에서 제공하는 광범위한 서비스 및 파트너 도구를 사용하여 데이터세트를 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 최솟값을 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 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 사용 패턴과 예측 사용 패턴을 문서화합니다. 비즈니스 이해관계자와 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 storage classes
- 클라우드 제품
- Amazon EC2 Auto Scaling
- Cloud Data Migration
- AWS Snow Family

관련 비디오:

- AWS OpsHub for Snow Family

COST 6. 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성하나요?

진행 중인 작업에 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-BP04 공유 리소스 사용 고려

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 Framework 성능 효율성 원칙의 검토 섹션을 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 여러 서비스가 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 대한 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
- 올바른 크기 조정을 위한 기회 식별
- Amazon CloudWatch의 기능
- Cost Optimization: Amazon EC2 Right Sizing
- AWS Compute Optimizer
- AWS 요금 계산기

관련 예제:

- Perform a Data-Driven Cost Modelling
- 계획된 AWS 리소스 구성 비용 예측
- Choose the right AWS tools

COST06-BP02 데이터를 기준으로 리소스 유형, 크기, 개수 선택

워크로드 및 리소스 특성에 대한 데이터를 기준으로 리소스 크기나 유형을 선택합니다. 예를 들어, 컴퓨팅, 메모리, 처리량, 쓰기 집약형과 같은 기준이 적용됩니다. 일반적으로는 워크로드의 이전(온프레미스) 버전, 설명서 또는 워크로드와 관련된 기타 정보 출처를 사용해 리소스 사용량을 선택합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

Amazon EC2는 다양한 사용 사례에 맞게 선택할 수 있도록 CPU, 메모리, 스토리지 및 네트워킹 용량 수준이 서로 다른 광범위한 인스턴스 유형을 제공합니다. 이러한 인스턴스 유형에는 CPU, 메모리, 스토리지, 네트워킹 기능이 서로 다르게 조합되어 있어 프로젝트에 적합한 리소스 조합을 다양하게 선택할 수 있습니다. 인스턴스 유형마다 크기가 다양하므로 워크로드의 수요에 따라 리소스를 조정할 수 있습니다. 필요한 인스턴스 유형을 결정하려면 인스턴스에서 실행하려는 애플리케이션 또는 소프트웨어의 시스템 요구 사항에 관한 세부 정보를 수집해야 합니다. 이러한 세부 정보에는 다음이 포함되어야 합니다.

- 운영 체제
- CPU 코어 수
- GPU 코어
- 시스템 메모리(RAM) 용량
- 스토리지 유형 및 공간
- 네트워크 대역폭 요구 사항

컴퓨팅 요구 사항의 목적과 필요한 인스턴스를 파악한 다음 다양한 Amazon EC2 인스턴스 패밀리를 살펴봅니다. Amazon은 다음 인스턴스 유형 패밀리를 제공합니다.

- 범용
- 컴퓨팅 최적화
- 메모리 최적화
- 스토리지 최적화
- 가속 컴퓨팅
- HPC 최적화

특정 Amazon EC2 인스턴스 패밀리가 충족할 수 있는 구체적인 목적과 사용 사례를 더 자세히 이해하려면 AWS 인스턴스 유형을 참조하세요.

시스템 요구 사항 수집은 요구 사항에 가장 적합한 구체적인 인스턴스 패밀리와 인스턴스 유형을 선택하는 데 매우 중요합니다. 인스턴스 유형 이름은 패밀리 이름과 인스턴스 크기로 구성됩니다. 예를 들어 t2.micro 인스턴스는 T2 패밀리에 속하며 마이크로 크기입니다.

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 또는 게시된 솔루션)를 사용하여 선택합니다. AWS 요금 계산기 또는 비용 관리 도구를 사용하면 정보를 바탕으로 인스턴스 유형, 크기 및 구성에 대해 결정을 내리는 데 도움이 될 수 있습니다.

구현 단계

- 데이터를 기반으로 리소스 선택: 비용 모델링 데이터를 사용하여 예상 워크로드 사용 수준을 선택하고 지정된 리소스 유형과 크기를 선택합니다. 비용 모델링 데이터를 기반으로 인스턴스에 필요한 데이터 전송 속도를 고려하여 가상 CPU 수, 총 메모리(GiB), 로컬 인스턴스 스토어 볼륨(GB), Amazon EBS 볼륨, 네트워크 성능 수준을 결정합니다. 항상 상세한 분석과 정확한 데이터를 기반으로 선택하여 비용을 효과적으로 관리하는 동시에 성능을 최적화합니다.

리소스

관련 문서:

- AWS 인스턴스 유형
- AWS Auto Scaling
- Amazon CloudWatch의 기능
- Cost Optimization: EC2 Right Sizing

관련 비디오:

- Selecting the right Amazon EC2 instance for your workloads
- Right size your service

관련 예제:

- It just got easier to discover and compare Amazon EC2 instance types

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

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 해당 서비스를 구성할 수 있습니다. 또한 AWS에서는 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 관찰성 개선: 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 Storage Lens를 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 Storage Lens 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 적정 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 Storage Lens를 사용할 수 있습니다. Amazon S3 Storage Lens가 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 플릿를 자동으로 확장할 수 있습니다. 스팟 인스턴스 사용에 대한 할인을 받을 수 있을 뿐만 아니라 예약 인스턴스 또는 Savings Plans를 사용하여 일반 온디맨드 인스턴스 요금의 할인된 요금을 받을 수 있습니다. 이 모든 요소를 결합하여 Amazon EC2 인스턴스의 비용 절감을 최적화하고 애플리케이션에 대해 원하는 규모와 성능을 결정할 수 있습니다. 또한 Auto Scaling 그룹(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 Scaling은 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 자동으로 선택할 수 있는 Amazon S3 Intelligent Tiering 및 Amazon EFS Infrequent Access 기능을 사용할 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Right-Sizing
- AWS Compute Optimizer
- Amazon CloudWatch의 기능
- CloudWatch Getting Set Up
- CloudWatch Publishing Custom Metrics
- Getting Started with Amazon EC2 Auto Scaling
- Amazon S3 Storage Lens
- Amazon S3 Intelligent-Tiering
- Amazon EFS Infrequent Access
- Launch an Amazon EC2 Instance Using the SDK

관련 비디오:

- Right Size Your Services

관련 예제:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive scaling with Amazon EC2 Auto Scaling
- Amazon S3 Storage Lens로 비용 최적화 및 사용량에 대한 가시성 확보

COST06-BP04 공유 리소스 사용 고려

여러 사업부를 위해 조직 수준에서 이미 배포된 서비스의 경우 공유 리소스를 사용하여 활용도를 높이고 총 소유 비용(TCO)을 줄이는 것을 고려합니다. 공유 리소스는 기존 솔루션을 사용하거나 구성 요소를 공유하거나 두 가지 방법을 모두 사용하여 관리 및 비용을 중앙 집중화하는 비용 효과적인 옵션이 될 수 있습니다. 모니터링, 백업 및 연결과 같은 일반적인 기능을 계정 경계 내에서 또는 전용 계정에서 관리합니다. 또한 표준화를 구현하고, 중복을 줄이고, 복잡성을 줄임으로써 비용을 절감할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

여러 워크로드로 인해 동일한 기능이 발생하는 경우 기존 솔루션과 공유 구성 요소를 사용하여 관리를 개선하고 비용을 최적화합니다. 비프로덕션 데이터베이스 서버 또는 디렉터리 서비스와 같은 기존 리소스(특히 공유 리소스)를 사용하여 보안 모범 사례 및 조직 규정에 따라 클라우드 비용을 절감하는 것을 고려합니다. 최적의 가치 실현과 효율성을 위해서는 쇼백 및 차지백을 사용하여 소비를 발생시키는 비즈니스의 관련 영역에 비용을 다시 할당하는 것이 중요합니다.

쇼백은 클라우드 비용을 소비자, 사업부, 총계정원장 계정 또는 기타 책임 주체와 같이 비용의 원인이 될 수 있는 범주로 분류한 보고서입니다. 쇼백의 목표는 팀, 사업부 또는 개인에게 사용된 클라우드 리소스의 비용을 보여 주는 것입니다.

차지백은 특정 재무 관리 프로세스에 적합한 전략을 기반으로 중앙 서비스 지출을 비용 단위에 할당하는 것을 의미합니다. 고객의 경우 차지백은 하나의 공유 서비스 계정에서 발생한 비용을 고객 보고 프로세스에 적합한 여러 금융 비용 범주에 청구합니다. 차지백 메커니즘을 설정하여 여러 사업부, 제품 및 팀에서 발생한 비용을 보고할 수 있습니다.

워크로드는 중요한 워크로드와 중요하지 않은 워크로드로 분류할 수 있습니다. 이 분류에 따라 덜 중요한 워크로드에 대해서는 일반 구성이 적용된 공유 리소스를 사용합니다. 비용을 더욱 최적화하려면 중요한 워크로드에 사용할 전용 서버를 예약합니다. 리소스를 공유하거나 여러 계정에 프로비저닝하여 효율적으로 관리합니다. 개발, 테스트 및 프로덕션 환경이 서로 다르더라도 안전한 공유가 가능하며 조직 구조를 변경하지 않아도 됩니다.

컨테이너식 애플리케이션에 대한 이해도를 높이고 비용 및 사용을 최적화하려면 애플리케이션이 공유 컴퓨팅 및 메모리 리소스를 사용하는 방식을 기반으로 개별 비즈니스 엔터티에 비용을 할당하는 데 도움이 되는 분할 비용 할당 데이터를 사용합니다. 분할 비용 할당 데이터를 사용하면 Amazon Elastic Container Service(Amazon ECS) 또는 Amazon Elastic Kubernetes Service(Amazon EKS)에서 실행되는 컨테이너 워크로드에서 작업 수준의 쇼백 및 차지백을 달성할 수 있습니다.

분산 아키텍처의 경우 각 VPC의 워크로드에 필요한 공유 서비스에 대한 중앙 집중식 액세스를 제공하는 공유 서비스 VPC를 구축합니다. 이러한 공유 서비스에는 디렉터리 서비스 또는 VPC 엔드포인트와 같은 리소스가 포함될 수 있습니다. 관리 오버헤드와 비용을 줄이려면 각 VPC에 리소스를 구축하는 대신 중앙 위치에서 리소스를 공유합니다.

공유 리소스를 사용하면 운영 비용을 절감하고 리소스 활용도를 극대화하며 일관성을 개선할 수 있습니다. 다중 계정 설계에서는 일부 AWS 서비스를 중앙에서 호스팅하고 허브에 있는 여러 애플리케이션 및 계정을 사용하여 이러한 서비스에 액세스하면 비용을 절감할 수 있습니다. AWS Resource Access Manager(AWS RAM)를 사용하여 VPC 서브넷 및 AWS Transit Gateway Attachment, AWS Network Firewall 또는 Amazon SageMaker AI 파이프라인과 같은 기타 공통 리소스를 공유할 수 있습니다. 다중 계정 환경에서 AWS RAM을 사용하면 리소스를 한 번 생성하여 다른 계정과 공유할 수 있습니다.

조직은 공유 비용에 효과적으로 태그를 지정하고 비용 중 상당 부분에 태그가 지정되지 않았거나 할당되지 않은 상태는 아닌지 확인해야 합니다. 공유 비용을 효과적으로 할당하지 않고 공유 비용 관리를 책임지는 사람이 없는 경우 공유 클라우드 비용이 급증할 수 있습니다. 리소스, 워크로드, 팀 또는 조직 수준에서 비용이 발생한 부분을 알아야 합니다. 이러한 지식이 있다면 달성한 비즈니스 성과와 비교할 때 해당 수준에서 얼마나 큰 가치가 제공되었는지 더욱 정확하게 이해할 수 있기 때문입니다. 궁극적으로 조직은 클라우드 인프라 공유를 통해 비용을 절감할 수 있습니다. 공유 클라우드 리소스에 대한 비용 할당을 장려하여 클라우드 지출을 최적화합니다.

구현 단계

- 기존 리소스 평가: 워크로드에 유사한 서비스를 사용하는 기존 워크로드를 검토합니다. 워크로드의 구성 요소에 따라 비즈니스 논리 또는 기술 요구 사항이 허용하는 경우 기존 플랫폼을 고려합니다.
- AWS RAM에서 리소스 공유 사용 및 그에 따라 제한: 조직 내 다른 AWS 계정과 리소스를 공유하는 데 AWS RAM를 사용합니다. 리소스를 공유할 때 여러 계정에 리소스를 복제할 필요가 없으므로 리소스 유지 관리에 따른 운영 부담이 최소화됩니다. 이 프로세스는 또한 생성한 리소스를 계정의 역할 및 사용자는 물론 다른 AWS 계정과 안전하게 공유하는 데 도움이 됩니다.
- 리소스에 태그 지정: 비용 보고 후보인 리소스에 태그를 지정하고 비용 범주 내에서 분류합니다. 비용 할당을 위해 이러한 비용 관련 리소스 태그를 활성화하여 AWS 리소스 사용을 명확하게 확인할 수 있도록 합니다. 비용 및 사용 가시성을 적절한 수준으로 세분화하는 데 집중하고, 비용 할당 보고 및 KPI 추적을 통해 클라우드 소비 행동에 영향을 미칩니다.

리소스

관련 모범 사례:

- SEC03-BP08 안전하게 조직과 리소스 공유

관련 문서:

- What is AWS Resource Access Manager?
- AWS services that you can use with AWS Organizations
- Shareable AWS resources
- AWS Cost and Usage (CUR) Queries

관련 비디오:

- AWS Resource Access Manager - granular access control with managed permissions
- How to design your AWS cost allocation strategy
- AWS Cost Categories

관련 예제:

- How-to chargeback shared services: An AWS Transit Gateway example
- How to build a chargeback/showback model for Savings Plans using the CUR
- Using VPC Sharing for a Cost-Effective Multi-Account Microservice Architecture
- Improve cost visibility of Amazon EKS with AWS Split Cost Allocation Data
- AWS 분할 비용 할당 데이터를 사용하여 Amazon ECS 및 AWS Batch의 비용 가시성 향상

COST 7. 비용 절감을 위해 요금 모델을 어떻게 사용하나요?

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

모범 사례

- COST07-BP01 요금 모델 분석 수행
- COST07-BP02 비용을 기준으로 리전 선택
- COST07-BP03 비용 효율적인 조건을 갖춘 서드파티 계약 선택

- COST07-BP04 워크로드의 모든 구성 요소에 대한 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

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.

절감형 플랜은 1년 또는 3년 동안 일정한 사용량(시간당 USD로 측정)을 약정하는 대가로, Amazon EC2, Lambda 및 AWS Fargate에서의 사용량을 낮은 가격으로 제공하는 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 가격으로 용량을 준비할 수 있도록 지원합니다. 자세한 내용은 예약을 통한 비용 최적화를 참조하세요.

프로덕션, 품질 및 개발 환경과 관련된 리소스에 대해 절감형 플랜을 포함하도록 선택할 수 있습니다. 또는 샌드박스 리소스는 필요할 때만 켜지므로 해당 환경의 리소스에 대한 온디맨드 모델을 선택할 수도 있습니다. Amazon 스팟 인스턴스를 사용하여 Amazon EC2 비용을 절감하거나 컴퓨팅 절감형 플랜을 사용하여 Amazon EC2, Fargate 및 Lambda 비용을 절감할 수 있습니다. AWS Cost Explorer 권장 사항 도구는 절감형 플랜을 통해 약정 할인을 받을 수 있는 기회를 제공합니다.

이전에 Amazon EC2용 예약형 인스턴스를 구매했거나 조직 내에서 비용 할당 관행을 마련한 경우에는 한동안 Amazon EC2 예약형 인스턴스를 계속 사용할 수 있습니다. 그러나 향후 보다 유연한 비용 절감 메커니즘으로 절감형 플랜을 사용할 전략을 수립할 것을 권장합니다. AWS Cost Management에

서 절감형 플랜(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, Amazon OpenSearch Service 비용을 절감합니다. 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. 다양한 내결함성 및 유연한 애플리케이션에 스팟 인스턴스를 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.

단, 온디맨드 용량 예약(ODCR)은 요금 할인이 아닙니다. 예약 용량에서 인스턴스를 실행하는지 여부와 무관하게 동등한 온디맨드 요금이 용량 예약에 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 절감형 플랜 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 Lab: Cost Explorer
  - Well-Architected Lab: Cost Visualization
- 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 패밀리, AWS 리전, 가용 영역에 대해 특정 인스턴스에 대한 절감형 플램 또는 예약형 인스턴스, 예약 요금 할인을 고려하세요.
- 약정 할인 분석 수행: 계정에서 Cost Explorer를 사용하여 절감형 플랜 및 예약 인스턴스 권장 사항을 검토합니다. 필요한 할인 및 위험과 함께 올바른 권장 사항을 구현하려면 Well-Architected Labs을 따르세요.

리소스

관련 문서:

- Accessing Reserved Instance recommendations
- 인스턴스 구입 옵션
- AWS Enterprise

관련 비디오:

- Save up to 90% and run production workloads on Spot

관련 예제:

- Well-Architected Lab: Cost Explorer
- Well-Architected Lab: Cost Visualization
- Well-Architected Lab: Pricing Models

COST07-BP02 비용을 기준으로 리전 선택

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 않을 경우 노출되는 위험 수준: 중간

구현 지침

글로벌한 AWS 클라우드 인프라는 전 세계 여러 위치에서 호스트되며, AWS 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. AWS 계산기를 사용

하여 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 아시아 태평양(시드니)(`ap-southwest-2`)이고 다른 리전 사용에 대한 제한 사항(예: 개인정보 처리방침, 보안)이 없는 경우, 개발이나 테스트처럼 중요하지 않은 Amazon EC2 인스턴스는 미국 동부(버지니아 북부)(`us-east-1`) 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 매트릭스 표

위의 표는 리전 6이 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 요금 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 배포에 대한 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 사용합니다.
- 필요한 데이터 전송 분석: 리전을 선택할 때 데이터 전송 비용을 고려합니다. 데이터는 고객 및 리소스 가까운 곳에 둡니다. 데이터가 흐르고 데이터 전송이 최소화된 더 저렴한 AWS 리전을 선택합니다. 데이터 전송에 대한 비즈니스 요구 사항에 따라 Amazon CloudFront, AWS PrivateLink, AWS Direct Connect, AWS Virtual Private Network를 사용하여 네트워킹 비용을 줄이고 성능을 개선하며 보안을 강화할 수 있습니다.

리소스

관련 문서:

- Accessing Reserved Instance recommendations
- Amazon EC2 요금 정책
- 인스턴스 구입 옵션
- 리전 테이블

관련 비디오:

- Save up to 90% and run production workloads on Spot

관련 예제:

- Overview of Data Transfer Costs for Common Architectures
- Cost Considerations for Global Deployments
- What to Consider when Selecting a Region for your Workloads

COST07-BP03 비용 효율적인 조건을 갖춘 서드파티 계약 선택

비용 효율적인 계약과 조건은 이러한 서비스의 비용이 제공하는 혜택에 따라 규모가 조정됨을 보장합니다. 조직에 추가적인 혜택을 제공할 때 조정되는 계약 및 요금을 선택합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

시장에는 클라우드 환경에서 비용을 관리하는 데 도움이 되는 여러 제품이 출시되어 있습니다. 제품마다 고객 요구 사항에 따라 기능이 약간씩 다를 수 있습니다. 예를 들어 비용 거버넌스 또는 비용 가시성에 초점을 맞춘 제품도 있고 비용 최적화에 초점을 맞춘 제품도 있습니다. 효과적인 비용 최적화 및 거버넌스를 위해서는 필요한 기능과 적절한 요금 모델을 갖춘 올바른 도구를 사용하는 것이 매우 중요합니다. 이러한 제품의 요금 모델은 다양합니다. 월 청구액의 일정 비율을 청구하기도 하고 실현된 절감액의 일정 비율을 청구하기도 합니다. 사실 필요한 기능에 대해서만 비용을 지불하는 것이 가장 좋습니다.

클라우드에서 서드파티 솔루션 또는 서비스를 사용할 때는 원하는 성과에 맞는 요금 구조가 중요합니다. 요금은 제공하는 결과와 가치에 따라 조정되어야 합니다. 예를 들어 제공하는 절감액의 일정 비율에 비례하여 절감액(결과)이 커질수록 더 많은 요금을 부과하는 소프트웨어가 있습니다. 지출 증가에 따라 더 많은 비용을 지불하는 라이선스 계약이 비용 최적화에 항상 가장 적합한 것은 아닙니다. 그러나 공급업체가 청구서의 모든 부분에 대해 명확한 혜택을 제공한다면 조정되는 방식의 이러한 수수료가 정당화될 수 있습니다.

예를 들어 Amazon EC2에 대한 권장 사항을 제공하고 전체 청구 금액의 일정 비율을 부과하는 솔루션의 경우 혜택을 제공하지 않는 다른 서비스를 사용하면 솔루션 요금이 더 비싸집니다. 또 다른 예로는 관리형 리소스 비용의 일정 비율로 요금을 부과하는 관리형 서비스가 있습니다. 인스턴스 크기가 크다고 해서 반드시 관리에 더 많은 노력이 필요한 것은 아니지만 부과되는 요금은 증가할 수 있습니다. 따라서 이러한 서비스 요금 방식에 서비스의 효율성을 증진하는 비용 최적화 프로그램 또는 기능이 포함되어 있는지 확인해야 합니다.

고객은 시장에서 이러한 제품이 더 고급이거나 사용하기 더 쉽다고 생각할 수 있습니다. 이러한 제품의 비용을 고려하고 장기적인 관점에서 잠재적 비용 최적화 결과에 대해 생각해야 합니다.

구현 단계

- 서드파티 계약 및 조건 분석: 서드파티 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:

- Accessing Reserved Instance recommendations
- 인스턴스 구입 옵션

관련 비디오:

- Save up to 90% and run production workloads on Spot

COST07-BP04 워크로드의 모든 구성 요소에 대한 요금 모델 구현

영구 실행되는 리소스는 절감형 플랜 또는 예약형 인스턴스와 같은 예약 용량을 활용해야 합니다. 단기간 용량은 스팟 인스턴스나 스팟 플릿을 사용하도록 구성됩니다. 온디맨드 인스턴스는 예약 용량을 사용할 만큼 충분히 오래 실행되지 않으며(리소스 유형에 따라 25~75%에 해당하는 기간에 실행됨) 중단할 수 없는 단기 워크로드에만 사용됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

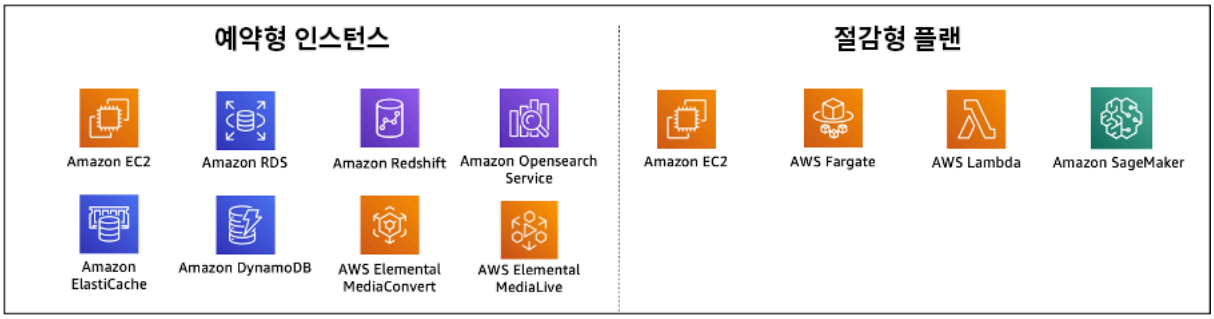
구현 가이드

AWS는 비용 효율성을 개선하기 위해 과거 사용량을 기반으로 여러 약정 권장 사항을 제공합니다. 이러한 권장 사항을 통해 절약할 수 있는 항목과 약정 사용 방법을 파악할 수 있습니다. 이러한 서비스를 온디맨드 또는 스팟으로 사용하거나 예약형 인스턴스(RI) 및 절감형 플랜(SP)을 통해 일정 기간에 대한 약정을 체결하고 온디맨드 비용을 줄일 수 있습니다. 워크로드를 최적화하려면 각 워크로드 구성 요소와 여러 AWS 서비스뿐만 아니라 이러한 서비스에 대한 약정 할인, 구매 옵션 및 스팟 인스턴스도 이해해야 합니다.

워크로드 구성 요소의 요구 사항을 고려하고 이러한 서비스에 대한 각기 다른 요금 모델을 이해합니다. 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.

예를 들어, AWS에서 이 웹 애플리케이션 아키텍처를 살펴보겠습니다. 이 샘플 워크로드는 Amazon Route 53, AWS WAF, Amazon CloudFront, Amazon EC2 인스턴스, Amazon RDS 인스턴스, 로드 밸런서, Amazon S3 스토리지, Amazon Elastic File System(Amazon EFS) 등 여러 개의 AWS 서비스로 구성됩니다. 이러한 각 서비스를 검토하고 다양한 요금 모델을 사용하여 잠재적인 비용 절감의 기회를 파악해야 합니다. 어떤 서비스는 RI 또는 SP에 적합할 수 있지만 어떤 서비스는 온디맨드로만 사용할

수 있습니다. 다음 이미지에서 볼 수 있듯이 일부 AWS 서비스는 RI 또는 SP를 사용하여 약정할 수 있습니다.



예약형 인스턴스 및 절감형 플랜을 사용하여 약정된 AWS 서비스

구현 단계

- 요금 모델 구현: 분석 결과를 사용하여 절감형 플랜, 예약형 인스턴스를 구매하거나 스팟 인스턴스를 구현합니다. 첫 약정 구매인 경우 목록에서 상위 5개 또는 10개의 권장 사항을 선택한 다음 한두 달 동안의 결과를 모니터링하고 분석하세요. AWS Cost Management Console에서 해당 프로세스를 ㅇ나내합니다. 콘솔에서 RI 또는 SP 권장 사항을 검토하고 권장 사항을 사용자 지정(유형, 결제 및 기간)하고 시간별 약정(예: 시간당 20 USD)을 검토한 다음, 장바구니에 추가합니다. 할인은 적격 사용량에 자동으로 적용됩니다. 예를 들어 정기적으로(2주마다 또는 매월) 약정 할인을 소량 구매합니다. 중단될 수 있거나 상태 정보를 저장하지 않는 워크로드에 대해 스팟 인스턴스를 구현합니다. 마지막으로 온디맨드 Amazon EC2 인스턴스를 선택하고 나머지 요구 사항에 맞게 리소스를 할당합니다.
- 워크로드 검토 주기: 요금 모델 적용 범위를 구체적으로 분석하는 워크로드에 대한 검토 주기를 구현합니다. 워크로드가 필요한 적용 범위에 도달하면 몇 달마다 또는 조직 사용량이 바뀔 때 부분적으로 추가 약정 할인을 구매합니다.

리소스

관련 문서:

- Understanding your Savings Plans recommendations
- Accessing Reserved Instance recommendations
- 예약형 인스턴스 구매 방법
- 인스턴스 구입 옵션
- 스팟 인스턴스
- 다른 AWS 서비스의 예약 모델
- Savings Plans Supported Services

관련 비디오:

- Save up to 90% and run production workloads on Spot

관련 예제:

- 절감형 플랜을 구매하기 전에 고려해야 할 사항은 무엇인가요?
- How can I use Cost Explorer to analyze my spending and usage?

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 확인하세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

정기적인 비용 모델링을 수행하면 여러 워크로드에 걸친 최적화 기회를 구현할 수 있습니다. 예를 들어, 전반적으로 여러 워크로드에 온디맨드 인스턴스를 사용하는 경우 변경 위험이 낮아지고 약정 기반 할인을 구현하여 전반적인 비용을 절감할 수 있습니다. 2주에서 1개월의 정기적인 주기로 분석을 수행하는 것이 좋습니다. 이렇게 하면 구매를 조금씩 조정할 수 있으므로 워크로드 및 워크로드 구성 요소의 변경에 따라 요금 모델의 적용 범위를 점진적으로 변경할 수 있습니다.

AWS Cost Explorer 권장 사항 도구를 사용하여 관리 계정에서 약정 할인의 기회를 찾을 수 있습니다. 관리 계정 수준의 권장 사항은 예약형 인스턴스(RI) 또는 절감형 플랜(SP)이 있는 AWS 조직 내 모든 계정의 사용량을 고려하여 계산됩니다. 또한 할인 공유가 활성화될 때 계산되어 계정 전체에서 비용 절감을 극대화하는 약정을 추천할 수 있습니다.

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 계산되어 각 계정의 절감 효과를 극대화합니다. 계정에 RI 약정과 SP 약정이 둘 다 있는 경우 다음 순서로 적용됩니다.

1. 영역 RI
2. 스탠다드 RI
3. 컨버터블 RI
4. 인스턴스 절감형 플랜
5. 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 패밀리에서 절감형 플랜은 먼저 소유자 계정의 사용량에 적용된 다음, 다른 계정의 사용량에 적용됩니다. 이는 공유를 활성화한 경우에만 발생합니다. 절감형 플랜은 가장 높은 절감률에 먼저 적용됩니다. 절감률이 같은 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 요금에서 부과됩니다. AWS 비용 관리에서 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 옵션으로 워크로드의 단기 비용을 분석하고, AWS 요금 모델을 분석하며, 이를 비즈니스 요구 사항에 맞춰 조정하여 총 소유 비용을 확인하고 비용 최적화 기회를 모색합니다.

구현 단계

약정 할인 분석 수행: 계정에서 Cost Explorer를 사용하여 절감형 플랜 및 예약형 인스턴스 권장 사항을 검토합니다. 절감형 플랜 권장 사항을 이해하고 월간 지출액과 월간 절감액을 추정해야 합니다. 계정 간 절감액을 극대화하기 위해 RI 또는 절감형 플랜 할인 공유를 활성화한 AWS 조직 내 모든 구성원 계정의 사용량을 고려하여 측정되는 관리 계정 수준의 권장 사항을 검토합니다. 필요한 할인 및 위험과 함께 올바른 권장 사항을 구현하려면 Well-Architected 실습을 따르면 됩니다.

리소스

관련 문서:

- AWS 요금은 어떻게 적용되나요
- 인스턴스 구입 옵션
- Saving Plan Overview
- Saving Plan recommendations
- Accessing Reserved Instance recommendations
- Understanding your Saving Plans recommendation
- How Savings Plans apply to your AWS usage

- 통합 결제에서 절감형 플랜
- Turning on shared reserved instances and Savings Plans discounts

관련 비디오:

- Save up to 90% and run production workloads on Spot

관련 예제:

- 절감형 플랜을 구매하기 전에 고려해야 할 사항은 무엇인가요?
- How can I use rolling Savings Plans to reduce commitment risk?
- 스팟 인스턴스를 사용하는 시기

COST 8. 데이터 전송 요금을 위한 계획은 어떻게 되나요?

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 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

클라우드에서 솔루션을 설계할 때 온프레미스 데이터 센터를 사용하여 아키텍처를 설계하는 습관이나 지식 부족으로 인해 데이터 전송 비용을 무시하는 경우가 많습니다. AWS의 데이터 전송 요금은 소스, 대상, 트래픽 볼륨에 따라 결정됩니다. 설계 단계에서 이러한 수수료를 고려하면 비용을 절감할 수 있습니다. 총 소유 비용(TCO)을 정확하게 추정하려면 워크로드에서 데이터 전송이 발생하는 위치, 전송

비용, 관련 이점을 이해하는 것이 매우 중요합니다. 그러면 정보를 바탕으로 결정을 내리고 아키텍처 의사 결정을 수정하거나 수락할 수 있습니다. 예를 들어 가용 영역 간에 데이터를 복제하는 다중 가용 영역 구성이 있는 경우가 있습니다.

워크로드에서 데이터를 전송하는 서비스 구성 요소를 모델링하고, 필요한 신뢰성과 복원력을 달성하기 위해 수용 가능한 비용인지(두 가용 영역의 컴퓨팅 및 스토리지 비용을 지불하는 것과 유사함) 판단합니다. 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.

데이터 전송을 모델링하는 동안 수집되는 데이터의 양과 데이터의 출처를 생각해 보세요. 또한 처리되는 데이터의 양과 필요한 스토리지 또는 컴퓨팅 용량을 고려하세요. 모델링하는 동안 워크로드 아키텍처에 대한 네트워킹 모범 사례를 따라 잠재적 데이터 전송 비용을 최적화합니다.

AWS Pricing Calculator를 통해 특정 AWS 서비스의 예상 비용과 예상 데이터 전송을 확인할 수 있습니다. 테스트 목적 또는 사전 프로덕션 환경에서 워크로드가 이미 실행 중인 경우 AWS Cost Explorer 또는 AWS Cost and Usage Report(CUR)를 사용하여 데이터 전송 비용을 파악하고 모델링합니다. 개념 증명(PoC)을 구성하거나 워크로드를 테스트하고 사실적으로 시뮬레이션된 로드로 테스트를 실행합니다. 다양한 워크로드 수요에서 비용을 모델링할 수 있습니다.

구현 단계

- 요구 사항 식별: 소스와 대상 간에 계획된 데이터 전송의 주요 목표와 비즈니스 요구 사항이 무엇인가요? 최종적으로 기대되는 비즈니스 결과는 무엇인가요? 비즈니스 요구 사항을 수집하고 예상 결과를 정의합니다.
- 소스 및 대상 식별: 데이터 전송의 데이터 소스와 대상이 무엇인가요? 예를 들면 AWS 리전 내부 전송, AWS 서비스로 전송, 인터넷으로 전송 등이 이에 해당될 수 있습니다.
  - AWS 리전 내부 데이터 전송
  - AWS 리전 간 데이터 전송
  - 인터넷으로 데이터 전송
- 데이터 분류 식별: 이 데이터 전송에 대한 데이터 분류가 어떻게 되나요? 어떤 종류의 데이터인가요? 데이터의 크기는 얼마나 되나요? 데이터를 얼마나 자주 전송해야 하나요? 데이터가 민감한가요?
- 사용할 AWS 서비스 또는 도구 식별: 이 데이터 전송에는 어떤 AWS 서비스가 사용되나요? 이미 프로비저닝된 서비스를 다른 워크로드에 사용할 수 있나요?
- 데이터 전송 비용 계산: 이전에 생성한 데이터 전송 모델링의 AWS 요금을 사용하여 워크로드에 대한 데이터 전송 비용을 계산합니다. 워크로드 사용량의 증가와 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 가져다주는 성과를 지정합니다. 구성 요소 간 전송인 경우 분리를 위한 것일 수 있으며, 가용 영역 간 전송인 경우 이중화를 위한 것일 수 있습니다.
- 데이터 전송 모델링 생성: 모든 정보를 수집한 후 여러 사용 사례와 다양한 워크로드에 대한 개념적 기본 데이터 전송 모델링을 생성합니다.

리소스

관련 문서:

- AWS caching solutions
- AWS 요금
- Amazon EC2 요금
- Amazon VPC 요금
- Understanding data transfer charges

관련 비디오:

- Monitoring and Optimizing Your Data Transfer Costs
- S3 Transfer Acceleration

관련 예제:

- Overview of Data Transfer Costs for Common Architectures
- AWS Prescriptive Guidance for Networking

COST08-BP02 데이터 전송 비용을 최적화할 구성 요소 선택

모든 구성 요소를 선택해야 하며, 데이터 전송 비용을 줄이도록 아키텍처를 설계해야 합니다. 이때 광역 네트워크(WAN) 최적화, 다중 가용 영역 구성 등의 구성 요소를 사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

데이터 전송을 위한 아키텍팅은 데이터 전송 비용을 최소화합니다. 이 과정에서는 콘텐츠 전송 네트워크를 사용해 사용자와 더 가까운 위치에 데이터를 배치하거나, 온프레미스에서 AWS로의 전용 네트워

크 링크를 사용합니다. 또한 WAN 최적화 및 애플리케이션 최적화를 사용하여 구성 요소 간에 전송되는 데이터의 양을 줄일 수도 있습니다.

데이터를 AWS 클라우드에 전송하거나 해당 클라우드 내부에서 전송할 때 데이터 전송을 최적화하는 데 적합한 AWS 서비스를 선택하려면 다양한 사용 사례, 데이터의 특성 및 사용 가능한 네트워크 리소스를 바탕으로 대상을 파악하는 것이 중요합니다. AWS는 여러 데이터 마이그레이션 요구 사항에 맞춤화된 다양한 데이터 전송 서비스를 제공합니다. 조직 내 비즈니스 요구 사항에 따라 적절한 데이터 스토리지 및 데이터 전송 옵션을 선택합니다.

워크로드 아키텍처를 계획하거나 검토할 때는 다음 사항을 고려하세요.

- AWS 내에서 VPC 엔드포인트 사용: VPC 엔드포인트는 VPC와 지원되는 AWS 서비스 간의 프라이빗 연결을 허용합니다. 이렇게 하면 데이터 전송 비용이 발생할 수 있는 퍼블릭 인터넷을 사용하지 않아도 됩니다.
- NAT 게이트웨이 사용: 프라이빗 서브넷의 인스턴스가 인터넷이나 VPC 외부의 서비스에 연결할 수 있도록 NAT 게이트웨이를 사용합니다. 가장 많은 트래픽을 전송하는 NAT 게이트웨이 뒤의 리소스가 NAT 게이트웨이와 동일한 가용 영역에 있는지 확인합니다. 그렇지 않은 경우 리소스와 동일한 가용 영역에 새 NAT 게이트웨이를 생성하여 교차 AZ 데이터 전송 요금을 줄이세요.
- AWS Direct Connect 사용: AWS Direct Connect에서는 퍼블릭 인터넷을 우회하고 온프레미스 네트워크와 AWS 간에 직접적인 프라이빗 연결을 설정합니다. 인터넷을 통해 대량의 데이터를 전송하는 것보다 더 비용 효율적이고 일관적인 방법일 수 있습니다.
- 리전 경계를 넘어선 데이터 전송 방지: AWS 리전 리전 간에(한 리전에서 다른 리전으로) 데이터를 전송하면 일반적으로 요금이 부과됩니다. 다중 리전 경로를 이용하려면 매우 신중하게 결정해야 합니다. 자세한 내용은 다중 리전 시나리오를 참조하세요.
- 데이터 전송 모니터링: Amazon CloudWatch 및 VPC 흐름 로그를 사용하여 데이터 전송 및 네트워크 사용에 대한 세부 정보를 캡처합니다. VPC에서 캡처한 네트워크 트래픽 정보(예: 네트워크 인터페이스에서 주고받는 IP 주소 또는 범위)를 분석합니다.
- 네트워크 사용량 분석: AWS Cost Explorer, CUDOS 대시보드, CloudWatch와 같은 측정 및 보고 도구를 사용하여 워크로드의 데이터 전송 비용을 파악합니다.

구현 단계

- 데이터 전송을 위한 구성 요소 선택: COST08-BP01 데이터 전송 모델링 수행에서 설명한 데이터 전송 모델링을 사용하여 데이터 전송 비용이 가장 큰 영역이나 워크로드 사용량이 변경되는 경우 데이터 전송 비용이 발생하는 영역을 집중적으로 살펴봅니다. 데이터 전송에 대한 필요를 줄이거나, 없애거나, 비용을 낮추는 대체 아키텍처나 추가 구성 요소를 찾습니다.

리소스

관련 모범 사례:

- COST08-BP01 데이터 전송 모델링 수행
- COST08-BP03 데이터 전송 비용을 줄이기 위한 서비스 구현

관련 문서:

- Cloud Data Migration
- AWS caching solutions
- Deliver content faster with Amazon CloudFront

관련 예제:

- Overview of Data Transfer Costs for Common Architectures
- AWS Network Optimization Tips
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format

COST08-BP03 데이터 전송 비용을 줄이기 위한 서비스 구현

데이터 전송을 줄이기 위한 서비스를 구현합니다. 예를 들어 엣지 로케이션 또는 콘텐츠 전송 네트워크(CDN)를 사용하여 최종 사용자에게 콘텐츠를 전송하고, 애플리케이션 서버 또는 데이터베이스 앞에 캐싱 계층을 구축하며, 클라우드 연결에 VPN 대신 전용 네트워크 연결을 사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

네트워크 데이터 전송 사용을 최적화하는 데 도움이 되는 다양한 AWS 서비스가 있습니다. 이러한 서비스는 워크로드 구성 요소, 유형 및 클라우드 아키텍처에 따라 클라우드에서 트래픽을 압축, 캐싱, 공유 및 분배하는 데 도움이 될 수 있습니다.

- Amazon CloudFront는 지연 시간이 짧고 전송 속도가 빠른 데이터 전송용 글로벌 콘텐츠 전송 네트워크입니다. 전 세계의 엣지 로케이션에서 데이터를 캐시하여 리소스 부하를 줄여줍니다. CloudFront를 사용하면 전 세계의 많은 사용자에게 콘텐츠를 전송하는 과정의 관리 작업을 줄이고

지연 시간을 최소화할 수 있습니다. Security Savings Bundle을 사용하면 시간이 지남에 따라 사용량을 늘리려는 경우 CloudFront 사용량을 최대 30%까지 절약할 수 있습니다.

- AWS Direct Connect를 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 VPN을 사용하면 프라이빗 네트워크와 AWS 글로벌 네트워크 간에 안전한 프라이빗 연결을 설정할 수 있습니다. 탄력적인 완전관리형 서비스로서 간소화된 연결을 제공하므로 소규모 사무실 또는 비즈니스 파트너에 적합합니다.
- VPC 엔드포인트를 사용하면 프라이빗 네트워킹을 통해 여러 AWS 서비스를 연결할 수 있으며, 퍼블릭 데이터 전송 및 NAT 게이트웨이 비용을 줄일 수 있습니다. 게이트웨이 VPC 엔드포인트는 시간당 비용이 없으며 Amazon S3 및 Amazon DynamoDB를 지원합니다. 인터페이스 VPC 엔드포인트는 AWS PrivateLink를 통해 제공되며 시간당 요금과 GB당 사용 비용이 있습니다.
- NAT 게이트웨이는 독립형 NAT 인스턴스와 달리 규모 조정 및 관리 기능을 기본 제공하여 비용을 절감합니다. 트래픽이 많은 인스턴스와 동일한 가용 영역에 NAT 게이트웨이를 배치하고, Amazon DynamoDB 또는 Amazon S3 액세스가 필요한 인스턴스에는 VPC 엔드포인트를 사용하여 데이터 전송 및 처리 비용을 줄이는 것을 고려하세요.
- 엣지에서 데이터를 수집하고 처리하기 위한 컴퓨팅 리소스가 있는 AWS Snow Family 디바이스를 사용합니다. AWS Snow Family 디바이스(Snowball Edge, Snowball Edge 및 Snowmobile)를 사용하면 페타바이트 규모의 데이터를 AWS 클라우드로 비용 효과적으로, 오프라인으로 이동할 수 있습니다.

구현 단계

- 서비스 구현: 데이터 전송 모델링을 사용하고 VPC 흐름 로그를 검토하여 서비스, 워크로드 유형에 따라 적절한 AWS 네트워크 서비스를 선택합니다. 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 caching solutions

- Amazon CloudFront
- AWS Snow Family
- Amazon CloudFront Security Savings Bundle

관련 비디오:

- Monitoring and Optimizing Your Data Transfer Costs
- AWS Cost Optimization Series: CloudFront
- VPC의 NAT 게이트웨이에 대한 데이터 전송 요금을 줄이려면 어떻게 해야 하나요?

관련 예제:

- How-to chargeback shared services: An AWS Transit Gateway example
- Understand AWS data transfer details in depth from cost and usage report using Athena query and QuickSight
- Overview of Data Transfer Costs for Common Architectures
- Using AWS Cost Explorer to analyze data transfer costs
- Cost-Optimizing your AWS architectures by utilizing Amazon CloudFront features
- VPC의 NAT 게이트웨이에 대한 데이터 전송 요금을 줄이려면 어떻게 해야 하나요?

수요 관리 및 리소스 공급

질문

- COST 9. 수요와 리소스 공급은 어떻게 관리하나요?

COST 9. 수요와 리소스 공급은 어떻게 관리하나요?

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

클라우드 컴퓨팅에 대한 워크로드 수요를 분석하려면 클라우드 환경에서 시작되는 컴퓨팅 작업의 패턴과 특성을 이해해야 합니다. 이 분석을 통해 사용자는 리소스 할당을 최적화하고 비용을 관리하며 성능이 요구 수준을 충족하는지 확인할 수 있습니다.

워크로드의 요구 사항을 파악합니다. 조직의 요구 사항에는 요청에 대한 워크로드 응답 시간이 표시되어야 합니다. 응답 시간을 사용하면 수요가 관리되는지 여부 또는 리소스 공급이 수요에 따라 변경되어야 하는지 여부를 확인할 수 있습니다.

분석에는 수요의 예측 가능성 및 반복 가능성, 수요 변경의 속도 및 수요 변경의 규모가 포함되어야 합니다. 월말 처리 또는 휴가철 피크와 같은 계절적 변동을 포함하기에 충분히 긴 기간에 걸쳐 분석을 수행합니다.

분석 작업에는 규모 조정 구현의 잠재적 이점이 반영되어야 합니다. 구성 요소의 예상 총 비용을 찾아보고 워크로드 수명에 걸쳐 사용량 및 비용이 증가하거나 감소하는지 살펴봅니다.

다음은 클라우드 컴퓨팅에 대한 워크로드 수요 분석을 수행할 때 고려해야 할 몇 가지 주요 측면입니다.

1. 리소스 사용률 및 성과 지표: 시간이 지남에 따라 AWS 리소스가 어떻게 사용되고 있는지 분석합니다. 사용량이 최대일 때와 사용량이 적을 때의 패턴을 확인하여 리소스 할당 및 규모 조정 전략을 최적화합니다. 응답 시간, 지연 시간, 처리량, 오류율과 같은 성과 지표를 모니터링합니다. 이러한 지표는 클라우드 인프라의 전반적인 상태와 효율성을 평가하는 데 도움이 됩니다.
2. 사용자 및 애플리케이션 규모 조정 동작: 사용자 행동과 이것이 워크로드 수요에 미치는 영향을 파악합니다. 사용자 트래픽 패턴을 조사하면 콘텐츠 전송 및 애플리케이션 응답성을 향상시키는 데 도움이 됩니다. 증가하는 수요에 맞추어 워크로드 규모가 어떻게 조정되는지 분석합니다. 로드 변동을 처리할 수 있도록 Auto Scaling 파라미터가 정확하고 효과적으로 구성되었는지 확인합니다.

3. 워크로드 유형: 일괄 처리, 실시간 데이터 처리, 웹 애플리케이션, 데이터베이스 또는 기계 학습과 같이 클라우드에서 실행되는 다양한 유형의 워크로드를 파악합니다. 워크로드 유형마다 리소스 요구 사항 및 성능 프로필이 다를 수 있습니다.
4. 서비스 수준에 관한 계약(SLA): 실제 성능을 SLA와 비교하여 규정 준수를 보장하고 개선이 필요한 영역을 파악합니다.

Amazon CloudWatch를 사용하여 지표를 수집 및 추적하고, 로그 파일을 수집 및 모니터링하며, 경보를 설정하고, AWS 리소스 변경에 자동으로 대응할 수 있습니다. 또한 Amazon CloudWatch를 사용하여 시스템 전반의 리소스 사용률, 애플리케이션 성능, 운영 상태를 파악할 수 있습니다.

AWS Trusted Advisor를 사용하면 모범 사례에 따라 리소스를 프로비저닝하여 시스템 성능 및 신뢰성을 개선하고, 보안을 강화하며, 비용을 절감할 기회를 모색할 수 있습니다. 비프로덕션 인스턴스를 끄고 수요 증가 또는 감소에 맞춰 Amazon CloudWatch 및 Auto Scaling을 사용할 수도 있습니다.

마지막으로, AWS Cost Explorer 또는 QuickSight를 AWS Cost and Usage Report(CUR) 파일 또는 애플리케이션 로그와 함께 사용하여 워크로드 수요에 대한 고급 분석을 수행할 수 있습니다.

전반적으로, 포괄적인 워크로드 수요 분석을 통해 조직은 리소스 프로비저닝, 규모 조정 및 최적화에 대해 합리적인 결정을 내려서 성능, 비용 효율성 및 사용자 만족도를 높일 수 있습니다.

구현 단계

- 기존 워크로드 데이터 분석: 기존 워크로드, 이전 버전의 워크로드 또는 예측된 사용 패턴의 데이터를 분석합니다. Amazon CloudWatch, 로그 파일 및 모니터링 데이터를 사용하여 워크로드가 어떻게 사용되었는지에 대한 인사이트를 얻을 수 있습니다. 워크로드의 전체 주기를 분석하여 월말 또는 연말 이벤트와 같은 주기적 변경 사항에 대한 데이터를 수집합니다. 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.
- 외부 영향 예측: 워크로드의 수요에 영향을 주거나 변화를 줄 수 있는 조직 전체의 팀원과 만납니다. 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:

- Amazon CloudWatch
- AWS Trusted Advisor

- AWS X-Ray
- AWS Auto Scaling
- AWS Instance Scheduler
- Getting started with Amazon SQS
- AWS Cost Explorer
- QuickSight

관련 예제:

- Monitor, Track and Analyze for cost optimization
- Searching and analyzing logs in CloudWatch

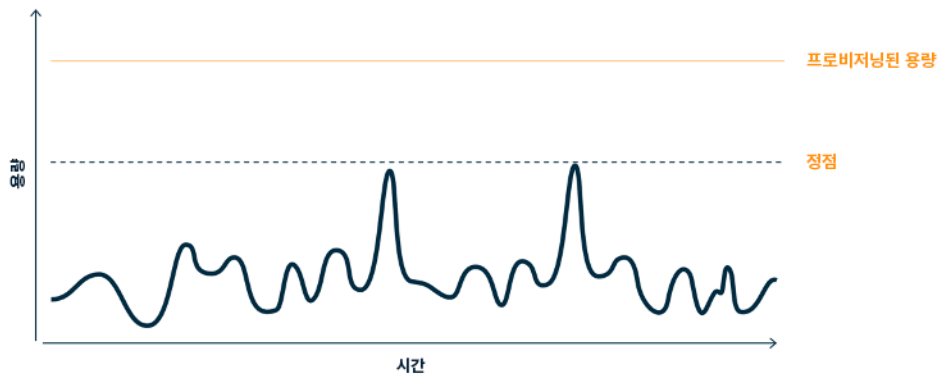
COST09-BP02 수요 관리를 위한 버퍼 또는 제한 구현

버퍼링 및 제한은 워크로드의 수요를 수정하여 평준화합니다. 클라이언트가 재시도를 수행할 때 제한을 구현합니다. 요청을 저장하고 나중에 처리하도록 버퍼링을 구현합니다. 클라이언트가 필요한 시간에 응답을 수신하도록 제한 및 버퍼가 설계되었는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

클라우드 컴퓨팅에서 수요를 관리하고 워크로드에 필요한 프로비저닝 용량을 줄이려면 버퍼 또는 제한을 구현하는 것이 중요합니다. 최적의 성능을 위해서는 피크, 요청 변경 속도, 필요한 응답 시간을 포함한 총 수요를 측정하는 것이 핵심입니다. 클라이언트가 요청을 재전송할 수 있게 되면 제한을 적용하는 것이 실용적입니다. 반대로 재시도 기능이 없는 클라이언트의 경우 이상적인 접근 방식은 버퍼 솔루션을 구현하는 것입니다. 이러한 버퍼는 요청 유입을 간소화하고 다양한 작업 속도로 애플리케이션 간의 상호 작용을 최적화합니다.



높은 프로비저닝 용량을 필요로 하는 두 개의 피크가 있는 수요 곡선

위 그림에 표시된 수요 곡선을 바탕으로 워크로드를 가정합니다. 이 워크로드에는 2개의 피크가 있으며, 이러한 피크를 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 2개의 피크를 처리하려면 프로비저닝된 용량이 필요하기 때문입니다. 워크로드 수요 곡선을 완화하면 워크로드에 프로비저닝된 용량을 줄이고 환경에 미치는 영향도 줄일 수 있습니다. 피크 속도를 낮추려면 제한 또는 버퍼링 솔루션을 구현하는 것이 좋습니다.

이해를 돕기 위해 제한과 버퍼링에 대해 살펴보겠습니다.

제한: 수요의 소스에 재시도 기능이 있는 경우 제한을 구현할 수 있습니다. 제한은 현재 요청을 처리할 수 없는 경우 나중에 다시 시도해야 함을 소스에 알려줍니다. 소스는 일정 기간 기다린 다음 요청을 재시도합니다. 제한을 구현하면 워크로드의 최대 리소스 양과 비용을 제한할 수 있는 장점이 있습니다. AWS에서는 Amazon API Gateway를 사용하여 제한을 구현할 수 있습니다.

버퍼 기반: 버퍼 기반 접근 방식은 생산자(대기열에 메시지를 보내는 구성 요소), 소비자(대기열에서 메시지를 받는 구성 요소) 및 대기열(메시지를 보관함)을 사용하여 메시지를 저장합니다. 메시지는 소비자가 읽은 후 처리되므로 소비자의 비즈니스 요구 사항을 충족하는 속도로 메시지를 실행할 수 있습니다. 버퍼 중심 방법을 사용하면 생산자의 메시지가 대기열이나 스트림에 보관되어 소비자가 작업 요구 사항에 맞는 속도로 액세스할 수 있습니다.

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 Kinesis에서는 여러 소비자가 같은 메시지를 읽을 수 있는 스트림을 제공합니다.

버퍼링 및 제한을 통해 워크로드에 대한 수요를 수정하여 피크를 원활하게 처리할 수 있습니다. 클라이언트가 작업을 재시도할 때는 제한을 사용하고 요청을 보류했다가 나중에 처리하려면 버퍼링을 사용

합니다. 버퍼 기반 접근 방식을 사용할 때는 필요한 시간에 요청을 처리하도록 워크로드를 설계해야 하며 작업에 대한 중복 요청을 처리할 수 있는지 확인해야 합니다. 전체 수요, 변경률 및 필수 응답 시간을 분석하여 필요한 제한 또는 버퍼의 크기를 적절하게 조정합니다.

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 제한 또는 버퍼의 크기를 결정합니다.
- 버퍼 또는 제한 구현: 워크로드에 버퍼 또는 제한을 구현합니다. Amazon Simple Queue Service(Amazon SQS)와 같은 대기열은 워크로드 구성 요소에 버퍼를 제공할 수 있습니다. Amazon API Gateway는 워크로드 구성 요소에 대한 제한을 제공할 수 있습니다.

리소스

관련 모범 사례:

- SUS02-BP06 버퍼링 또는 제한 개선으로 수요 곡선 완화
- REL05-BP02 요청 제한

관련 문서:

- AWS Auto Scaling
- AWS Instance Scheduler
- Amazon API Gateway
- Amazon Simple Queue Service
- Getting started with Amazon SQS
- Amazon Kinesis

관련 비디오:

- Choosing the Right Messaging Service for Your Distributed App

관련 예제:

- Managing and monitoring API throttling in your workloads

- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Enabling Tiering and Throttling in a Multi-Tenant Amazon EKS SaaS Solution Using Amazon API Gateway
- Application integration Using Queues and Messages

COST09-BP03 동적으로 리소스 공급

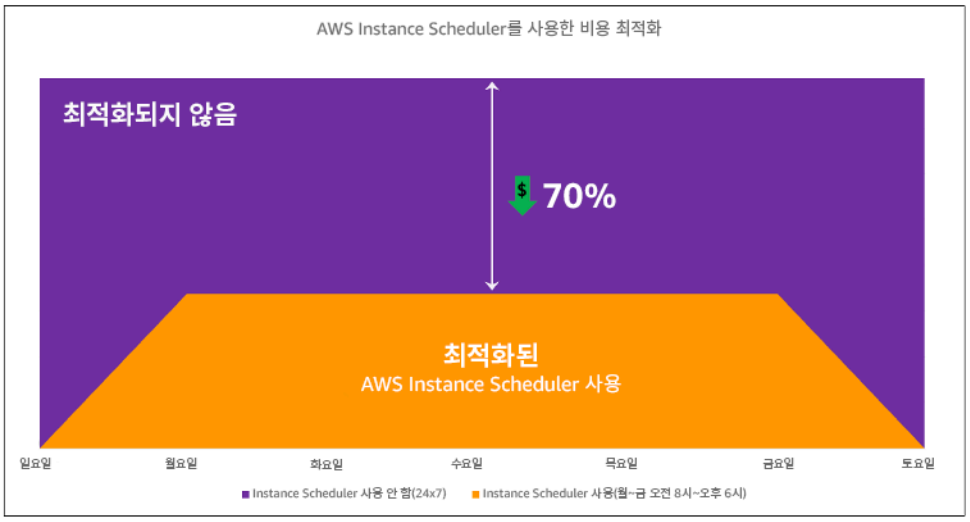
리소스가 계획된 방식으로 프로비저닝됩니다. 자동 크기 조정과 같은 수요 기반이거나, 수요를 예측할 수 있고 리소스가 시간을 기준으로 제공되는 시간 기반일 수 있습니다. 이러한 방법을 사용하면 과다 프로비저닝 또는 과소 프로비저닝을 최소화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 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(RDS) 인스턴스의 시작 및 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. Amazon EC2 인스턴스 및 스팟 플릿, Amazon ECS, Amazon DynamoDB 및 Amazon Aurora와 통합되어 애플리케이션 용량을 조정하는 완전관리형의 무료 서비스입니다. Auto Scaling이 제공하는 자동 리소스 검색 기능을 사용하면 워크로드에서 구성 가능한 리소스를 쉽게 찾을 수 있습니다. 또한 기본적으로 포함되어 있는 확장 전략을 통해 성능, 비용 또는 둘 사이의 균형을 최적화할 수 있으며 예측 조정 기능을 통해 주기적으로 발생하는 스파이크를 지원할 수 있습니다.

Auto Scaling 그룹을 조정하는 데 사용할 수 있는 다양한 규모 조정 옵션이 있습니다.

- 항상 현재 인스턴스 수준 유지 관리
- 수동 조정
- 일정에 근거하여 조정
- 온디맨드 기반 조정
- 예측 조정 사용

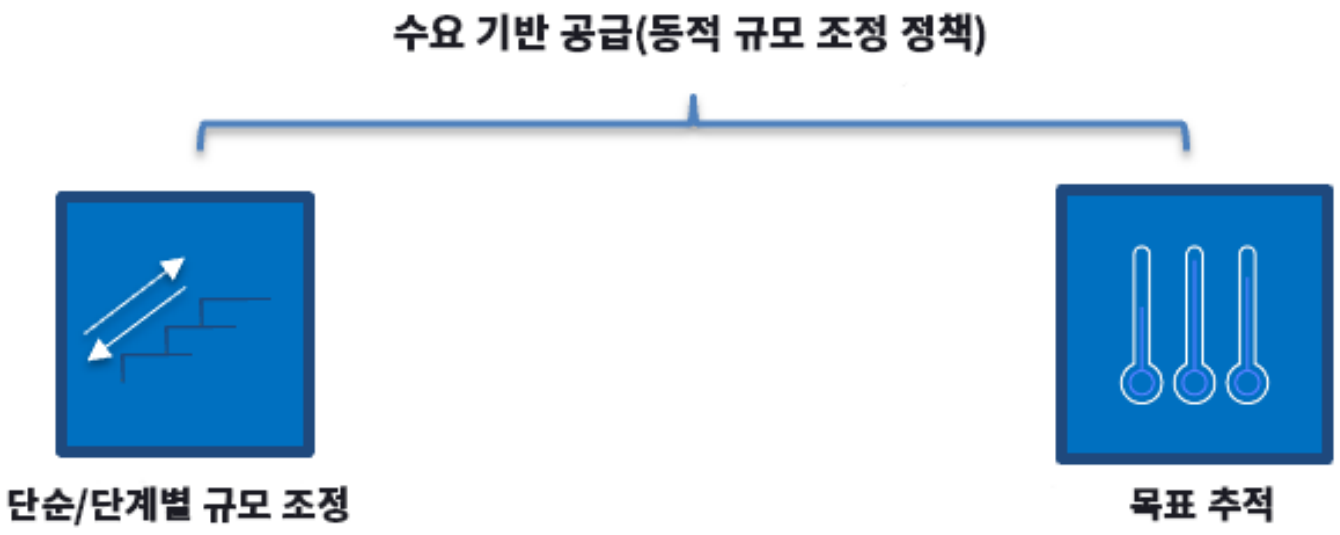
Auto Scaling 정책은 서로 다르며 동적 및 예약 규모 조정 정책으로 분류될 수 있습니다. 동적 정책은 수동 또는 동적 규모 조정으로, 예약 또는 예측 규모 조정입니다. 동적, 예약 및 예측 규모 조정에 규모 조정 정책을 사용할 수 있습니다. 또한 Amazon CloudWatch의 지표와 경보를 사용하여 워크로드에 대한 규모 조정 이벤트를 트리거할 수 있습니다. 최신 기능과 개선 사항에 액세스하려면 시작 템플릿을 사용하는 것이 좋습니다. 시작 구성을 사용할 때 일부 Auto Scaling 기능을 사용할 수 없습니다. 예를 들어, 스팟 및 온디맨드 인스턴스를 모두 시작하거나 여러 인스턴스 유형을 지정하는 Auto Scaling 그룹은 생성할 수 없습니다. 이러한 기능을 구성하려면 시작 템플릿을 사용해야 합니다. 시작 템플릿을 사용할 때는 각 템플릿의 버전을 지정하는 것이 좋습니다. 시작 템플릿 버전 관리를 사용하면 전체 파라미터 세트의 하위 세트를 생성할 수 있습니다. 그런 다음, 해당 세트를 재사용하여 동일한 시작 템플릿의 다른 버전을 만들 수 있습니다.

AWS Auto Scaling을 사용하거나 AWS API 또는 SDK를 사용하여 코드에서 규모 조정을 통합할 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 수평적 스케일링 및 수직적 스케일링과 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 Balancing은 여러 리소스에 걸쳐 수요를 분산하여 규모를 조정하는 데 도움이 됩니다. ASG 및 Elastic Load Balancing을 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 기간에는 리소스 수를 늘려 성능을 유지하고 수요 감소 기간에는 용량을 줄여 비용을 절감할 수 있습니다.

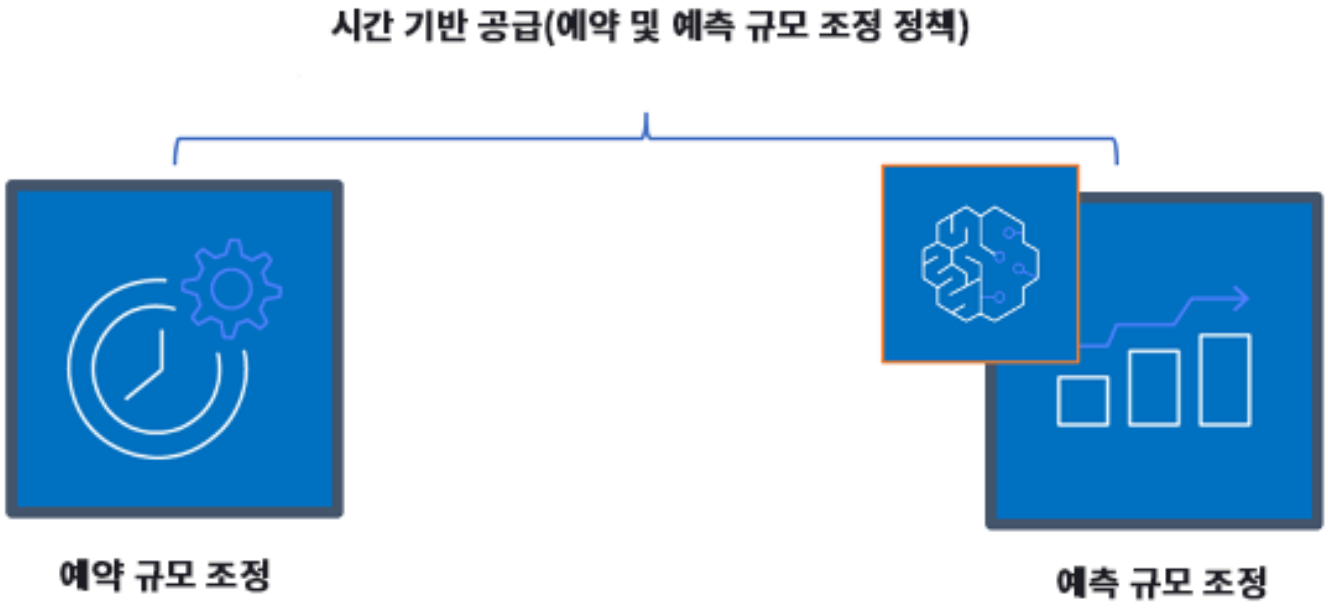


수요 기반 동적 규모 조정 정책

- 단순/단계별 규모 조정: 고객이 수동으로 정의한 단계에 따라 지표를 모니터링하고 인스턴스를 추가 또는 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 기간에 추가 리소스를 제공하거나 용량을 늘릴 수 있습니다.



시간 기반 규모 조정 정책

예약 또는 예측 Auto Scaling을 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 또는 스케일 인을 예약할 수 있습니다. 예측 규모 조정은 패턴을 사용하여 스케일 아웃하는 반면에 예약된 규모 조정은 미리 정의된 시간을 사용하여 스케일 아웃합니다. 또한 Auto Scaling 그룹의 속성 기반 인스턴스 유형 선택(ABS) 전략을 사용하여 vCPU, 메모리 및 스토리지와 같은 일련의 속성으로서 인스턴스 요구 사항을 표현할 수 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 Scaling은 지정된 속성에 맞는 인스턴스를 선택하고 시작하여 인스턴스 유형을 수동으로 선택할 필요성이 사라집니다.

또한 AWS API 및 SDK와 AWS CloudFormation을 활용하여 필요한 경우 전체 환경을 자동으로 프로비저닝하고 폐기할 수 있습니다. 이 방식은 정의된 업무 시간이나 일정 기간에만 실행되는 개발 또는 테스트 환경에 적합합니다. API를 사용해 환경 내에서 리소스 규모를 조정할 수 있습니다(수직적 스케일링). 예를 들어 인스턴스 크기나 클래스를 변경하여 프로덕션 워크로드의 규모를 스케일 업할 수 있습니다. 이렇게 하려면 인스턴스를 중지했다가 시작한 후 다른 인스턴스 크기나 클래스를 선택합니다. 크기를 늘리거나, 성능(IOPS)을 조정하거나, 사용 중에 볼륨 유형을 변경하기 위해 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 규모 조정을 구성합니다. 시간 기반 일정을 구성하기 위해 예약된 규모 조정의 예측 규모 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 사용하여 트래픽 흐름의 일일 및 주간 패턴에 앞서 Auto Scaling 그룹의 Amazon EC2 인스턴스 수를 늘릴 수 있습니다. 시작하는 데 오래 걸리는 애플리케이션이 있고 정기적으로 트래픽이 급증하는 경우 예측 규모 조정 사용을 고려해야 합니다. 예측 규모 조정은 예측한 로드가 발생 전에 용량을 초기화함으로써 반응적인 속성의 동적 규모 조정을 단독으로 사용하는 것과 비교하여 더 빠르게 규모를 조정할 수 있도록 합니다. 예를 들어, 사용자가 업무 시간 시작과 함께 워크로드를 사용하기 시작하고 업무 시간이 지나면 사용하지 않는 경우, 예측 규모 조정은 업무 시간 전에 용량을 추가할 수 있습니다. 그러면 변화하는 트래픽에 대응하기 위한 동적 규모 조정의 지연이 사라집니다.

- 동적 자동 규모 조정 구성: 활성 워크로드 지표를 기반으로 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 플릿를 자동으로 확장할 수 있습니다. 스팟 인스턴스 사용에 대한 할인을 받을 수 있을 뿐만 아니라 예약 인스턴스 또는 Savings Plans를 사용하여 일반 온디맨드 인스턴스 요금의 할인된 요금을 받을 수 있습니다. 이러한 모든 요소를 결합하면 Amazon EC2 인스턴스의 비용 절감을 최적화할 수 있으며 애플리케이션에서 원하는 규모 및 성능을 얻을 수 있습니다.

리소스

관련 문서:

- AWS Auto Scaling
- AWS Instance Scheduler
- Auto Scaling 그룹의 크기 조정
- Getting Started with Amazon EC2 Auto Scaling
- Getting started with Amazon SQS
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Predictive scaling for Amazon EC2 Auto Scaling

관련 비디오:

- Target Tracking Scaling Policies for Auto Scaling
- AWS Instance Scheduler

관련 예제:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive Scaling with Amazon EC2 Auto Scaling
- AWS CloudFormation과 함께 Instance Scheduler를 사용하여 EC2 인스턴스를 예약하려면 어떻게 해야 하나요?

시간 경과에 따른 최적화

Questions

- COST 10. 새로운 서비스를 어떻게 평가하나요?
- COST 11. 작업 비용을 어떻게 평가하나요?

COST 10. 새로운 서비스를 어떻게 평가하나요?

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 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 변동성을 고려합니다. 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 비

용이 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 활동과 문제 해결 방법을 식별 및 검증하고 우선순위를 지정한 후 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 검토하며, 12개월마다 데이터베이스를 검토하고, 6개월마다 컴퓨팅 및 단기 스토리지를 검토하며, 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 예제:

- AWS Support Proactive Services
- SAP 워크로드에 대한 정기 검토를 계획

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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 AWS에서 This is My Architecture 동영상을 검토하세요. All-In 시리즈에서는 고객 사례와 AWS 서비스의 적용 실례를 살펴볼 수 있습니다. 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 솔루션

스 아키텍트의 아키텍트 지침을 받을 수 있습니다. 마지막으로, 단계별 자습서로 이루어진 시작하기 리소스 자료를 검토할 수 있습니다.

검토 프로세스를 시작하기 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

- 정기적 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.
- 새 서비스 구현: 분석 결과가 변경 실행인 경우 먼저 워크로드의 기준을 수행하여 현재 성과당 비용을 파악합니다. 변경을 실행한 다음 분석을 수행하여 결과별로 새로운 비용을 확인합니다.

리소스

관련 문서:

- AWS 뉴스 블로그
- AWS의 새로운 소식
- AWS 설명서
- AWS Getting Started
- AWS 일반 리소스

관련 비디오:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In series
- How to Build This

COST 11. 작업 비용을 어떻게 평가하나요?

모범 사례

- COST11-BP01 운영 자동화 실행

COST11-BP01 운영 자동화 실행

자동화를 통해 관리 작업, 배포, 인적 오류 위험 완화, 규정 준수 및 기타 운영에 드는 시간과 노력을 정량화하는 데 중점을 두고 클라우드에서의 운영 비용을 평가합니다. 운영 노력에 필요한 시간과 관련 비용을 평가하고 가능한 경우 수동 작업을 최소화하기 위해 관리 작업을 자동화합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

운영 자동화를 통해 워크로드를 배포, 관리 또는 운영할 때 일관되고 안정적인 환경을 제공할 수 있어 수동 작업의 빈도를 줄이고, 효율성을 개선하며, 결과적으로 고객에게 이익을 안겨줄 수 있습니다. 인프라 리소스를 수동 운영 작업에 투입할 필요가 없어 대신 가치가 높은 태스크와 혁신에 투자할 수 있으므로 비즈니스 가치가 향상됩니다. 기업은 클라우드에서 워크로드를 관리할 수 있는 검증된 방법을 필요로 합니다. 이 솔루션은 위험과 신뢰성을 극대화하면서 안전하고 빠르며 비용 효율적이어야 합니다.

먼저 전반적인 운영 비용을 살펴보며 필요한 작업을 기준으로 운영 활동의 우선순위를 정하는 것부터 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 작업에 대한 자동화 우선순위를 지정하여 수동 작업의 비율을 줄이세요.

검토 작업에는 잠재적 이점이 반영되어야 합니다. 예를 들어 자동이 아닌 수동으로 작업을 수행하는 데 소요된 시간을 검토할 수 있습니다. 반복적이고 가치가 높으며 시간이 많이 걸리고 복잡한 활동을 자동화하는 데 우선순위를 둡니다. 작업자가 실수할 위험이 높거나 그 영향력이 큰 활동은 보통 자동화하면 좋습니다. 이러한 위험으로 인해 원치 않는 추가 운영 비용이 발생하는 경우가 많기 때문입니다(예: 운영 팀의 초과 근무).

AWS Systems Manager 또는 AWS Config과 같은 자동화 도구를 사용하여 운영, 규정 준수, 모니터링, 수명 주기 및 종료 프로세스를 간소화합니다. AWS 서비스, 도구 및 서드파티 제품을 사용하여 특정 요구 사항에 맞게 구현하는 자동화를 사용자 지정할 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 사용량을 지속적으로 감사하여 위험 및 규정 준수 평가를 간소화합니다.
- AWS Backup: 데이터 보호를 중앙에서 관리하고 자동화합니다.
- AWS Config: 컴퓨팅 리소스를 구성하고, 구성 및 리소스 인벤토리를 평가, 감사, 검증합니다.
- AWS CloudFormation: 코드형 인프라를 사용하여 가용성이 높은 리소스를 실행합니다.

- AWS CloudTrail: IT 변경 관리, 규정 준수 및 제어.
- Amazon EventBridge는 이벤트를 예약하고 AWS Lambda를 트리거하여 조치를 취합니다.
- AWS Lambda: 반복 프로세스를 이벤트로 트리거하거나 AWS 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: 템플릿 사용, 규정 준수 및 제어 기능을 갖춘 코드형 인프라.

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 자동화하는 데 도움이 됩니다. 이를 통해 고객은 클라우드에 최적화된 자동화되고 강력하며 민첩한 IT 운영 및 거버넌스 기능을 배포할 수 있습니다. 자세한 모니터링 예제와 권장 모범 사례는 운영 우수성 원칙 백서를 참조하세요.

구현 단계

- 한 번 구축으로 여러 번 배포: CloudFormation, AWS SDK 또는 AWS CLI와 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 배포를 안내합니다. 또한 Service Catalog를 사용하면 누구나 승인된 셀프 서비스 클라우드 리소스를 검색할 수 있도록 AWS에서 사용 가능한 코드형 인프라 승인 템플릿을 생성하고 관리할 수 있습니다.
- 지속적인 규정 준수의 자동화: 사전 정의된 표준에 따라 기록된 구성을 자동으로 평가 및 수정하는 것을 고려해 보세요. AWS Organizations를 AWS Config 및 AWS CloudFormation의 기능과 함께 사용하면 수백 개의 구성원 계정에 대해 대규모 구성 준수를 효율적으로 관리하고 자동화할 수 있습니다. 구성 변경 사항 및 AWS 리소스 간 관계를 검토하고 리소스 구성 기록을 자세히 살펴볼 수 있습니다.

- 모니터링 작업 자동화: AWS에서는 서비스를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 이러한 도구를 구성하여 모니터링 작업을 자동화할 수 있습니다. 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록 워크로드의 모든 부분에서 모니터링 데이터를 수집하는 모니터링 계획을 만들고 구현하세요. 예를 들어, 자동화된 모니터링 도구를 사용하여 시스템 상태 검사, 인스턴스 상태 검사 및 Amazon CloudWatch 경보에 문제가 있는 경우 Amazon EC2를 관찰하여 보고하도록 할 수 있습니다.
- 유지 관리 및 운영 자동화: 사람의 개입 없이 일상적인 작업을 자동으로 실행합니다. AWS 서비스 및 도구를 사용하면 특정 요구 사항에 맞게 구현하고 사용자 지정할 AWS 자동화를 선택할 수 있습니다. 예를 들어 EC2 Image Builder를 사용하여 AWS 또는 온프레미스에서 사용할 가상 머신 및 컨테이너 이미지를 구축, 테스트, 배포하거나 AWS SSM으로 EC2 인스턴스를 패치할 수 있습니다. AWS 서비스로 원하는 작업을 수행할 수 없거나 리소스 필터링을 통해 더 복잡한 작업이 필요한 경우, AWS Command Line Interface(AWS CLI) 또는 AWS SDK 도구를 사용하여 운영을 자동화하세요. AWS CLI는 AWS Management Console을 사용하지 않고 스크립트를 통해 AWS 서비스를 제어하고 관리하는 전체 프로세스를 자동화하는 기능을 제공합니다. AWS 서비스와 상호 작용할 기본 AWS SDK를 선택합니다. 다른 코드 예제는 AWS SDK 코드 예제 리포지토리를 참조하세요.
- 자동화를 통해 지속적인 수명 주기 생성: 규정이나 중복성뿐만 아니라 비용 최적화를 위한 성숙한 수명 주기 정책을 수립하고 유지하는 것이 중요합니다. AWS Backup을 사용하여 버킷, 볼륨, 데이터베이스, 파일 시스템과 같은 데이터 저장소의 데이터 보호를 중앙에서 관리하고 자동화할 수 있습니다. 또한 Amazon Data Lifecycle Manager를 사용하여 EBS 스냅샷 및 EBS 지원 AMI의 생성, 보존 및 삭제를 자동화할 수 있습니다.
- 불필요한 리소스 삭제: 샌드박스 또는 개발 AWS 계정에서 미사용 리소스를 쌓아놓는 경우는 매우 흔합니다. 개발자는 일반적인 개발 주기의 일부로 다양한 서비스와 리소스를 만들고 실험한 다음 더 이상 필요하지 않게 된 리소스를 삭제하지 않습니다. 미사용 리소스는 조직에 불필요하고 때로는 높은 비용을 초래할 수 있습니다. 이러한 리소스를 삭제하면 이러한 환경을 운영하는 데 드는 비용을 절감할 수 있습니다. 확실하지 않은 경우 데이터가 필요하지 않거나 백업되지 않음을 확인하세요. AWS CloudFormation을 사용하여 배포된 스택을 정리할 수 있습니다. 이렇게 하면 템플릿에 정의된 대부분의 리소스가 자동으로 삭제됩니다. 또는 aws-nuke와 같은 도구를 사용하여 AWS 리소스 삭제를 자동화할 수 있습니다.

리소스

관련 문서:

- Modernizing operations in the AWS 클라우드
- AWS Services for Automation
- Infrastructure and automation

- AWS Systems Manager Automation
- 자동 및 수동 모니터링
- AWS automations for SAP administration and operations
- AWS Managed Services
- AWS Professional Services

관련 비디오:

- Automate Continuous Compliance at Scale in AWS
- AWS Backup Demo: Cross-Account & Cross-Region Backup
- Patching for your Amazon EC2 Instances

관련 예제:

- Reinventing automated operations (Part I)
- Reinventing automated operations (Part II)
- Automate deletion of AWS resources by using aws-nuke
- Delete unused Amazon EBS volumes by using AWS Config and AWS SSM
- Automate continuous compliance at scale in AWS
- IT Automations with AWS Lambda

지속 가능성

지속 가능성 원칙에는 클라우드 워크로드를 구축할 때 사용된 서비스의 영향을 이해하고, 전체 워크로드 수명 주기에 걸쳐 영향을 정량화하며, 이러한 영향을 줄이기 위한 설계 원칙과 모범 사례를 적용하는 작업이 포함됩니다. 구현에 대한 권장 가이드는 지속 가능성 원칙 백서에서 확인할 수 있습니다.

모범 사례 영역

- 리전 선택
- 수요에 맞춘 조정
- 소프트웨어 및 아키텍처
- Data
- 하드웨어 및 서비스

- 프로세스 및 문화

리전 선택

질문

- SUS 1 워크로드에 적합한 리전을 선택하려면 어떻게 해야 하나요?

SUS 1 워크로드에 적합한 리전을 선택하려면 어떻게 해야 하나요?

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 확립의 이점: Amazon 재생 에너지 프로젝트 또는 게시된 탄소 집약도가 낮은 리전과 가까운 곳에 워크로드를 배치하면 클라우드 워크로드의 탄소 발자국을 줄이는 데 도움이 될 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 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

- 잠재적 리전 최종 후보 결정: 규정 준수, 사용 가능한 기능, 비용 및 지연 시간을 비롯한 비즈니스 요구 사항을 기준으로 다음 단계를 따라 워크로드의 잠재적 리전을 평가하고 최종 후보를 결정할 수 있습니다.
  - 필수 지역 규정(예: 데이터 주권)에 따라 해당 리전이 이를 준수하는지 확인합니다.
  - AWS 리전별 서비스 목록을 사용하여 해당 리전에 워크로드 실행에 필요한 서비스와 기능이 있는지 확인합니다.
  - AWS Pricing Calculator를 사용하여 각 리전에서 워크로드 비용을 계산합니다.
  - 최종 사용자 위치와 각 AWS 리전 사이의 네트워크 지연 시간을 테스트합니다.
- 리전 선택: Amazon 재생 에너지 프로젝트 근처의 리전 및 그리드의 탄소 집약도가 다른 위치(또는 리전)보다 낮은 리전을 선택합니다.
  - 온실 가스 협약(시장 기반 및 위치 기반 방법)을 기준으로 매년 탄소 배출량을 추적 및 비교하기 위해 관련 지속 가능성 지침을 식별합니다.
  - 탄소 배출량을 추적하는 데 사용하는 방법을 기준으로 리전을 선택합니다. 지속 가능성 지침을 기반으로 리전을 선택하는 방법에 대한 자세한 내용은 How to select a Region for your workload based on sustainability goals를 참조하세요.

리소스

관련 문서:

- Understanding your carbon emission estimations
- Amazon Around the Globe
- Renewable Energy Methodology
- What to Consider when Selecting a Region for your Workloads

관련 비디오:

- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

수요에 맞춘 조정

질문

- SUS 2 클라우드 리소스를 비즈니스 요구에 어떻게 맞추나요?

SUS 2 클라우드 리소스를 비즈니스 요구에 어떻게 맞추나요?

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. 사용자가 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 인프라 동적 규모 조정
- 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 인프라 동적 규모 조정

클라우드의 탄력성을 활용하고 인프라를 동적으로 조정하여, 클라우드 리소스 공급을 수요에 맞게 조정하고 워크로드의 용량 초과 프로비저닝을 방지할 수 있습니다.

일반적인 안티 패턴:

- 사용자 로드에 따라 인프라 규모를 조정하지 않습니다.
- 항상 인프라 규모를 수동으로 조정합니다.
- 조정 이벤트 후에 다시 스케일 다운하는 대신 증가된 용량을 그대로 둡니다.

이 모범 사례 확립의 이점: 워크로드 탄력성을 구성하고 테스트하면 클라우드 리소스 공급을 수요에 효율적으로 일치시키고 과도한 프로비저닝을 방지할 수 있습니다. 클라우드의 탄력성을 활용하여 수요

가 급증하는 도중과 그 이후에 용량을 자동으로 조정하여 비즈니스 요구 사항을 충족하는 데 필요한 리소스만큼만 사용할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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 스케일링하거나 인스턴스 수를 수정하여 수평(안 또는 밖으로)으로 스케일링하거나 둘을 모두 조합하여 규모를 조정할 수 있습니다.

다양한 접근 방식을 사용하여 리소스 공급과 수요를 일치시킬 수 있습니다.

- 타겟 추적 접근 방식: 규모 조정 지표를 모니터링하고 필요에 따라 용량을 자동으로 늘리거나 줄입니다.
- 예측 규모 조정: 일별 및 주별 추세를 고려하여 스케일 인합니다.
- 일정 기반 접근 방식: 예측 가능한 로드 변화에 따라 자체 규모 조정 일정을 설정합니다.
- 서비스 규모 조정: 기본적으로 설계별로 규모 조정되는 서버리스와 같은 서비스를 선택하거나 Auto Scaling을 기능으로 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 규모 조정과 함께 또는 서비스의 기능을 사용하여 탄력성을 지원하는 메커니즘을 제공합니다. AWS는 사용자 로드가 적은 기간에 워크로드를 빠르고 쉽게 스케일 다운할 수 있도록 다양한 Auto Scaling 메커니즘을 제공합니다. 다음은 Auto Scaling 메커니즘 예제입니다.

Auto Scaling 메커니즘	사용 장소
Amazon EC2 Auto Scaling	애플리케이션의 사용자 로드를 처리하는 데 사용할 수 있는 적절한 수의 Amazon EC2 인스턴스가 있는지 확인하는 데 사용합니다.
Application Auto Scaling	Lambda 함수 또는 Amazon Elastic Container Service(Amazon ECS) 서비스와 같이 Amazon EC2 이외의 개별 AWS 서비스에서 리소스 규모를 자동으로 조정하는 데 사용합니다.
Kubernetes Cluster Autoscaler	AWS에서 Kubernetes 클러스터를 자동으로 조정하는 데 사용합니다.

- 규모 조정은 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 다운에 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 규모 조정 정책에 사용자 지정 지표(예: 메모리 사용률)를 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 그룹의 인스턴스 수에 비례하여 증가하거나 감소합니다.
- Auto Scaling 그룹의 경우 수동 규모 조정 대신 동적 규모 조정을 사용합니다. 또한 동적 규모 조정에서 목표 추적 조정 정책을 사용하는 것이 좋습니다.
- 워크로드 배포에서 스케일 아웃 및 스케일 인 이벤트를 모두 처리할 수 있는지 확인합니다. 스케일 인 이벤트에 대한 테스트 시나리오를 생성하여 워크로드가 예상대로 작동하고 사용자 환경에 영향(예: 스티키 세션 손실)을 미치지 않는지 확인합니다. 활동 내역을 사용하여 Auto Scaling 그룹의 조정 활동을 확인할 수 있습니다.
- 워크로드의 예측 가능한 패턴을 평가하고 예측 및 계획된 수요 변화에 따라 사전 예방적으로 확장합니다. 예측 규모 조정에서는 용량을 과도하게 프로비저닝할 필요가 없습니다. 자세한 내용은 Predictive Scaling with Amazon EC2 Auto Scaling을 참조하세요.

리소스

관련 문서:

- Getting Started with Amazon EC2 Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Analyze user behavior using Amazon OpenSearch Service, Amazon Data Firehose and Kibana
- Amazon CloudWatch란 무엇인가요?
- Amazon RDS의 성능 개선 도우미로 DB 로드 모니터링
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling
- Introducing Karpenter - An Open-Source, High-Performance Kubernetes Cluster Autoscaler
- Deep Dive on Amazon ECS Cluster Auto Scaling

관련 비디오:

- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment
- AWS re:Invent 2022 - Scaling containers from one user to millions
- AWS re:Invent 2,023 - Scaling FM inference to hundreds of models with Amazon SageMaker AI
- AWS re:Invent 2023 - Harness the power of Karpenter to scale, optimize & upgrade Kubernetes

관련 예제:

- Autoscaling

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: SLA를 지속 가능성 목표에 맞게 조정하면 비즈니스 요구 사항을 충족하는 동시에 리소스를 최적으로 사용할 수 있습니다.

이 모범 사례가 확립되지 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

- 지속 가능성 목표 이해: 탄소 배출 감소 또는 리소스 활용 개선과 같은 조직의 지속 가능성 목표를 식별합니다.
- SLA 검토: SLA를 평가하여 SLA가 비즈니스 요구 사항을 지원하는지 평가합니다. SLA를 초과 충족하는 경우 추가 검토를 수행합니다.
- 장단점 이해: 워크로드의 복잡성(예: 대량의 동시 사용자), 성능(예: 지연 시간), 지속 가능성에 미치는 영향(예: 필요한 리소스) 전반의 절충점을 이해합니다. 일반적으로 세 번째 요소 대신 첫 두 요소가 우선 고려됩니다.
- SLA 조정: 허용 가능한 수준으로 서비스를 줄여 지속 가능성에 미치는 영향을 크게 줄이는 절충안을 제시합니다.
  - 지속 가능성 및 신뢰성: 가용성이 높은 워크로드는 리소스를 더 많이 소비하는 경향이 있습니다.
  - 지속 가능성 및 성능: 성능을 높이기 위해 더 많은 리소스를 사용하면 환경에 더 큰 영향을 미칠 수 있습니다.
  - 지속 가능성 및 보안: 워크로드 보안이 지나치면 환경에 더 큰 영향을 미칠 수 있습니다.
- 가능한 경우 지속 가능성 SLA 정의: 워크로드에 대한 지속 가능성 SLA를 포함합니다. 예를 들어 최소 사용률 수준을 컴퓨팅 인스턴스의 지속 가능성 SLA로 정의합니다.
- 효율적인 설계 패턴 사용: 비즈니스에 중요한 기능에 우선순위를 두고 중요하지 않은 기능에 대해 더 낮은 서비스 수준(예: 응답 시간 또는 복구 시간 목표)을 허용하는 AWS의 마이크로 서비스와 같은 설계 패턴을 사용합니다.
- 의사 소통 및 책임 확립: 개발팀 및 고객을 포함한 모든 관련 이해관계자와 SLA를 공유합니다. 보고 기능을 사용하여 SLA를 추적하고 모니터링합니다. SLA의 지속 가능성 목표를 달성하기 위한 책임을 할당합니다.

- 인센티브 및 보상 사용: 인센티브와 보상을 사용하여 지속 가능성 목표에 맞는 SLA를 달성하거나 초과 달성합니다.
- 검토 및 반복: SLA가 진화하는 지속 가능성 및 성능 목표에 부합하는지 정기적으로 검토하고 조정합니다.

리소스

관련 문서:

- Understand resiliency patterns and trade-offs to architect efficiently in the cloud
- Importance of Service Level Agreement for SaaS Providers

관련 비디오:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

SUS02-BP03 미사용 자산의 생성 및 유지 관리 중지

워크로드에서 미사용 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: 미사용 자산을 제거하면 리소스가 절약되고 워크로드의 전반적인 효율성이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

미사용 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, 데이터세트, 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

- 인벤토리 구성: 포괄적인 인벤토리를 구성하여 워크로드 내의 모든 자산을 식별합니다.
- 사용 분석: 지속적인 모니터링을 사용하여 더 이상 필요하지 않은 정적 자산을 식별합니다.
- 미사용 자산 제거: 더 이상 필요하지 않은 자산을 제거할 계획을 세웁니다.
  - 자산을 제거하기 전에 제거로 인해 아키텍처가 받는 영향을 평가합니다.
  - 중복 생성 자산을 통합하여 중복 처리를 제거합니다.
  - 애플리케이션을 업데이트하여 더 이상 필요 없는 자산을 생성하고 저장하지 않습니다.
- 서드파티와 커뮤니케이션: 더 이상 필요하지 않은 관리 자산의 생산 및 저장을 중단하도록 서드파티에 지시합니다. 중복 자산 통합을 요청합니다.
- 수명 주기 정책 사용: 수명 주기 정책을 사용하여 미사용 자산을 자동으로 삭제합니다.
  - Amazon S3 수명 주기를 사용하여 전체 수명 주기 동안 객체를 관리할 수 있습니다.
  - Amazon Data Lifecycle Manager를 사용하여 Amazon EBS 지원 AMI 및 Amazon EBS 스냅샷의 생성, 보존 및 삭제를 자동화할 수 있습니다.
- 검토 및 최적화: 워크로드를 정기적으로 검토하여 사용되지 않는 자산을 식별하고 제거합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- AWS 계정 계정에서 더 이상 필요하지 않은 활성 리소스를 종료하려면 어떻게 해야 하나요?

관련 비디오:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Preserving and maximizing the value of digital media assets using Amazon S3

- AWS re:Invent 2023 - Optimize costs in your multi-account environments

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 줄이며 환경에 미치는 영향을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

AWS 클라우드 인프라는 리전, 가용 영역, 배치 그룹, 엣지 로케이션(예: AWS Outposts 및 AWS 로컬 영역)과 같은 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 줄입니다.

구현 단계

- 워크로드의 네트워크 액세스 패턴을 분석하여 사용자가 애플리케이션을 사용하는 방법을 식별합니다.
  - Amazon CloudWatch 및 AWS CloudTrail과 같은 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 선택에서 설명합니다.
  - 데이터 위치: 데이터를 많이 사용하는 애플리케이션의 경우(예: 빅 데이터 및 기계 학습) 애플리케이션 코드는 최대한 데이터와 가까운 위치에서 실행되어야 합니다.

  - 사용자 위치: 사용자가 직접 사용하는 애플리케이션의 경우 워크로드의 사용자와 가까운 하나 이상의 리전을 선택합니다.
  - 기타 제약 조건: What to Consider when Selecting a Region for your Workloads에 나와 있는 비용 및 규정 준수 등 제약 요건을 고려합니다.
- 자주 사용하는 자산에 로컬 캐싱 또는 AWS Caching Solutions를 사용하여 성능을 개선하고, 데이터 이동을 줄이며, 환경에 미치는 영향을 줄입니다.

Service	사용해야 하는 경우
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.

Service	사용해야 하는 경우
Lambda@Edge	객체가 캐시에 없는 경우 시작되는 컴퓨팅 집약적 작업에 사용합니다.
Amazon CloudFront Functions	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
- Amazon ElastiCache 설명서
- What is Amazon CloudFront?
- Amazon CloudFront 주요 기능
- AWS 글로벌 인프라
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- 배치 그룹
- AWS 로컬 영역
- AWS Outposts

관련 비디오:

- Demystifying data transfer on AWS
- Scaling network performance on next-gen Amazon EC2 instances
- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020 - AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

관련 예제:

- AWS Networking 워크숍
- Architecting for sustainability - Minimize data movement across networks

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 사례가 확립되지 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

- 에너지 효율적인 워크스테이션 사용: 팀원에게 에너지 효율이 높은 워크스테이션과 주변 디바이스를 제공합니다. 이러한 디바이스에서 효율적인 전력 관리 기능(예: 저전력 모드)을 사용하여 에너지 사용량을 줄입니다.
- 가상화 사용: 가상 데스크톱 및 애플리케이션 스트리밍을 사용하여 업그레이드 및 디바이스 요구 사항을 제한합니다.
- 원격 협업 장려: 팀원이 Amazon Chime 또는 AWS Wickr 같은 원격 협업 도구를 사용하여 출장의 필요성 및 이와 관련된 탄소 배출량을 줄이도록 장려합니다.
- 에너지 효율적인 소프트웨어 사용: 불필요한 기능 및 프로세스를 제거하거나 해제하여 팀원에게 에너지 효율적인 소프트웨어를 제공합니다.

- 수명 주기 관리: 디바이스 수명 주기에 대한 프로세스 및 시스템의 영향을 평가하고 비즈니스 요구 사항을 충족하면서 디바이스 교체 요구 사항을 최소화하는 솔루션을 선택합니다. 워크스테이션이나 소프트웨어를 정기적으로 유지 관리하고 업데이트하여 효율성을 유지하고 개선합니다.
- 원격 디바이스 관리: 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?
- Cost Optimizer for Amazon WorkSpaces
- Amazon AppStream 2.0 설명서
- NICE DCV

관련 비디오:

- Managing cost for Amazon WorkSpaces on AWS

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.

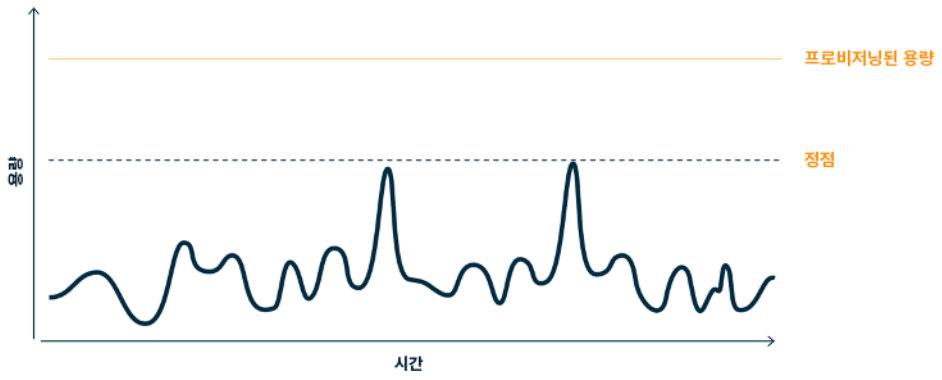
이 모범 사례 확립의 이점: 수요 곡선을 단순화하면 워크로드에 필요한 프로비저닝 용량이 줄어듭니다. 프로비저닝 용량을 줄이면 에너지 소비와 환경에 미치는 영향도 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

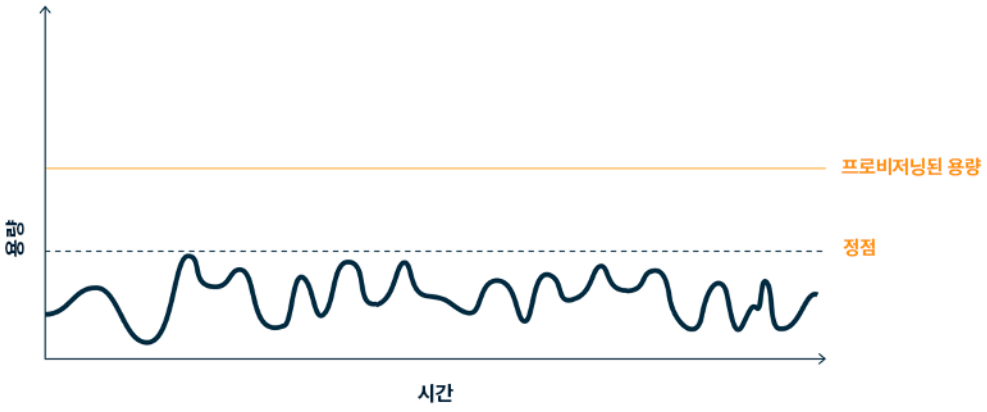
워크로드 수요 곡선을 완화하면 워크로드에 프로비저닝된 용량을 줄이고 환경에 미치는 영향도 줄일 수 있습니다. 아래 그림에 표시된 수요 곡선을 바탕으로 워크로드를 가정합니다. 이 워크로드에는 2개

의 피크가 있으며, 이러한 피크를 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 2개의 피크를 처리하려면 프로비저닝된 용량이 필요하기 때문입니다.



높은 프로비저닝 용량을 필요로 하는 두 개의 피크가 있는 수요 곡선.

버퍼링 또는 제한을 사용하여 수요 곡선을 수정하고 피크를 완화할 수 있습니다. 이렇게 되면 프로비저닝된 용량과 소비되는 에너지가 줄어듭니다. 클라이언트가 재시도를 수행할 때 제한을 구현합니다. 요청을 저장하고 나중에 처리하도록 버퍼링을 구현합니다.



제한이 수요 곡선과 프로비저닝된 용량에 미치는 영향.

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:

- Getting started with Amazon SQS
- Application integration Using Queues and Messages
- Managing and monitoring API throttling in your workloads
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Application integration Using Queues and Messages

관련 비디오:

- AWS re:Invent 2022 - Application integration patterns for microservices
- AWS re:Invent 2023 - Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

소프트웨어 및 아키텍처

질문

- SUS 3 소프트웨어 및 아키텍처 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 하나요?

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

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 미사용 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

구성 요소의 사용률을 균등하게 하고 워크로드의 과다한 프로비저닝을 최소화하는 이벤트 기반 아키텍처와 같은 효율적인 아키텍처 패턴을 사용합니다. 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 Auto Scaling 작업자를 사용하여 사용률을 극대화합니다. 대기열 기반 아키텍처를 고려해야 하는 몇 가지 예는 다음과 같습니다.

대기열 처리 메커니즘	설명
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 SQS 및 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.

예약 메커니즘	설명
Amazon EventBridge Scheduler	Amazon EventBridge의 기능으로, 예약된 작업을 대규모로 생성, 실행 및 관리할 수 있습니다.

예약 메커니즘	설명
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 태스크는 Amazon EventBridge 규칙을 사용하여 일정에 따라 또는 EventBridge 이벤트에 대한 응답으로 태스크를 실행합니다.
Instance Scheduler	Amazon EC2 및 Amazon Relational Database Service 인스턴스의 시작 및 중지 일정을 구성합니다.

- 아키텍처에서 폴링과 웹후크 메커니즘을 사용하는 경우 이를 이벤트로 바꿉니다. 이벤트 기반 아키텍처를 사용하여 매우 효율적인 워크로드를 구축합니다.
- AWS의 서버리스를 활용하여 과도하게 프로비저닝된 인프라를 제거합니다.
- 입력 대기 중인 유휴 리소스를 방지하기 위해 아키텍처의 개별 구성 요소의 적절한 크기를 지정합니다.
  - AWS Cost Explorer의 적정 크기 조정 권장 사항 또는 AWS Compute Optimizer를 사용하여 적정 크기 조정 기회를 식별할 수 있습니다.
  - 자세한 내용은 올바른 크기 조정: 워크로드에 맞게 인스턴스 프로비저닝을 참조하세요.

리소스

관련 문서:

- What is Amazon Simple Queue Service?
- What is Amazon MQ?
- Scaling based on Amazon SQS
- AWS Step Functions란 무엇입니까?
- AWS Lambda란 무엇입니까?
- Amazon SQS에서 AWS Lambda 사용
- What is Amazon EventBridge?
- Managing Asynchronous Workflows with a REST API

관련 비디오:

- AWS re:Invent 2023 - Navigating the journey to serverless event-driven architecture
- AWS re:Invent 2023 - Using serverless for event-driven architecture & domain-driven design
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- Asynchronous Message Patterns | AWS Events

관련 예제:

- AWS Graviton 프로세서 및 Amazon EC2 스팟 인스턴스를 사용하는 이벤트 기반 아키텍처

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: 미사용 구성 요소를 제거하면 낭비를 최소화하고 클라우드 워크로드의 전반적인 효율성을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

클라우드 워크로드에서 사용되지 않거나 사용률이 낮은 구성 요소는 불필요한 컴퓨팅, 스토리지 또는 네트워크 리소스를 사용합니다. 이러한 구성 요소를 제거하거나 리팩터링하여 낭비를 직접 줄이고 클라우드 워크로드의 전반적인 효율성을 개선합니다. 이는 수요 변화 또는 새로운 클라우드 서비스 출시로 시작될 수 있는 반복적인 개선 프로세스입니다. 예를 들어, AWS Lambda 함수 실행 시간의 급격한 저하는 메모리 크기를 줄여야 하는 필요성을 나타낼 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

- AWS 리소스 인벤토리 생성: AWS 리소스 인벤토리를 생성합니다. AWS에서 AWS 리소스 탐색기를 활성화하여 AWS 리소스를 탐색하고 구성할 수 있습니다. 자세한 내용은 AWS re:Invent 2022 - How to manage resources and applications at scale on AWS를 참조하세요.
- 사용률 모니터링: 워크로드의 중요한 구성 요소(예: Amazon CloudWatch 지표의 CPU 사용률, 메모리 사용률 또는 네트워크 처리량)에 대한 사용률 지표를 모니터링하고 캡처합니다.
- 사용되지 않는 구성 요소 식별: 아키텍처에서 사용되지 않거나 활용도가 낮은 구성 요소를 식별합니다.
  - 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.
- 미사용 구성 요소 제거: 더 이상 필요하지 않은 구성 요소 및 관련 자산(예: Amazon ECR 이미지)을 사용 중지합니다.
  - Automated Cleanup of Unused Images in Amazon ECR
  - Delete unused Amazon Elastic Block Store (Amazon EBS) volumes by using AWS Config and AWS Systems Manager
- 사용률이 낮은 구성 요소 리팩터링: 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.
- 개선 사항 평가: 작업 단위를 완료하기 위해 워크로드에서 프로비저닝한 리소스를 파악합니다. 이 정보를 사용하여 구성 요소를 제거하거나 리팩터링하여 달성한 개선 사항을 평가합니다.
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline

리소스

관련 문서:

- AWS Trusted Advisor
- Amazon CloudWatch란 무엇인가요?
- 올바른 크기 조정: 워크로드에 맞게 인스턴스 프로비저닝
- Optimizing your cost with Rightsizing Recommendations

관련 비디오:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three

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 사용하면 리소스 사용을 최소화하고 성능을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

- 효율적인 프로그래밍 언어 사용: 워크로드에 효율적인 운영 체제와 프로그래밍 언어를 사용합니다. 에너지 효율적인 프로그래밍 언어(Rust 포함)에 대한 자세한 내용은 Sustainability with Rust를 참조하세요.
- AI 코딩 도우미 사용: 코드를 효율적으로 작성하기 위해 Amazon Q Developer와 같은 AI 코딩 도우미를 사용하는 방법을 고려하세요.
- 코드 검토 자동화: 워크로드를 개발하는 동안 자동화된 코드 검토 프로세스를 채택하여 품질을 개선하고 버그와 안티 패턴을 식별합니다.
  - Automate code reviews with Amazon CodeGuru Reviewer
  - Detecting concurrency bugs with Amazon CodeGuru
  - Raising code quality for Python applications using Amazon CodeGuru
- 코드 프로파일러 사용: 코드 프로파일러를 사용하여 가장 많은 시간 또는 리소스를 사용하는 코드 영역을 최적화 대상으로 식별합니다.
  - Reducing your organization's carbon footprint with Amazon CodeGuru Profiler
  - Understanding memory usage in your Java application with Amazon CodeGuru Profiler
  - Improving customer experience and reducing cost with Amazon CodeGuru Profiler
- 모니터링 및 최적화: 지속적인 모니터링 리소스를 사용하여 리소스 요구 사항이 높거나 구성이 최적화되지 않은 구성 요소를 식별합니다.
  - 컴퓨팅 집약적인 알고리즘을 동일한 결과를 내는 보다 단순하고 효율적인 버전으로 대체합니다.
  - 정렬 및 서식 지정과 같은 불필요한 코드를 제거합니다.
- 코드 리팩터링 또는 변환 사용: 애플리케이션 유지 관리 및 업그레이드에 Amazon Q 코드 변환을 사용할 수 있을지 살펴봅니다.
  - Upgrade language versions with Amazon Q Code Transformation
  - AWS re:Invent 2023 - Automate app upgrades & maintenance using Amazon Q Code Transformation

리소스

관련 문서:

- What is Amazon CodeGuru Profiler?
- FPGA instances
- AWS에서의 구축을 위한 도구의 AWS SDK

관련 비디오:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

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 워크로드가 환경에 미치는 전반적인 영향을 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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

- 인벤토리 구성: 아키텍처에 사용되는 디바이스의 인벤토리를 구성합니다. 디바이스는 모바일, 태블릿, IOT 디바이스, 스마트 라이트 또는 공장의 스마트 디바이스일 수 있습니다.

- 에너지 효율적인 디바이스 사용: 아키텍처에 에너지 효율적인 디바이스를 사용하는 것을 고려합니다. 사용하지 않을 때는 디바이스의 전원 관리 구성을 사용하여 저전력 모드로 전환합니다.
- 효율적인 애플리케이션 실행: 디바이스에서 실행되는 애플리케이션을 최적화합니다.
  - 백그라운드에서 작업을 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 대용량 고품질 미디어 파일을 사용자가 모바일 디바이스, 태블릿, 웹 브라우저 및 커넥티드 TV에서 재생할 수 있는 형식으로 변환할 수 있습니다.
  - 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.
- 공급업체 참여: 지속 가능한 소재를 사용하고 공급망의 투명성과 환경 인증을 제공하는 디바이스 공급업체와 협력합니다.
- 무선 업데이트(OTA) 업데이트 사용: 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 커넥티드 디바이스를 대규모로 원격으로 관리할 수 있습니다.
- 관리형 Device Farm 사용: 새로운 기능 및 업데이트를 테스트하려면 대표적인 하드웨어 집합과 함께 관리형 Device Farm을 사용하고 개발을 반복하여 지원되는 디바이스를 최대화하세요. 자세한 내용은 SUS06-BP05 테스트에 관리형 Device Farm 사용를 참조하세요.
- 지속적인 모니터링 및 개선: 디바이스의 에너지 사용량을 추적하여 개선이 필요한 부분을 식별합니다. 새로운 기술이나 모범 사례를 사용하여 이러한 디바이스가 환경에 미치는 영향을 개선합니다.

리소스

관련 문서:

- AWS Device Farm란 무엇입니까?
- AppStream 2.0 설명서
- NICE DCV

- OTA tutorial for updating firmware on devices running FreeRTOS
- Optimizing Your IoT Devices for Environmental Sustainability

관련 비디오:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm

SUS03-BP05 데이터 액세스 및 스토리지 패턴을 가장 잘 지원하는 소프트웨어 패턴 및 아키텍처 사용

데이터가 워크로드 내에서 사용되고, 사용자가 소비하며, 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 전반적인 활용도를 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

장기 워크로드 지속 가능성을 개선하려면 워크로드의 데이터 액세스 및 스토리지 특성을 지원하는 아키텍처 패턴을 사용합니다. 이러한 패턴은 데이터를 효율적으로 검색하고 처리하는 데 도움이 됩니다. 예를 들어, 고유한 분석 사용 사례에 최적화된 목적별 서비스를 통해 AWS 기반 현대적 데이터 아키텍처를 사용합니다. 이러한 아키텍처 패턴은 효율적인 데이터 처리를 가능하게 하고 리소스 사용을 줄여줍니다.

구현 단계

- 데이터 특성 이해: 데이터 특성 및 액세스 패턴을 분석하여 클라우드 리소스에 적합한 구성을 식별합니다. 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, 업데이트 빈도, 급증 또는 일관된 상태
- 최적의 아키텍처 패턴 사용: 데이터 액세스 및 스토리지 패턴을 가장 잘 지원하는 아키텍처 패턴을 사용합니다.
  - Patterns for enabling data persistence
  - Let's Architect! Modern data architectures
  - Databases on AWS: The Right Tool for the Right Job
- 목적별 서비스 사용: 목적에 적합한 기술을 사용합니다.
  - 기본적으로 압축된 데이터와 함께 작동하는 기술을 사용합니다.
    - Athena 압축 지원 파일 형식
    - Format Options for ETL Inputs and Outputs in AWS Glue
    - Amazon Redshift를 사용하여 Amazon S3에서 압축된 데이터 파일 로드
  - 아키텍처에서 데이터 처리에 목적별 분석 서비스를 사용합니다. AWS 목적별 분석 서비스에 대한 자세한 내용은 AWS re:Invent 2022 - Building modern data architectures on AWS를 참조하세요.
  - 가장 많이 나타나는 쿼리 패턴을 가장 효과적으로 지원하는 데이터베이스 엔진을 사용합니다. 효율적인 쿼리를 위해 데이터베이스 인덱스를 관리합니다. 자세한 내용은 AWS 데이터베이스 및 AWS re:Invent 2022 - Modernize apps with purpose-built databases를 참조하세요.
- 데이터 전송 최소화: 아키텍처에 사용되는 네트워크 용량을 줄이는 네트워크 프로토콜을 선택합니다.

리소스

관련 문서:

- Amazon Redshift를 사용한 열 기반 데이터 형식에서 COPY 명령
- Converting Your Input Record Format in Firehose
- 열 형식으로 변환하여 Amazon Athena에서 쿼리 성능 향상

- Amazon Aurora의 성능 개선 도우미로 DB 로드 모니터링
- Amazon RDS의 성능 개선 도우미로 DB 로드 모니터링
- Amazon S3 Intelligent-Tiering 스토리지 클래스
- Build a CQRS event store with Amazon DynamoDB

관련 비디오:

- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge

관련 예제:

- AWS Purpose Built Databases 워크숍
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

Data

질문

- SUS 4 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 하나요?

SUS 4 데이터 관리 정책 및 패턴을 활용하여 지속 가능성 목표를 지원하려면 어떻게 해야 하나요?

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 효과적으로 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 데이터세트의 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.

모범 사례 확립의 이점: 데이터 분류 정책을 구현하면 에너지 효율이 가장 높은 데이터 스토리지 계층을 확인할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 데이터세트의 중요도 수준을 고려하는 분류 체계를 만들어 데이터 분류 정책을 구현합니다.

구현 단계

- 데이터 인벤토리 작업 수행: 워크로드에 존재하는 다양한 데이터 유형의 인벤토리 작업을 수행합니다.

- 데이터 그룹화: 조직에 대한 위험을 기준으로 데이터의 중요도, 기밀성, 무결성 및 가용성을 확인합니다. 이러한 요구 사항을 사용하여 채택한 데이터 분류 티어 중 하나로 데이터를 그룹화합니다. 예를 들어, 데이터를 분류하고 Startup을 보호하는 간단한 4단계를 참조하세요.
- 데이터 분류 수준 및 정책 정의: 각 데이터 그룹에 대해 데이터 분류 수준(예: 퍼블릭 또는 기밀) 및 처리 정책을 정의합니다. 데이터에 태그를 적절히 지정합니다. 데이터 분류 범주에 대한 자세한 내용은 Data Classification 백서를 참조하세요.
- 주기적 검토: 태그가 지정되지 않은 데이터와 분류되지 않은 데이터가 있는지 환경을 정기적으로 검토하고 감사합니다. 자동화를 사용하여 이 데이터를 식별하고 데이터를 적절하게 분류하며 태그를 지정합니다. 예제로 Data Catalog and crawlers in AWS Glue를 참조하세요.
- 데이터 카탈로그 설정: 감사 및 거버넌스 기능을 제공하는 데이터 카탈로그를 설정합니다.
- 문서화: 각 데이터 클래스에 대한 데이터 분류 정책 및 처리 절차를 문서화합니다.

리소스

관련 문서:

- Leveraging AWS 클라우드 to Support Data Classification
- Tag policies from AWS Organizations

관련 비디오:

- AWS re:Invent 2022 - Enabling agility with data governance on AWS
- AWS re:Invent 2023 - Data protection and resilience with AWS storage

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 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

따라서 성능 효율성을 극대화하려면 액세스 패턴에 가장 적합한 스토리지 솔루션을 선택하거나, 스토리지 솔루션에 따라 액세스 패턴을 변경하는 것이 좋습니다.

구현 단계

- 데이터 및 액세스 특성 평가: 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, 업데이트 빈도, 급증 또는 일관된 상태
- 올바른 스토리지 기술 선택: 데이터 특성 및 액세스 패턴을 지원하는 적절한 스토리지 기술로 데이터를 마이그레이션합니다. 다음은 AWS 스토리지 기술의 몇 가지 예와 주요 특성입니다.

유형	기술	주요 특징
객체 스토리지	Amazon S3	무제한 확장성, 높은 가용성과 액세스 가능성을 위한 여러 옵션을 갖춘 객체 스토리지 서비스입니다. Amazon S3 안팎에서 객체를 전송하고 객체에 액세스할 때 전송 가속화 또는 액세스 포인트와 같은 서비스를 사용하여 위치, 보안 요구 사항 및 액세스 패턴을 지원할 수 있습니다.
아카이빙 스토리지	Amazon S3 Glacier	데이터 아카이빙을 위해 구축된 Amazon S3의 스토리지 클래스.

유형	기술	주요 특징
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 성능이 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 서비스. Amazon EBS에는 IOPS 집약적 트랜잭션 워크로드를 위한 SSD 지원 스토리지와 처리량 집약적 워크로드를 위한 HDD 지원 스토리지가 포함됩니다.

유형	기술	주요 특징
관계형 데이터베이스	Amazon Aurora, Amazon RDS, Amazon Redshift	원자성, 일관성, 격리, 내구성(ACID) 트랜잭션을 지원하고 참조 무결성과 강력한 데이터 일관성을 유지하도록 설계되었습니다. 많은 기존 애플리케이션, 엔터프라이즈 리소스 계획(ERP), 고객 관계 관리(CRM) 및 전자 상거래 시스템의 데이터가 관계형 데이터베이스를 사용하여 저장됩니다.
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.

- 스토리지 할당 자동화: 크기가 고정된 스토리지 시스템(예: Amazon EBS 또는 Amazon FSx)의 경우 사용 가능한 스토리지 공간을 모니터링하고 임곗값에 도달 시 스토리지 할당을 자동화합니다. Amazon CloudWatch를 활용하여 Amazon EBS 및 Amazon FSx에 대한 다양한 지표를 수집하고 분석할 수 있습니다.
- 올바른 스토리지 클래스 선택: 데이터에 적합한 스토리지 클래스를 선택합니다.
  - Amazon S3 스토리지 클래스는 객체 수준에서 구성할 수 있습니다. 단일 버킷에는 모든 스토리지 클래스에 저장된 객체가 포함될 수 있습니다.
  - 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 volume types

- Amazon EC2 인스턴스 스토어
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O Characteristics
- Amazon S3 스토리지 사용
- What is Amazon S3 Glacier?

관련 비디오:

- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2022 - Building modern data architectures on AWS
- AWS re:Invent 2022 - Modernize apps with purpose-built databases
- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Advanced data modeling with Amazon DynamoDB

관련 예제:

- Amazon S3 Examples
- AWS Purpose Built Databases 워크숍
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

SUS04-BP03 정책을 사용하여 데이터세트의 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 사용하면 워크로드에서 효율적인 데이터 액세스 및 보존이 보장됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

데이터세트는 일반적으로 수명 주기 동안 보존 및 액세스 요구 사항이 각각 다릅니다. 예를 들어, 애플리케이션이 한정된 기간에 일부 데이터세트에 자주 액세스해야 할 수 있습니다. 그 후 해당 데이터세트에 자주 액세스하지 않습니다. 시간 경과에 따른 데이터 스토리지 및 컴퓨팅의 효율성을 개선하려면 수명 주기 정책을 구현합니다. 수명 주기 정책은 시간 경과에 따른 데이터 처리 방식을 정의하는 규칙입니다.

수명 주기 구성 규칙을 통해, 데이터세트를 보다 에너지 효율적인 스토리지 계층으로 전환하고 아카이브하거나 삭제하도록 특정 스토리지 서비스에 요청할 수 있습니다. 이 방법은 활성 데이터 스토리지 및 검색을 최소화하여 에너지 소비를 줄입니다. 또한 더 이상 사용되지 않는 데이터를 보관하거나 삭제하는 등의 방식은 규정 준수 및 데이터 거버넌스를 지원합니다.

구현 단계

- 데이터 분류 사용: 워크로드에서 데이터 세트를 분류합니다.
- 처리 규칙 정의: 각 데이터 클래스의 처리 절차를 정의합니다.
- 자동화 활성화: 수명 주기 규칙을 적용하는 자동화된 수명 주기 정책을 설정합니다. 다양한 AWS 스토리지 서비스에 대한 자동화된 수명 주기 정책을 설정하는 몇 가지 예는 다음과 같습니다.

스토리지 서비스	자동화된 수명 주기 정책을 설정하는 방법
Amazon S3	Amazon S3 수명 주기를 사용하여 전체 수명 주기 동안 객체를 관리할 수 있습니다. 액세스 패턴을 알 수 없거나 패턴이 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.

스토리지 서비스	자동화된 수명 주기 정책을 설정하는 방법
	Amazon S3 Storage Lens 지표를 활용하여 수명 주기 관리에서 최적화 기회와 격차를 식별할 수 있습니다.
Amazon Elastic Block Store	Amazon Data Lifecycle Manager를 사용하여 Amazon EBS 지원 AMI 및 Amazon EBS 스냅샷의 생성, 보존 및 삭제를 자동화할 수 있습니다.
Amazon Elastic File System	Amazon EFS 수명 주기 관리는 파일 시스템에 대한 파일 스토리지를 자동으로 관리합니다.
Amazon Elastic 컨테이너 레지스트리	Amazon ECR 수명 주기 정책은 사용 기간 또는 개수에 따라 이미지를 만료시켜 컨테이너 이미지의 정리를 자동화합니다.
AWS Elemental MediaStore	MediaStore 컨테이너에 장기 객체를 저장해야 하는 방법을 제어하는 객체 수명 주기 정책을 사용할 수 있습니다.

- 미사용 자산 삭제: 미사용 볼륨, 스냅샷 및 보존 기간이 지난 데이터를 삭제합니다. 삭제를 위해 Amazon DynamoDB Time To Live 또는 Amazon CloudWatch 로그 보존과 같은 네이티브 서비스 기능을 사용합니다.
- 집계 및 압축: 수명 주기 규칙에 따라 관련 데이터를 집계 및 압축합니다.

리소스

관련 문서:

- Amazon S3 스토리지 클래스 분석을 사용하여 Amazon S3 수명 주기 규칙 최적화
- Evaluating Resources with AWS Config 규칙

관련 비디오:

- AWS re:Invent 2021 - Amazon S3 Lifecycle best practices to optimize your storage spend
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3

- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle
- Reduce Your Storage Costs Using Amazon S3 Storage Lens

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 시스템의 오버 프로비저닝을 최소화하면 유휴 리소스가 줄어들고 워크로드의 전반적인 효율성이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 임곗값에 도달할 때 스토리지 크기를 늘리도록 자동화를 생성합니다.
- 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 performance
  - Amazon EBS volume performance on Linux instances

- 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:

- Extend the file system after resizing an EBS volume
- Modify a volume using Amazon EBS Elastic Volumes
- Amazon FSx 설명서
- Amazon Elastic File System이란 무엇입니까?

관련 비디오:

- Deep Dive on Amazon EBS Elastic Volumes
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings
- Optimizing Amazon EFS for cost and performance, using best practices

SUS04-BP05 불필요하거나 중복된 데이터 제거

불필요하거나 중복된 데이터를 제거하여 데이터세트를 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 켭니다.

이 모범 사례 확립의 이점: 불필요한 데이터를 제거하면 워크로드에 필요한 스토리지 크기와 워크로드 환경에 미치는 영향이 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

불필요한 중복 데이터 세트를 제거하면 스토리지 비용과 환경 발자국을 줄일 수 있습니다. 컴퓨팅 리소스가 불필요한 데이터 대신 중요한 데이터만 처리하기 때문에 이 방식은 컴퓨팅을 더 효율적으로 만듭니다. 불필요한 데이터의 삭제를 자동화합니다. 파일 및 블록 수준에서 데이터 중복을 제거하는 기술을 사용합니다. 네이티브 데이터 복제 및 중복성에 대한 서비스 기능을 사용합니다.

구현 단계

- 퍼블릭 데이터세트 평가: AWS Data Exchange 및 Open Data on AWS에서 공개적으로 사용 가능한 기존 데이터세트를 사용하여 데이터를 저장하지 않아도 되는지 평가합니다.
- 데이터 중복 제거: 블록 및 객체 수준에서 데이터 중복을 제거할 수 있는 메커니즘을 사용합니다. 다음은 AWS의 데이터 중복을 제거하는 방법의 몇 가지 예입니다.

스토리지 서비스	중복 제거 메커니즘
Amazon S3	AWS Lake Formation FindMatches를 사용하여 새로운 FindMatches ML 트랜스폼을 통해 데이터세트에서 일치하는 레코드(식별자가 없는 레코드 포함)를 찾습니다.
Amazon FSx	Amazon FSx for Windows에서 데이터 중복 제거를 사용합니다.
Amazon Elastic Block Store 스냅샷	스냅샷은 증분식 백업이어서 마지막 스냅샷 이후 변경된 디바이스의 블록만이 저장됩니다.

- 수명 주기 정책 사용: 수명 주기 정책을 사용하여 불필요한 데이터를 자동으로 삭제합니다. 삭제를 위해 Amazon DynamoDB Time To Live, Amazon S3 수명 주기 또는 Amazon CloudWatch 로그 보존과 같은 기본 서비스 기능을 사용합니다.
- 데이터 가상화 사용: AWS의 데이터 가상화 기능을 사용하여 소스의 데이터를 유지 관리하고 데이터 중복을 방지합니다.
  - Cloud Native Data Virtualization on AWS
  - Optimize Data Pattern Using Amazon Redshift Data Sharing

- 증분식 백업 사용: 증분식 백업을 만들 수 있는 백업 기술을 사용합니다.
- 네이티브 내구성 사용: Amazon S3의 내구성 및 Amazon EBS의 복제를 활용하여 자체 관리형 기술(예: 독립 디스크의 이중화 어레이(RAID)) 대신 내구성 목표를 달성합니다.
- 효율적인 로깅 사용: 로그 및 추적 데이터를 중앙 집중화하고, 동일한 로그 항목을 중복 제거하며, 필요에 따라 세부적으로 조정하는 메커니즘을 설정합니다.
- 효율적인 캐싱 사용: 합당한 상황에서만 캐시를 미리 채웁니다.
- 캐시 모니터링 및 자동화를 설정하여 그에 따라 캐시 크기를 조정합니다.
- 오래된 버전의 자산 제거: 새 버전의 워크로드를 푸시할 때 객체 스토어 및 엣지 캐시에서 오래된 배포 및 자산을 제거합니다.

리소스

관련 문서:

- Change log data retention in CloudWatch Logs
- Data deduplication on Amazon FSx for Windows File Server
- Features of Amazon FSx for ONTAP including data deduplication
- Amazon CloudFront의 파일 무효화
- Using AWS Backup to back up and restore Amazon EFS file systems
- What is Amazon CloudWatch Logs?
- Amazon RDS에서 백업 작업
- Integrate and deduplicate datasets using AWS Lake Formation

관련 비디오:

- Amazon Redshift Data Sharing Use Cases

관련 예제:

- Amazon Athena를 사용하여 Amazon S3 서버 액세스 로그를 분석하려면 어떻게 해야 하나요?

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 한 명 이상의 소비자와 데이터를 공유할 수 있습니다. 이를 통해 워크로드에 필요한 스토리지 리소스를 줄일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

여러 사용자 또는 애플리케이션이 동일한 데이터세트에 액세스하는 경우 워크로드에 효율적인 인프라를 사용하기 위해서는 공유 스토리지 기술을 사용해야 합니다. 공유 스토리지 기술은 데이터세트를 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

- 공유 스토리지 사용: 데이터의 소비자가 다수인 경우 데이터를 공유 스토리지로 마이그레이션합니다. AWS 기반 공유 스토리지 기술의 몇 가지 예는 다음과 같습니다.

스토리지 옵션	사용해야 하는 경우
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 EFS를 참조하세요.
Amazon FSx	Amazon FSx 파일 시스템 선택을 참조하세요.

스토리지 옵션	사용해야 하는 경우
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.

- 필요에 따라 데이터 가져오기: 필요한 경우에만 공유 파일 시스템에 데이터를 복사하거나 데이터를 가져옵니다. 예를 들어, Amazon S3 기반 Amazon FSx for Lustre 파일 시스템을 생성하고 작업을 처리하는 데 필요한 데이터의 하위 세트만 Amazon FSx로 로드할 수 있습니다.
- 불필요한 데이터 삭제: 사용 패턴에 따라 적절하게 데이터를 삭제합니다(SUS04-BP03 정책을 사용하여 데이터세트의 수명 주기 관리 참조).
- 비활성 클라이언트 분리: 자주 사용하지 않는 클라이언트에서 볼륨을 분리합니다.

리소스

관련 문서:

- Linking your file system to an Amazon S3 bucket
- Using Amazon EFS for AWS Lambda in your serverless applications
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns
- Using Amazon FSx with your on-premises data repository

관련 비디오:

- Storage cost optimization with Amazon EFS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - File storage for builders and data scientists on Amazon Elastic File System

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 않을 경우 노출되는 위험 수준: 중간

구현 지침

조직 전체에서 데이터를 이동하려면 컴퓨팅, 네트워킹 및 스토리지 리소스가 필요합니다. 기술을 사용하여 데이터 이동을 최소화하고 워크로드의 전반적인 효율성을 개선합니다.

구현 단계

- 근접성 사용: 워크로드에 대한 리전을 선택할 때 데이터 또는 사용자와의 근접성을 결정 요소로 고려하세요.
- 서비스 분할: 리전에서 사용되는 서비스를 분할하여 해당 리전별 데이터가 사용되는 리전 내에 저장되도록 합니다.
- 효율적인 파일 형식 사용: 네트워크를 통해 데이터를 이동하기 전에 효율적인 파일 형식(예: Parquet 또는 ORC)을 사용하고 데이터를 압축합니다.
- 데이터 이동 최소화: 미사용 데이터를 이동하지 마세요. 사용하지 않는 데이터의 이동을 방지하는 데 도움이 되는 몇 가지 예는 다음과 같습니다.
  - 관련 데이터에 대한 API 응답으로만 줄입니다.
  - 상세한 경우 데이터를 집계합니다(레코드 수준 정보는 필요하지 않음).
  - Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing을 참조하세요.
  - AWS Lake Formation에서 크로스 계정 데이터 공유를 고려하세요.
- 엣지 서비스 사용: 워크로드 사용자에게 더 가까운 위치에서 코드를 실행할 수 있는 서비스를 사용합니다.

Service	사용해야 하는 경우
Lambda@Edge	객체가 캐시에 없는 경우 실행되는 컴퓨팅 집약적 작업에 사용합니다.

Service	사용해야 하는 경우
CloudFront 함수	HTTP(s) 요청 및 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
- AWS 글로벌 인프라
- CloudFront 글로벌 엣지 네트워크를 비롯한 Amazon CloudFront 주요 기능
- Amazon OpenSearch Service에서 HTTP 요청 압축
- Intermediate data compression with Amazon EMR
- Amazon Redshift를 사용하여 Amazon S3에서 압축된 데이터 파일 로드
- Serving compressed files with Amazon CloudFront

관련 비디오:

- Demystifying data transfer on AWS

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 리소스가 줄어들고 환경에 미치는 영향이 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

- 데이터 분류: SUS04-BP01 데이터 분류 정책 구현에 설명된 대로 데이터 분류 정책을 구현합니다.
- 백업 전략 설계: 데이터 분류 중요도를 사용하고 목표 복구 시간(RTO) 및 목표 복구 시점(RPO)에 기반한 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.
- 자동 백업 사용: 자동화된 솔루션 또는 관리형 서비스를 사용하여 비즈니스 크리티컬 데이터를 백업합니다.
  - AWS Backup는 AWS 서비스, 클라우드 및 온프레미스에서 데이터 보호를 중앙 집중화하고 자동화하는 데 도움이 되는 완전관리형 서비스입니다. AWS Backup을 사용하여 자동 백업을 생성하는 방법에 대한 실습 지침은 Well-Architected Labs - Testing Backup and Restore of Data를 참조하세요.
  - Automate backups and optimize backup costs for Amazon EFS using AWS Backup.

리소스

관련 모범 사례:

- REL09-BP01 백업해야 하는 모든 데이터 확인 및 백업 또는 소스에서 데이터 복제
- REL09-BP03 자동으로 데이터 백업 수행
- REL13-BP02 복구 목표 달성을 위해 정의된 복구 전략 사용

관련 문서:

- Using AWS Backup to back up and restore Amazon EFS file systems
- Amazon EBS snapshots
- Amazon Relational Database Service의 백업 작업
- APN 파트너: 백업을 지원할 수 있는 파트너
- AWS Marketplace: 백업에 사용할 수 있는 제품
- Backing Up Amazon EFS
- Backing Up Amazon FSx for Windows File Server
- Backup and Restore for Amazon ElastiCache (Redis OSS)

관련 비디오:

- AWS re:Invent 2023 - Backup and disaster recovery strategies for increased resilience
- AWS re:Invent 2023 - What's new with AWS Backup
- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS

하드웨어 및 서비스

질문

- SUS 5 지속 가능성 목표를 지원하기 위해 아키텍처에서 클라우드 하드웨어 및 서비스를 어떻게 선택하고 사용하나요?

SUS 5 지속 가능성 목표를 지원하기 위해 아키텍처에서 클라우드 하드웨어 및 서비스를 어떻게 선택하고 사용하나요?

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 워크로드가 환경에 미치는 영향을 줄이고 비용을 절감하며 성능 벤치마크를 유지할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

워크로드에 필요한 총 하드웨어 수를 최적으로 선택하여 전반적인 효율성을 개선합니다. AWS 클라우드에서는 수요 변화에 맞춰 AWS Auto Scaling과 같은 다양한 메커니즘을 통해 리소스를 동적으로 확장 또는 축소할 수 있는 유연성을 제공합니다. 또한 최소한의 노력으로 리소스를 수정할 수 있는 API 및 SDK를 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

- 인스턴스 유형 선택: 요구 사항에 가장 적합한 올바른 인스턴스 유형을 선택합니다. Amazon Elastic Compute Cloud 인스턴스를 선택하고 속성 기반 인스턴스 선택과 같은 메커니즘을 사용하는 방법에 대해 알아보려면 다음을 참조하세요.
  - 워크로드에 적합한 EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
  - Amazon EC2 플릿의 속성 기반 인스턴스 유형 선택.
  - 속성 기반 인스턴스 유형 선택을 사용하여 Auto Scaling 그룹 생성.
- 규모 조정: 가변 워크로드 규모를 조정하기 위해 작은 증분 단위를 사용합니다.
- 다양한 컴퓨팅 구매 옵션 사용: 여러 컴퓨팅 구매 옵션으로 인스턴스 유연성, 확장성, 비용 절감의 균형을 조정합니다.

  - Amazon EC2 온디맨드 인스턴스는 인스턴스 유형, 위치 또는 시간을 유연하게 지정할 수 없는 상태 저장 방식의 새로운 급증 워크로드에 가장 적합합니다.
  - Amazon EC2 스팟 인스턴스는 내결함성 및 유연성이 뛰어난 애플리케이션에서 다른 옵션을 보완하는 데 유용한 방법입니다.
  - Compute Savings Plans를 활용하면 요구 사항(예: AZ, 리전, 인스턴스 패밀리 또는 인스턴스 유형)이 변경될 경우 유연성을 확보할 수 있는 안정적인 상태의 워크로드에 적합합니다.
- 인스턴스 및 가용 영역 다양성 사용: 인스턴스 및 가용 영역을 다양화하여 애플리케이션 가용성을 극대화하고 초과 용량을 활용합니다.
- 인스턴스 적정 크기 조정: AWS 도구에서 적정 크기 조정 권장 사항을 사용하여 워크로드를 조정합니다. 자세한 내용은 Optimizing your cost with Rightsizing Recommendations 및 Right Sizing: Provisioning Instances to Match Workloads를 참조하세요.
  - AWS Cost Explorer의 적정 크기 조정 권장 사항 또는 AWS Compute Optimizer를 사용하여 적정 크기 조정 기회를 식별합니다.
- 서비스 수준에 관한 계약(SLA) 협상: 자동화가 대체 리소스를 배포하는 동안 일시적으로 용량을 줄일 수 있는 SLA를 협상합니다.

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- Attirbute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- AWS Compute Optimizer 설명서
- Operating Lambda: Performance optimization
- Auto Scaling 설명서

관련 비디오:

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2022 - Optimizing Amazon Elastic Kubernetes Service for performance and cost on AWS
- AWS re:Invent 2023 - Sustainable compute: reducing costs and carbon emissions with AWS

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 하나의 인스턴스 유형을 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)에서 권장 사항을 확인하고 분석하지 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 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

- 인스턴스 유형 학습 및 탐색: 워크로드가 환경에 미치는 영향을 줄일 수 있는 인스턴스 유형을 찾습니다.
  - AWS의 새로운 소식을 구독하여 최신 AWS 기술 및 인스턴스에 대한 최신 정보를 확인합니다.
  - 다양한 AWS 인스턴스 유형에 대해 알아봅니다.
  - re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances 및 Deep dive into AWS Graviton3 and Amazon EC2 C7g instances를 보고 Amazon EC2에서 와트당 에너지 사용 대비 최고 성능을 제공하는 AWS Graviton 기반 인스턴스에 대해 알아보세요.
- 영향이 가장 적은 인스턴스 유형 사용: 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 Graviton으로 워크로드를 이동하는 방법에 대한 자세한 내용은 AWS Graviton Fast Start 및 Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances를 참조하세요.
- AWS 관리형 서비스를 사용할 때 AWS Graviton 옵션 선택을 고려하세요.
- 지속 가능성에 미치는 영향이 가장 적고 비즈니스 요구 사항을 충족하는 인스턴스를 제공하는 리전으로 워크로드를 마이그레이션합니다.
- 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia 및 Amazon EC2 DL1과 같이 워크로드에 따라 특정한 목적별 하드웨어의 장점을 활용합니다. AWS Inf2 인스턴스와 같은 Inferentia 인스턴스는 동급 Amazon EC2 인스턴스에 비해 최대 50% 더 우수한 와트당 성능을 제공합니다.
- Amazon SageMaker AI Inference Recommender를 사용하여 ML 추론 엔드포인트를 적정 크기로 조정합니다.
- 사용량이 급증하는 워크로드의 경우(추가 용량에 대한 요구 사항이 적은 워크로드) 성능 버스트 가능 인스턴스를 사용합니다.
- 상태 비저장 및 내결함성 워크로드의 경우 Amazon EC2 스팟 인스턴스를 사용하여 클라우드의 전체 활용률을 높이고 미사용 리소스가 지속 가능성에 미치는 영향을 줄입니다.

- 운영 및 최적화: 워크로드 인스턴스를 운영 및 최적화합니다.
  - 임시 워크로드의 경우 인스턴스 Amazon CloudWatch 지표(예: `CPUUtilization`)를 평가하여 인스턴스가 유휴 상태인지 활용률이 낮은지를 식별합니다.
  - 안정적인 워크로드의 경우 정기적으로 AWS Compute Optimizer와 같은 AWS 적정 크기 조정 도구를 확인하여 최적화 기회를 식별하고 인스턴스를 적정 크기로 조정합니다. 추가 예제 및 권장 사항은 다음 실습을 참조하세요.
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
    - Well-Architected Lab - Optimize Hardware Patterns and Observice Sustainability KPIs

리소스

관련 문서:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- AWS Graviton
- Amazon EC2 DL1
- Amazon EC2 용량 예약 플릿
- Amazon EC2 스팟 플릿
- 함수: Lambda 함수 구성
- Amazon EC2 플릿의 속성 기반 인스턴스 유형 선택
- Building Sustainable, Efficient, and Cost-Optimized Applications on AWS
- How the Contino Sustainability Dashboard Helps Customers Optimize Their Carbon Footprint

관련 비디오:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 = What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

관련 예제:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS

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 작업을 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 테닛 컨트롤 플레인으로 인해 많은 사용자에게 서비스의 환경 영향을 분산시킵니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

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 ElastiCache는 관리형 데이터베이스 서비스를 제공합니다. Amazon Athena, Amazon EMR, Amazon OpenSearch Service에서는 관리형 분석 서비스를 제공합니다.

구현 단계

1. 워크로드 인벤토리 작성: 서비스 및 구성 요소에 대한 워크로드 인벤토리를 작성합니다.
2. 후보 식별: 관리형 서비스로 대체할 수 있는 구성 요소를 평가하고 식별합니다. 관리형 서비스 사용을 고려할 수 있는 몇 가지 예는 다음과 같습니다.

작업	AWS에서 사용하는 기능
데이터베이스 호스팅	Amazon Elastic Compute Cloud(Amazon EC2)에서 자체 Amazon RDS 인스턴스를 유지 관리하는 대신, 관리형 Amazon Relational Database Service(RDS)를 사용합니다.
컨테이너 워크로드 호스팅	자체 컨테이너 인프라를 구현하는 대신 AWS Fargate를 사용합니다.

작업	AWS에서 사용하는 기능
웹 앱 호스팅	정적 웹 사이트 및 서버 측 렌더링 웹 앱을 위한 완전관리형 CI/CD 및 호스팅 서비스로 AWS Amplify Hosting을 사용합니다.

3. 마이그레이션 계획 수립: 종속성을 식별하고 마이그레이션 계획을 수립합니다. 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. 테스트 수행: 관리형 서비스로 마이그레이션하기 전에 서비스를 테스트합니다.
5. 자체 호스팅 서비스 교체: 마이그레이션 계획을 사용하여 자체 호스팅 서비스를 관리형 서비스로 교체합니다.
6. 모니터링 및 조정: 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 비디오:

- AWS re:Invent 2021 - Cloud operations at scale with AWS Managed Services
- AWS re:Invent 2023 - Best practices for operating on AWS

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 않을 경우 노출되는 위험 수준: 중간

구현 지침

높은 처리 용량이 필요한 경우, 그래픽 처리 디바이스(GPU) 및 Field-Programmable Gate Array(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

- 컴퓨팅 가속기 탐색: 요구 사항을 해결할 수 있는 가속 컴퓨팅 인스턴스를 식별합니다.
- 목적별 하드웨어 사용: 기계 학습 워크로드의 경우 AWS Trainium, AWS Inferentia 및 Amazon EC2 DL1과 같이 워크로드에 적합한 목적별 하드웨어의 장점을 활용합니다. AWS Inferentia 인스턴스(예: Inf2 인스턴스)는 동급 Amazon EC2 인스턴스에 비해 최대 50% 더 우수한 와트당 성능을 제공합니다.
- 사용량 지표 모니터링: 가속 컴퓨팅 인스턴스의 사용량 지표를 수집합니다. 예를 들어, Amazon CloudWatch의 NVIDIA GPU 지표 수집에서와 같이 CloudWatch 에이전트를 사용하여 GPU에 대한 `utilization_gpu` 및 `utilization_memory`와 같은 지표를 수집할 수 있습니다.
- 적정 크기로 조정: 코드, 네트워크 운영 및 하드웨어 액셀러레이터 설정을 최적화하여 기본 하드웨어가 반드시 제대로 활용되도록 해야 합니다.
  - GPU 설정 최적화
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- 최신 상태 유지: 최신 고성능 라이브러리 및 GPU 드라이버를 사용합니다.

- 불필요한 인스턴스 해제: 자동화를 사용하여 사용하지 않는 GPU 인스턴스 사용을 해제합니다.

리소스

관련 문서:

- 가속 컴퓨팅
- Let's Architect! Architecting with custom chips and accelerators
- 워크로드에 적합한 EC2 인스턴스 유형을 선택하려면 어떻게 해야 하나요?
- Amazon EC2 VT1 Instances
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

관련 비디오:

- AWS re:Invent 2021 - How to select Amazon EC2 GPU instances for deep learning
- AWS Online Tech Talks - Deploying Cost-Effective Deep Learning Inference
- AWS re:Invent 2023 - Cutting-edge AI with AWS and NVIDIA
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

프로세스 및 문화

질문

- SUS 6 조직의 프로세스가 지속 가능성 목표를 어떻게 지원하고 있나요?

SUS 6 조직의 프로세스가 지속 가능성 목표를 어떻게 지원하고 있나요?

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

모범 사례

- SUS06-BP01 지속 가능성 목표 소통 및 전달

- SUS06-BP02 지속 가능성 개선을 신속하게 도입할 수 있는 방법 채택
- SUS06-BP03 워크로드를 최신 상태로 유지
- SUS06-BP04 구축 환경의 사용률 제고
- SUS06-BP05 테스트에 관리형 Device Farm 사용

SUS06-BP01 지속 가능성 목표 소통 및 전달

기술은 지속 가능성의 핵심 조력자입니다. IT 팀은 조직의 지속 가능성 목표에 대한 의미 있는 변화를 주도하는 데 중요한 역할을 합니다. 이러한 팀은 회사의 지속 가능성 목표를 명확하게 이해하고 운영 전반에 걸쳐 이러한 우선순위를 소통하고 전달하기 위해 노력해야 합니다.

일반적인 안티 패턴:

- 조직의 지속 가능성 목표가 무엇인지, 지속 가능성 목표가 팀에 어떻게 적용되는지를 모릅니다.
- 클라우드 워크로드가 환경에 미치는 영향에 대한 인식과 교육이 부족합니다.
- 우선순위를 지정할 구체적인 영역에 대해 잘 모릅니다.
- 지속 가능성 이니셔티브에 직원과 고객을 참여시키지 않습니다.

이 모범 사례 확립의 이점: 인프라 및 시스템 최적화부터 혁신적인 기술 사용에 이르기까지 IT 팀은 조직의 탄소 배출량을 줄이고 리소스 소비를 최소화할 수 있습니다. 지속 가능성 목표의 커뮤니케이션은 IT 팀이 변화하는 지속 가능성 문제를 지속적으로 개선하고 이에 적응할 수 있는 역량을 제공할 수 있습니다. 또한 이러한 지속 가능한 최적화는 비용 절감으로 이어지기도 하므로 비즈니스 사례가 강화됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 가이드

IT 팀의 주요 지속 가능성 목표는 시스템과 솔루션을 최적화하여 리소스 효율성을 높이고 조직의 탄소 발자국과 전반적인 환경 영향을 최소화하는 것입니다. 교육 프로그램 및 운영 대시보드와 같은 공동의 서비스 및 이니셔티브는 조직이 IT 운영을 최적화하고 탄소 발자국을 크게 줄이는 데 도움이 되는 솔루션을 구축하도록 지원할 수 있습니다. 클라우드는 물리적 인프라 및 에너지 조달 책임을 클라우드 제공업체의 공동 책임으로 이전할 뿐만 아니라 클라우드 기반 서비스의 리소스 효율성을 지속적으로 최적화할 수 있는 기회를 제공합니다.

팀이 클라우드의 고유한 효율성과 공동 책임 모델을 사용하면 조직의 환경 영향을 의미 있게 줄일 수 있습니다. 따라서 조직의 전반적인 지속 가능성 목표에 기여하고 보다 지속 가능한 미래를 향한 여정에서 전략적 파트너로서 이러한 팀의 가치를 입증할 수 있습니다.

구현 단계

- 목표 정의: IT 프로그램에 대해 잘 정의된 목표를 수립합니다. 여기에는 IT, 지속 가능성 및 재무와 같은 다양한 부서의 책임 있는 이해관계자로부터 의견을 얻는 것이 포함됩니다. 이러한 팀은 탄소 감축 및 리소스 최적화와 같은 영역을 포함하여 조직의 지속 가능성 목표에 부합하는 측정 가능한 목표를 정의해야 합니다.
- 비즈니스의 탄소 회계 경계 이해: 온실가스(GHG) 프로토콜과 같은 탄소 회계 방법이 클라우드의 워크로드와 어떤 관련이 있는지 이해합니다(자세한 내용은 클라우드 지속 가능성 참조).
- 탄소 회계를 위한 클라우드 솔루션 사용: AWS의 탄소 회계 솔루션과 같은 클라우드 솔루션을 사용하여 운영, 포트폴리오 및 가치 체인 전반에서 GHG 배출량에 대한 범위 1, 2 및 3을 추적할 수 있습니다. 이러한 솔루션을 통해 조직은 GHG 배출 데이터 수집을 간소화하고, 보고를 간소화하고, 인사이트를 도출하여 기후 전략에 반영할 수 있습니다.
- IT 포트폴리오의 탄소 발자국 모니터링: IT 시스템의 탄소 배출량을 추적하고 보고합니다. AWS Customer Carbon Footprint Tool을 사용하여 AWS 사용량에서 생성된 탄소 배출량을 추적, 측정, 검토 및 예측할 수 있습니다.
- 프록시 지표를 통해 리소스 사용량에 대해 팀과 소통: 프록시 지표를 통해 리소스 사용량을 추적하고 보고합니다. 클라우드의 온디맨드 요금 모델에서 리소스 사용량은 일반적으로 이해 가능한 지표인 비용과 관련이 있습니다. 최소한 비용을 프록시 지표로 사용하여 각 팀의 리소스 사용량 및 개선 사항에 대해 소통합니다.
  - Cost Explorer에서 시간별 세부 내역 활성화 및 비용 및 사용 보고서(CUR) 생성: CUR은 모든 AWS 서비스에 대해 일별 또는 시간별 사용 세부 내역, 요금, 비용 및 사용 속성을 제공합니다. Cloud Intelligence Dashboards 및 Sustainability Proxy Metrics Dashboard를 비용 및 사용량 기반 데이터의 처리와 시각화를 위한 출발점으로 사용합니다. 자세한 내용은 다음을 참조하세요.
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline
- 지속적 최적화 및 평가: 개선 프로세스를 사용하여 효율성 및 지속 가능성을 위한 클라우드 워크로드를 포함하여 IT 시스템을 지속적으로 최적화합니다. 최적화 전략 구현 전후 탄소 발자국을 모니터링합니다. 탄소 발자국 감소를 기준으로 효과를 평가합니다.
- 지속 가능성 문화 조성: 훈련 프로그램(예: AWS Skill Builder)을 사용하여 직원에게 지속 가능성에 대해 교육합니다. 직원을 지속 가능성 이니셔티브에 참여시킵니다. 직원의 성공 사례를 공유하고 축하합니다. 인센티브를 사용하여 지속 가능성 목표를 달성하는 경우 직원에게 보상합니다.

리소스

관련 문서:

- Understanding your carbon emission estimations

관련 비디오:

- AWS re:Invent 2,023 - Accelerate data-driven circular economy initiatives with AWS
- AWS re:Invent 2,023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

관련 예제:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports

관련 교육:

- Sustainability Transformation on AWS
- SimuLearn - Sustainability Reporting
- Decarbonization with AWS

SUS06-BP02 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 추적하는 프로세스를 구축하면 지속적으로 새로운 기능을 채택하고, 문제를 제거하며, 워크로드 효율성을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

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

- 조직의 지속 가능성 목표 이해 및 전달: 탄소 배출 감소 또는 수자원 관리와 같은 조직의 지속 가능성 목표를 이해합니다. 이러한 목표를 클라우드 워크로드의 지속 가능성 요구 사항으로 작성합니다. 이러한 요구 사항을 주요 이해관계자에게 전달합니다.
- 백로그에 지속 가능성 요구 사항 추가: 개발 백로그에 지속 가능성 개선을 위한 요구 사항을 추가합니다.
- 반복 및 개선: 반복 개선 프로세스를 사용하여 이러한 개선을 식별 및 평가하고, 우선순위를 지정하며, 테스트 및 배포합니다.
- 최소 기능 제품(MVP)을 사용하여 테스트: 테스트의 비용과 환경에 미치는 영향을 줄이기 위해 최소 기능 구성 요소를 사용하여 잠재적인 개선 사항을 개발하고 테스트합니다.
- 프로세스 간소화: 개발 프로세스를 지속적으로 개선하고 간소화합니다. 예를 들어, 지속적 통합 및 지속적 전달(CI/CD) 파이프라인을 통해 소프트웨어 전송 프로세스를 자동화함으로써 잠재적인 개선 사항을 테스트하고 배포하여 작업량을 줄이고 수동 프로세스로 인한 오류를 제한합니다.
- 교육 및 인식: 팀원에게 지속 가능성에 대해 알리고 이들의 활동이 조직의 지속 가능성 목표에 미치는 영향을 교육할 수 있는 교육 프로그램을 운영합니다.
- 평가 및 조정: 개선의 영향을 지속적으로 평가하고 필요에 따라 조정합니다.

리소스

관련 문서:

- AWS가 지원하는 지속 가능성 솔루션

관련 비디오:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure
- AWS re:Invent 2,023 - What's new with AWS observability and operations

SUS06-BP03 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 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

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

- 프로세스 정의: 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 영향을 줄입니다.
  - 성능 효율성을 높입니다.
  - 계획된 개선 작업의 장애 요인을 제거합니다.

  - 지속 가능성에 미치는 영향을 측정 및 관리할 수 있는 능력을 증진합니다.
- 인벤토리 작업 수행: 워크로드 소프트웨어 및 아키텍처를 조사하여 업데이트하는 데 필요한 구성 요소를 식별합니다.
  - AWS Systems Manager Inventory를 사용하여 Amazon EC2 인스턴스에서 운영 체제(OS), 애플리케이션, 인스턴스 메타데이터를 수집하고 소프트웨어를 실행 중인 인스턴스, 소프트웨어 정책에 필요한 구성, 업데이트해야 할 인스턴스를 신속하게 파악할 수 있습니다.
- 업데이트 절차 학습: 워크로드 구성 요소의 업데이트 방법을 파악합니다.

워크로드 구성 요소	업데이트 방법
머신 이미지	EC2 Image Builder를 사용하여 Linux 또는 Windows Server 이미지용 Amazon Machine Image(AMI)에 대한 업데이트를 관리합니다.
컨테이너 이미지	Amazon Elastic Container Registry(Amazon ECR)를 기존 파이프라인과 함께 사용하여 Amazon Elastic Container Registry(Amazon ECS) 이미지를 관리합니다.
AWS Lambda	AWS Lambda에는 버전 관리 기능이 있습니다.

- 자동화 사용: 업데이트를 자동화하여 새 기능 배포에 필요한 작업 수준을 줄이고 수동 프로세스로 인한 오류를 제한합니다.
  - CI/CD를 사용하여 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 비디오:

- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance
- All Things Patch: AWS Systems Manager

SUS06-BP04 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: 구축 환경의 활용도를 높이면 클라우드 워크로드의 전반적인 효율성을 개선하는 동시에 빌더가 효율적으로 개발, 테스트, 구축할 수 있도록 리소스를 할당할 수 있습니다.

이 모범 사례가 확립되지 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 인프라 사용: 코드형 인프라를 사용하여 구축 환경을 프로비저닝합니다.
- 자동화 사용: 자동화를 사용하여 개발 및 테스트 환경의 수명 주기를 관리하고 구축 리소스의 효율성을 극대화합니다.
- 활용도 극대화: 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.
- Bastion Host 플릿을 배포하는 대신, 네이티브 클라우드 서비스를 도입하여 보안 인스턴스 쉘에 액세스합니다.
- 구축 작업에 따라 구축 리소스를 자동으로 조정합니다.

리소스

관련 문서:

- AWS Systems Manager Session Manager
- Amazon EC2 성능 버스트 가능 인스턴스
- AWS CloudFormation란 무엇입니까?
- AWS CodeBuild란 무엇입니까?
- Instance Scheduler on AWS

관련 비디오:

- AWS re:Invent 2023 - Continuous integration and delivery for AWS

SUS06-BP05 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: 클라우드 지원 애플리케이션을 테스트하기 위해 관리형 Device Farm을 사용하면 다음과 같은 여러 가지 이점이 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 사례가 확립되지 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 정의: 테스트 요구 사항 및 계획(예: 테스트 유형, 운영 체제 및 테스트 일정)을 정의합니다.
  - Amazon CloudWatch RUM을 사용하여 클라이언트 측 데이터를 수집 및 분석하고 테스트 계획을 수립할 수 있습니다.
- 관리형 Device Farm 선택: 테스트 요구 사항을 지원할 수 있는 관리형 Device Farm을 선택합니다. 예를 들어 AWS Device Farm을 사용하여 변경 사항이 대표적인 하드웨어 세트에 미치는 영향을 테스트하고 파악할 수 있습니다.
- 자동화 사용: 지속적 통합 및 지속적 전달(CI/CD)을 사용하여 테스트를 예약하고 실행합니다.
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services
- 검토 및 조정: 테스트 결과를 지속적으로 검토하고 필요한 개선을 수행합니다.

리소스

관련 문서:

- AWS Device Farm device list
- CloudWatch RUM 대시보드 보기

관련 비디오:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM

관련 예제:

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS
- Appium Web tests for AWS Device Farm

고지 사항

고객은 본 문서의 정보를 독립적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공의 목적으로만 제공되고, (b) 사전 통지 없이 변경될 수 있는 현재 AWS 제품 및 관행을 나타내고, (c) AWS 및 그 계열사, 공급업체 또는 라이선스 제공자로부터 어떠한 약속이나 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 '있는 그대로' 제공됩니다. 고객에 대한 AWS의 책임 및 채무는 AWS 계약에 준거합니다. 본 문서는 AWS와 고객 간의 어떠한 계약도 구성하지 않으며 이를 변경하지도 않습니다.

Copyright © 2,024 Amazon Web Services, Inc. or its affiliates.

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하십시오.