AWS Well-Architected 프레임워크

2019년 7월

이 문서에서는 사용자가 스스로 클라우드 기반 아키텍처를 검토 및 개선하고 설계에 대한 의사결정이 비즈니스에 미치는 영향을 더 확실히 파악할 수 있도록 AWS Well-Architected 프레임워크에 대해 설명합니다. 또한 Well-Architected 프레임워크의 부문으로 정의되는 다섯 가지 개념의 일반적인 설계 원칙과 구체적인 모범 사례 및 지침에 대해서도 설명합니다.

고지 사항

고객은 본 문서에 포함된 정보를 독립적으로 평가할 책임이 있습니다. 본 문서는 (a)정보 제공만을 위한 것이며, (b)현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c)AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 "있는 그대로" 제공됩니다. 고객에 대한 AWS의 책임 및 의무는 AWS 계약에 의해 관리되며 본 문서는 AWS와 고객 사이의 어떠한 계약에도 속하지 않으며 계약을 변경하지도 않습니다.

소개

AWS Well-Architected 프레임워크를 이용하면 AWS에서 시스템을 구축하면서 내리게 되는 결정의 장점과 단점을 이해할 수 있습니다. 이 프레임워크를 통해 클라우드상의 안정적이고 안전하며 효율적이고 경제적인 시스템을 설계하고 운영하기 위한 설계 모범 사례를 알아볼 수 있습니다. 모범 사례에 대해 아키텍처를 일관적으로 측정하고 개선할 영역을 식별할 수 있는 방법을 제공합니다. 아키텍처 검토 프로세스는 아키텍처 결정에 대한 구조적인 대화이며 감사 메커니즘이 아닙니다. Well-Architected 시스템을 마련하면 비즈니스의 성공 가능성이 대폭 높아진다고 생각합니다.

AWS 솔루션스 아키텍트들은 광범위한 업종 및 사용 사례에 걸쳐 오랫동안 솔루션을 설계한 경험이 있습니다. AWS는 수천여 고객의 AWS 아키텍처를 설계하고 검토해 왔습니다. 그리고 이러한 경험을 바탕으로 클라우드 시스템 설계의 모범 사례와 핵심 전략을 밝혀냈습니다.

AWS Well-Architected 프레임워크 설명서에는 특정 아키텍처가 클라우드 모범 사례와 잘 맞는지를 파악하기 위한 근본적인 질문이 다수 수록되어 있습니다. 이 프레임워크를 이용하면 클라우드 기반의 현대적 시스템에 대해 고객이 기대하는 품질 기준에 따라 일정한 방식으로 시스템을 평가하고 그러한 품질을 달성하기 위해 필요한 수정 조치를 확인할 수 있습니다. AWS가 진화를 거듭하고 Amazon이 고객과 협력하면서 점점 더 많은 지식을 얻게 됨에 따라 앞으로도 Well-Architected 개념을 더욱 정교하게 가다듬고자 합니다.

이 프레임워크는 CTO(최고 기술 책임자), 아키텍트, 개발자, 운영팀 구성원 등 기술 업무 담당자를 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 홈 페이지를 참조하십시오.

AWS에서는 워크로드 검토를 위한 무료 서비스도 제공됩니다. AWS Well-Architected Tool(AWS WA Tool)은 AWS Well-Architected 프레임워크를 사용하여 아키텍처를 검토 및 측정하는 일관된 프로세스를 제공하는 클라우드 서비스입니다. AWS WA Tool은 워크로드를 더 안정적이고 안전하고 효율적이고 비용 효과적으로 만드는 권장 사항을 제공합니다.

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

정의

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

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

AWS Well-Architected 프레임워크는 운영 우수성, 보안, 안정성, 성능 효율성, 비용 최적화라는 다섯 가지 기반을 토대로 합니다.

표 1. AWS Well-Architected 프레임워크 부문

이름	설명
운영 우수성	시스템을 운영하고 모니터링하여 비즈니스 가치를 제공하고 지원 프로세스와 절차를 지속적으로 개선하는 능력입니다
보안	위험 평가 및 완화 전략을 통해 정보, 시스템 및 자산을 보호하는 동시에 비즈니스 가치를 제공하는 능력입니다
안정성	인프라나 서비스의 시스템 장애를 복구하고, 컴퓨팅 리소스를 동적으로 확보하여 수요에 대응하거나, 구성 오류나 일시적 네트워크 문제와 같은 장애를 완화하는 시스템의 성능입니다
성능 효율성	컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력입니다
비용 최적화	시스템을 실행하여 최저 가격으로 비즈니스 가치를 제공할 수 있는 기능입니다

AWS Well-Architected 프레임워크에서 사용되는 용어는 다음과 같습니다.

- 구성 요소는 요구 사항을 충족하는 데 집합적으로 사용되는 코드, 구성 및 AWS 리소스입니다. 구성 요소는 대개 기술 소유권 단위이며 각기 분리되어 있습니다.
- 워크로드는 비즈니스 가치를 제공하는 일련의 구성 요소를 가리킵니다. 워크로드는 일반적으로 비즈니스 및 기술 책임자가 언급하는 수준의 디테일에 해당합니다.
- 이정표는 설계, 테스트, 출시 및 프로덕션으로 이어지는 제품 수명주기 전반에 걸쳐 아키텍처가 개선되는 과정에서 발생하는 주요 변화 시점을 표시합니다.
- 아키텍처는 워크로드에서 구성 요소가 연동되는 방식입니다. 아키텍처 다이어그램에는 구성 요소의 통신 및 상호 작용 방식이 주로 표시되는 경우가 많습니다.
- 조직 내에서 기술 포트폴리오는 기업을 운영하는 데 필요한 워크로드 모음입니다.

워크로드를 설계할 때는 업무 상황에 따라 이러한 핵심 요소를 절충해야 합니다. 이러한 비즈니스 의사결정이 엔지니어링 우선 순위에 영향을 미칠 수 있습니다. 개발 환경에서 안정성을 희생하더라도 비용을 절감하도록 최적화할 수도 있고, 미션 크리티컬 솔루션의 경우 비용 증가를 감수하고 안정성을 기준으로 최적화할 수도 있습니다. 전자 상거래 솔루션의 경우 성능이 매출과 고객 구매 성향에 영향을 미칠 수 있습니다. 보안 및 운영 우수성은 일반적으로 다른 기반과 절충 관계에 있지 않습니다.

아키텍처

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

AWS는 해당 기능을 갖춘 중앙 집중형 팀을 보유하는 대신, 팀에 기능을 배포하는 것을 선호합니다. 예를 들어 팀이 내부 표준을 준수하는지 확인하는 것처럼 의사 결정 권한을 분산시키기로 결정할 때 위험이 있습니다. AWS는 두 가지 방법으로 이러한 위험을 완화합니다. 첫째로, 각 팀이 해당 역량을 갖추는 방법을 중점적으로 설명하는 사례[1] 을 도입하고, 팀이 충족해야 하는 표준에 대한 기준을 높일 수 있는 전문가[2]를 배치합니다. 둘째로, 해당 사례를 실제로 적용하기 위한 메커니즘을 구현합니다.[3] 표준 충족 여부를 확인하는 자동화된 검사를 수행함 이러한 분산된 접근 방식은 Amazon 리더십 원칙을 통해 뒷받침되며 고객을 통해 Working backward를 수행하는 문화를 수립합니다. 고객 중심의 팀은 고객의 요구 사항에 대응하여 제품을 개발합니다.

아키텍처의 경우 이는 모든 팀이 아키텍처를 생성하고 모범 사례를 따르는 기능을 갖추고 있음을 의미합니다. AWS는 새로운 팀이 이러한 기능을 확보하거나 기존 팀이 기준을 높이도록 돕기 위해, 설계를 검토하고 AWS 모범 사례가 무엇인지 이해하는 데 도움을 주는 책임 엔지니어의 가상 커뮤니티에 액세스할 수 있도록 권한을 활성화합니다. 주요 엔지니어링 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 내용을 기록하여 새 팀원을 위한 온보딩 자료의 일부로 사용할 수 있습니다.

AWS의 모범 사례는 수천 개의 시스템을 인터넷 규모로 운영한 경험에서 비롯된 것입니다. AWS는 데이터를 사용하여 모범 사례를 정의하는 것을 선호하지만, 책임 엔지니어와 같은 주제 전문가를 활용하여 설정하기도 합니다. 책임 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-Architected는 내부 검토 프로세스의 고객 중심 구현 결과이며, 주요 현장에서 활동하는 솔루션 아키텍처 및 내부 엔지니어링 팀에서 엔지니어링 사고를 체계화한 것입니다. Well-Architected는 이러한 결과를 활용할 수 있는 확장 가능한 메커니즘입니다.

아키텍처의 분산된 소유권을 가진 주요 엔지니어링 커뮤니티의 접근 방식에 따라, AWS는 고객 요구 사항을 중심으로 하는 Well-Architected 엔터프라이즈 아키텍처가 실현될 수 있다고 믿습니다. 모든 워크로드 전반에 걸쳐 Well-Architected 검토 방식을 수행하는 기술 리더(CTO 또는 개발 관리자)는 기술 포트폴리오의 위험을 더 효과적으로 이해할 수 있습니다. 이 접근 방식을 사용하

[1] 작업 수행 방식, 프로세스, 표준 및 용인된 규범

[2] "좋은 의도만으로는 안 됩니다. 무언가를 해내려면 좋은 메커니즘이 필요합니다" Jeff Bezos. 즉, 사람이 다할 수 있는 최선의 노력을 규칙이나 프로세스 준수 여부를 확인하는 메커니즘(자동화된 메커니즘인 경우가 많음)으로 대체하는 것입니다.

[3] Working backward는 아마존 혁신 프로세스의 기본 요소입니다. AWS는 고객 및 고객이 원하는 대상부터 시작하며, 자체 작업을 정의하고 안내합니다.

면 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 프레임워크는 다음과 같은 여러 가지 일반 설계 원칙을 확립하여 우수한 클라우드 설계를 촉진합니다.

- 필요 용량에 대한 추측 중단: 필요한 인프라 용량을 추측할 필요가 없습니다. 시스템을 배포하기 전에 용량을 결정했다가는 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 확장하거나 축소할 수 있기 때문입니다.
- 프로덕션 규모의 테스트 시스템: 클라우드에서는 온디맨드 방식으로 프로덕션 규모의 테스트 환경을 만들고, 테스트를 완료한 다음 해당 리소스를 폐기할 수 있습니다. 테스트 환경을 실행하는 동안에만 비용을 지불하면 되기 때문에 온프레미스 테스트 비용의 몇 분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.
- 자동화를 통해 더 간편해진 아키텍처 실험: 자동화를 통해 수작업 없이 저렴한 비용으로 시스템을 만들고 복제할 수 있습니다. 자동화 과정의 변경 사항을 추적하고, 그 효과를 감사하고, 필요하면 이전의 파라미터로 되돌릴 수 있습니다.
- 아키텍처의 지속적인 혁신: 아키텍처의 지속적인 혁신을 허용합니다. 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 달라지는데, 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 선택이 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.
- 실전 연습을 통한 개선: 프로덕션 환경에서 이벤트를 시뮬레이션하기 위한 실전 연습을 정기적으로 실시하여 아키텍처 및 프로세스가 어떻게 작동하는지 테스트할 수 있습니다. 그러면 어느 분야에서 개선이 필요한지 파악하고, 조직이 이벤트에 대처하는 경험을 쌓도록 도울 수 있습니다.

프레임워크의 5가지 부문

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

운영 우수성

운영 우수성 원칙에는 시스템을 운영하고 모니터링하여 비즈니스 가치를 제공하고 지원 프로세스와 절차를 지속적으로 개선하는 능력입니다 이(가) 포함됩니다.

운영 우수성 부문에서는 설계 원리 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 규범적 지침은 운영 우수성 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 운영 우수성에 대한 여섯개의 설계 원칙이 있습니다.

- 코드를 통한 운영: 애플리케이션 코드를 위해 사용하였던 엔지니어링 원칙을 클라우드에서 인프라를 포함한 환경에 적용할 수 있습니다. 클라우드에서는 전체 워크로드(애플리케이션, 인프라)를 코드로 정의하고 코드와 함께 업데이트할 수 있고 운영 절차를 코드로 구현하고 이벤트에 대응하여 이를 트리거하면 실행을 자동화할 수 있습니다. 운영을 코드로 수행하여 인적 오류를 제한하고 이벤트에 대한 지속적인 대응을 활성화합니다.
- 문서 어노테이션: 온프레미스 환경에서 문서는 수동으로 생성되고, 사람에 의해 사용되고, 변경에 맞춰 계속 동기화하기가 어렵습니다. 클라우드에서는 빌드 시마다 주석이 달린 문서 성생을 자동화할 수 있습니다(또는 직접 작성한 문서에 자동으로 주석을 달 수 있음). 주석이 달린 문서는 사람 뿐 아니라 시스템이 사용할 수 있습니다. 주석을 운영 코드에 대한 입력으로 사용할 수 있습니다.
- 작게 자주 발생하고 원복 가능한 변경 구성 : 요소를 정기적으로 업데이트할 수 있도록 워크로드를 설계하고, 실패할 경우 되돌릴 수 있는 작은 증분으로 변경 내용을 적용합니다(가능하면 고객에게 영향을 주지 않도록).
- 운영 절차를 수시 정제하기: 운영 절차를 사용할 때 개선할 여지가 있는지 확인합니다. 워크로드가 개선되면 절차도 적절하게 개선합니다. 정기적인 게임 데이를 설정하여 모든 절차가 효과가 있으며 팀이 이러한 절차에 친숙한지 여부를 검토하고 검증합니다.
- 실패 예측: “사전 분석(pre-mortem)” 연습을 수행하여 잠재적인 실패 소스를 식별하고 이를 해소하거나 완화할 수 있도록 합니다. 실패 시나리오를 테스트하고 그에 따른 영향을 이해했는지 여부를 검증합니다. 응답 절차를 테스트하여 효과가 있는지, 팀이 이 실행 단계에 친숙한지 확

인합니다. 정기적인 게임 데이를 준비하여 시뮬레이션된 이벤트에 대한 팀의 대응 및 워크로드를 테스트합니다.

- 모든 운영상 실패로부터 학습: 모든 운영상 이벤트 및 실패로부터 파악한 내용을 통해 개선합니다. 팀 전반 및 전체 조직에 걸쳐 파악한 내용을 공유합니다.

정의

클라우드에는 운영 우수성에 대한 셋개의 모범 사례 영역이 있습니다.

- 준비
- 운영
- 개선

운영 팀은 비즈니스 및 고객의 요구 사항을 이해하여야 하고 이를 통해 비즈니스 성과를 효과적으로, 그리고 효율적으로 지원할 수 있어야 합니다. 운영상 이벤트 대응을 위한 운영 절차를 생성하고, 비즈니스 요구 사항 지원을 위한 효율성을 검증합니다. 운영 단계에서는 원하는 비즈니스 성과 달성을 측정하는 데 사용되는 지표를 수집합니다. 비즈니스 컨텍스트, 비즈니스 우선순위, 고객 요구 사항 등 모든 요소는 계속해서 변화합니다. 시간에 따른 변화에 대응하여 개선 사항을 반영하고 성과를 통해 파악한 내용을 통합하도록 운영 단계를 설계하는 것이 중요합니다.

모범 사례

준비

운영 우수성을 달성하려면 효과적인 준비가 필요합니다. 비즈니스 성공은 비즈니스, 개발 및 운영 단계 전반에 걸쳐 공유된 이해와 목표를 통해 이루어집니다. 표준화는 워크로드 설계와 관리를 간소화하면서 운영 성공을 달성하게 해 줍니다. 애플리케이션, 플랫폼 및 인프라 구성 요소뿐만 아니라 고객 경험 및 행동 양식에 대한 인사이트를 확보하고 모니터링하는 메커니즘이 포함된 워크로드를 설계합니다.

워크로드나 변경이 프로덕션 단계와 운영 준비로 이동할 준비가 되어 있는지 검증할 메커니즘을 생성합니다. 운영 준비를 통해 워크로드가 정의된 표준을 충족하는지, 필요한 절차가 적절하게 런북 및 플레이북에서 캡처되었는지 여부를 체크리스트를 통해 검증합니다. 워크로드를 효과적으로 지원할 훈련을 받은 인력이 충분히 있는지 검증합니다. 이전하기 전에 운영상 이벤트 및 실패에 대한 응답을 테스트합니다. 실패 주입 및 게임 데이 이벤트를 통해 지원되는 환경에서 대응을 실습합니다.

AWS는 클라우드에서 코드기반 운영을 활성화하고 운영 절차를 안전하게 실험, 개발하고 실패를 실습하는 기능을 제공합니다. AWS CloudFormation을 사용하면 운영 제어 수준이 점점 증가하는 일관된 템플릿 형식의 샌드박스 개발, 테스트 및 생산 환경을 갖출 수 있습니다. AWS는 다양한 로그 수집 및 기능 모니터링을 통해 모든 계층의 워크로드에 대한 가시성을 활성화합니다. Amazon CloudWatch, AWS CloudTrail 및 VPC 흐름 로그를 사용하여 리소스, 애플리케이션 프

로그래밍 인터페이스(API) 및 네트워크 흐름 로그 사용 데이터를 수집할 수 있습니다. 수집된 플러그인 또는 CloudWatch Logs 에이전트를 사용하여 운영 체제에 대한 정보를 CloudWatch로 집계합니다.

다음 질문은 운영 우수성에 대한 이러한 고려 사항을 중점적으로 다룹니다. (운영 우수성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.).

OPS 1: 우선 순위를 결정하는 방법

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

OPS 2: 워크로드 상태를 파악할 수 있도록 워크로드를 설계하는 방법

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

OPS 3: 결함을 줄이고 문제를 쉽게 해결하고 프로덕션 환경으로의 흐름을 개선하는 방법

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

OPS 4: 배포 위험을 완화하는 방법

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 5: 워크로드를 지원할 준비가 되었는지 확인하는 방법

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

워크로드에 대한 최소 아키텍처 표준을 구현합니다. 워크로드에 대한 이점 및 운영상 부담과 표준을 구현하기 위한 비용 사이에서 균형을 찾습니다. 지원되는 표준 수를 줄임으로써 수용할 수 있는 수준보다 낮은 표준이 오류로 인해 적용될 가능성을 낮출 수 있습니다. 운영 인력은 제한된 리소스인 경우가 많습니다.

운영 활동을 코드로 구현하여 운영 인력의 생산성을 최대화하고, 오류율을 최소화하고, 자동화된 응답을 사용할 수 있습니다. 클라우드의 탄력성을 활용하는 배포 실습을 도입하여 시스템을 사전 배포할 수 있도록 함으로써 보다 빠른 구현을 달성합니다.

운영

워크로드 운영의 성공은 비즈니스 및 고객 성과 달성에 따라 측정됩니다. 예상 결과를 정의하고, 성공을 측정하는 방법을 결정하고 이러한 계산에 사용될 워크로드 및 운영 지표를 식별하여 운영이 성공적인지 여부를 결정합니다. 운영 상태에는 워크로드 상태, 워크로드에 대한 운영 상태와 성공이 모두 포함됩니다(예: 배포 및 인시던트 응답). 식별되는 운영 개선 또는 저하 상태로 기준선(baseline)을 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, 파악되지 않은 다른 이벤트의 해결책을 지원하는 데는 플레이북을 사용합니다. 비즈니스 및 고객 영향을 기반으로 이벤트 응답의 우선순위를 지정합니다. 이벤트 응답에 대해 알람이 발생하는지, 연결된 실행 프로세스가 있는지 여부를 특별히 식별된 소유자와 함께 확인합니다. 이벤트를 해결하는 데 필요한 인력을 미리 정의하고 에스컬레이션 트리거를 포함하여 필요할 경우 영향(예: 기간, 규모 및 범위)을 기반으로 추가 인력의 참여를 유도합니다. 권한이 있는 개인을 식별하고 참여시켜 이전에 해결되지 않은 이벤트 대응에 대해 대응 과정이 비즈니스에 영향을 미쳤는지 확인합니다.

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

계획되지 않은 이벤트 및 계획된 이벤트의 예측하지 못한 영향의 근본 원인을 확인합니다. 이 정보는 이후 이벤트 발생을 완화하기 위해 절차를 업데이트하는 데 사용됩니다. 해당하는 경우 영향을 받은 커뮤니티에 근본 원인을 전달합니다.

AWS에서는 AWS의 기본 지표 및 워크로드로부터 수집된 지표에 대한 대시보드 보기를 생성할 수 있습니다. CloudWatch 또는 타사 애플리케이션을 활용하여 운영 활동의 비즈니스, 워크로드 및 운영 수준 보기를 표시하고 집계할 수 있습니다. AWS는 근본 원인 분석 및 해결 지원을 통해 워크로드 문제를 식별할 수 있는 AWS X-Ray, CloudWatch, CloudTrail, VPC 흐름 로그 등의 로그 기능을 통해 워크로드 인사이트를 제공합니다.

다음 질문은 운영 우수성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

OPS 6: 워크로드 상태를 파악하는 방법

워크로드 지표를 정의, 파악 및 분석하면 워크로드 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.

OPS 7: 운영 상태를 파악하는 방법

운영 지표를 정의, 파악 및 분석하면 운영 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.

OPS 8: 워크로드 및 운영 이벤트를 관리하는 방법

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

정기 운영 및 계획되지 않은 이벤트에 대한 응답이 자동화되어야 합니다. 배포, 릴리스 관리, 변경 사항, 롤백에 대한 수동 프로세스는 사용해서는 안 됩니다. 릴리스는 수행 빈도가 높지 않은 대량 배치가 아니어야 합니다. 변경 사항이 대량인 경우 롤백이 더 까다롭습니다. 롤백 계획 또는 장애 영향 완화가 실패할 경우 운영 연속성이 중단됩니다. 응답이 비즈니스 연속성 유지에 효과적이도록 지표가 비즈니스 요구 사항을 부합하도록 합니다. 수동 응답이 있는 1회성 분산 데이터는 계획되지 않은 이벤트 발생 시 더 큰 운영 중단을 유발합니다.

개선

운영 우수성을 유지하려면 운영 개선이 필요합니다. 연속적이고 증분적 개선을 이뤄내는 데에 주력하여 작업 주기를 조절합니다. 워크로드 및 운영 절차 모두를 포함하여 개선의 여지(예: 기능 요청, 문제 해결, 규정 준수 요구 사항)를 정기적으로 평가하고 우선순위를 조정합니다. 절차 내에 피

드백 루프를 포함시켜 개선할 영역을 빠르게 식별하고 운영 실행을 통해 학습한 내용을 파악합니다.

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

AWS 개발자 도구를 사용하면 AWS 및 타사의 다양한 소스 코드, 빌드, 테스트 및 개발 도구와 연동하여 연속적인 구축, 테스트 및 개발 활동을 구현할 수 있습니다. 배포 활동 결과는 배포 및 개발 모두에 대해 개선할 여지를 식별하는 데 사용할 수 있습니다. 운영 및 배포 활동의 데이터를 통합하는 지표 데이터에 대한 분석을 수행하여 비즈니스 및 고객 성과에 대한 해당 활동의 영향을 분석할 수 있습니다. 이 데이터는 개선할 여지 및 방법을 식별하기 위한 팀 교차 후행 분석에 활용할 수 있습니다.

다음 질문은 운영 우수성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

OPS 9: 운영을 개선하는 방법
시간과 리소스를 할애하여 점진적 개선을 지속적으로 수행하면 운영 효율성을 높일 수 있습니다.

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

주요 AWS 서비스

운영 우수성에 필수적인 AWS 서비스은(는) AWS CloudFormation, 모범 사례를 기반으로 템플릿을 생성하는 데 사용할 수 있습니다. AWS CloudFormation을 사용하면 순서에 따라 일관된 방식으로 개발 단계부터 생산 환경에 이르기까지 리소스를 프로비저닝할 수 있습니다. 이며, 다음 서비스 및 기능이 운영 우수성의 셋개 영역을 지원합니다.

- 준비: AWS Config 및 AWS Config 규칙은 워크로드에 대한 표준을 생성하고 생산 단계로 진입하기 전에 환경이 이러한 표준을 준수하는지 여부를 확인하는 데 사용될 수 있습니다.
- 운영: Amazon CloudWatch를 사용하면 워크로드 운영 상태를 모니터링할 수 있습니다.
- 개선: Amazon Elasticsearch Service(Amazon ES)를 활용하면 로그 데이터를 분석하여 실행 가능한 인사이트를 빠르고 안전하게 확보할 수 있습니다.

리소스

운영 우수성 관련 AWS 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

설명서

- DevOps and AWS

백서

- Operational Excellence Pillar

동영상

- DevOps at Amazon

보안

보안 원칙에는 위험 평가 및 완화 전략을 통해 정보, 시스템 및 자산을 보호하는 동시에 비즈니스 가치를 제공하는 능력입니다 이(가) 포함됩니다.

보안 부문에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 보안 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 보안에 대한 일곱개의 설계 원칙이 있습니다.

- 강력한 자격 증명 기반 구현: 권한을 최소화한 보안 주체를 구현하고 AWS 리소스와의 각 상호 작용에 대한 적절한 권한을 부여하여 업무를 분리합니다. 보안 주체 관리를 중앙 집중화하고 장기 자격 증명에 대한 의존도를 줄이거나 제거합니다.
- 추적 기능 활성화: 실시간으로 환경에 대한 작업 및 변경 사항을 모니터링하고 알림을 전송하며 감사합니다. 로그 및 측정치를 시스템에 통합하여 자동으로 대응하고 조치를 취합니다.
- 모든 계층에 보안 적용: 단일 외부 계층 보호에 중점을 두는 대신 다른 보안 제어를 사용하여 심층 방어 접근 방식을 적용합니다. 모든 계층(예: 엣지 네트워크, VPC, 서브넷, 로드 밸런서, 모든 인스턴스, 운영 체제 및 애플리케이션)에 적용합니다.
- 보안 모범 사례의 자동 적용: 자동화된 소프트웨어 기반의 보안 메커니즘은 더욱 빠르게 안전한 확장 능력을 향상 시켜주고 비용 효율적입니다. 버전 제어 템플릿에서 코드로 정의되고 관리되는 제어 기능의 구현을 비롯한 보안 아키텍처를 생성합니다.
- 전송 및 보관 중인 데이터 보호: 데이터를 민감도 수준으로 분류하고 적절한 경우 암호화, 토큰화 및 액세스 제어와 같은 메커니즘을 사용합니다.
- 사람들이 데이터에 쉽게 액세스할 수 없도록 유지: 데이터의 직접 액세스 또는 수동 처리의 필요성을 줄이거나 없애기 위한 메커니즘 및 도구를 생성합니다. 이를 통해 민감한 데이터를 처리할 때 손실 또는 수정 및 수동 작업으로 인한 오류 위험을 줄일 수 있습니다.
- 보안 이벤트 준비: 조직의 요구 사항에 부합하는 인시던트 관리 프로세스를 통해 사고에 대비합니다. 사고 대응 시뮬레이션을 실행하고 자동화된 도구를 사용하여 감지, 조사 및 복구 속도를 높입니다.

정의

클라우드에는 보안에 대한 다섯개의 모범 사례 영역이 있습니다.

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

시스템을 설계하기 전에 보안에 영향을 미치는 방법을 익혀야 합니다. 작업을 수행할 수 있는 대상 및 작업 내용을 제어하고 싶을 수 있습니다. 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 서비스를 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

자격 증명 및 액세스 관리

자격 증명 및 액세스 관리는 정보 보안 프로그램의 핵심 요소로, 허가 받고 인증된 사용자에 한해 허용되는 방식으로만 리소스에 액세스할 수 있도록 하는 것을 말합니다. 예를 들어 보안 주체(계정에서 작업을 수행하는 사용자, 그룹, 서비스 및 역할)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, 멀티 팩터 인증(MFA) 등 강력한 암호를 요구할 수 있는 기능도 있습니다. 기존의 디렉터리 서비스와 연동되도록 할 수도 있습니다. 시스템이 AWS에 액세스해야 하는 워크로드의 경우 IAM이 인스턴스 프로필, 자격 증명 연동, 임시 자격 증명을 통해 보안 액세스를 보장합니다.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다. (보안 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.).

SEC 1: 자격 증명 및 인증을 관리하는 방법
자격 증명 및 인증 메커니즘에는 워크로드에서 직접적 또는 간접적으로 액세스 권한을 부여하는 암호, 토큰 및 키가 포함됩니다. 우발적 사용 또는 악의적 사용의 위험을 줄일 수 있게 적절한 메커니즘을 사용하여 자격 증명을 보호해야 합니다.
SEC 2: 인적 액세스를 제어하는 방법
무단 액세스의 영향과 관련 위험을 줄이려면 정의된 비즈니스 요구 사항에 맞는 제어 기능을 구현하여 인적 액세스를 제어해야 합니다. 이러한 제어는 AWS 계정의 권한 있는 사용자/관리자와 애플리케이션 최종 사용자에게 모두 적용됩니다.
SEC 3: 프로그래밍 방식 액세스를 제어하는 방법
적절하게 정의/제한/분리된 액세스 권한을 통해 프로그래밍 방식 액세스 또는 자동화된 액세스를 제어하면 무단 액세스 위험을 줄일 수 있습니다. 프로그래밍 방식 액세스에는 워크로드 내부의 액세스 및 AWS 관련 리소스 액세스가 포함됩니다.

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

AWS는 Identity and access management를 사용하여 도울 수 있는 리소스를 제공합니다. 모범 사례를 배우려면 자격 증명 및 인증 관리, 인적 액세스 제어 및 프로그래밍 방식 액세스 제어에 대한 실습을 살펴보십시오.

탐지 제어

탐지 제어를 사용하여 잠재적 보안 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 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(Amazon S3)를 사용하여 액세스 요청을 기록할 수 있습니다.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 4: 보안 이벤트를 감지하고 조사하는 방법
이벤트는 로그와 지표에서 캡처하고 분석하는 방식으로 파악할 수 있습니다. 보안 이벤트 및 잠재적 위협에 대한 조치를 취하면 워크로드를 보호할 수 있습니다.
SEC 5: 새로운 보안 위협을 방어하는 방법
AWS 및 업계의 최신 모범 사례와 위협 관련 정보를 지속적으로 확인하면 새로운 위험을 파악할 수 있습니다. 그러면 워크로드를 보호하는 데 적합한 제어 기능을 확인하여 우선 순위를 지정하고 구현하는 보안 위협 모델을 생성할 수 있습니다.

Well-Architected 설계에서 로그 관리가 중요한 이유는 보안 또는 과학 수사에서부터 규제 또는 법적 요구 사항에 이르기까지 다양합니다. 잠재적 보안 인시던트를 식별하려면 로그를 분석 및 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 안정적으로 데이터를 처리할 수 있습니다.

인프라 보호

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

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(Amazon VPC)를 사용하여 안전하고 확장 가능한 프라이빗 환경을 만들고, 여기에서 게이트웨이, 라우팅 테이블, 퍼블릭 및 프라이빗 서브넷 같은 토폴로지를 정의해야 합니다.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 6: 네트워크 보호 방법
퍼블릭 및 프라이빗 네트워크에는 내/외부 네트워크 기반 위협으로부터 네트워크를 보호할 수 있는 다계층 방어가 필요합니다.
SEC 7: 컴퓨팅 리소스를 보호하는 방법
워크로드의 컴퓨팅 리소스는 다계층 방어를 통해 내/외부 위협으로부터 보호해야 합니다. 컴퓨팅 리소스에는 EC2 인스턴스, 컨테이너, AWS Lambda 함수, 데이터베이스 서비스, IoT 디바이스 등이 포함됩니다.

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

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

데이터 보호

시스템을 설계하려면 먼저 보안과 관련된 기본적인 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 기본 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 기간 동안 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 리전 간 데이터 이동을 하지 않습니다. 특정 리전에 저장된 콘텐츠는 사용자가 명시적으로 기능을 활성화하거나 그 기능을 제공하는 서비스를 이용하지 않는 한 해당 리전을 벗어나지 않습니다.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 8: 데이터 분류 방법
분류는 중요도 수준을 기반으로 데이터를 분류하는 방법을 제공하여 적절한 보호 및 보존 제어를 결정하도록 합니다.
SEC 9: 저장된 데이터 보호 방법
무단 액세스 또는 손실 위험을 줄이기 위해 암호화를 비롯한 제어를 구현하고 요구 사항을 정의하여 저장된 데이터를 보호합니다.
SEC 10: 전송 중인 데이터 보호 방법
무단 액세스 또는 노출 위험을 줄이기 위해 암호화를 비롯한 제어를 구현하고 요구 사항을 정의하여 전송 중인 데이터를 보호합니다.

AWS는 저장된 데이터 및 전송 중인 데이터를 암호화할 수 있는 여러 가지 수단을 제공합니다. 데이터를 암호화하기 쉽도록 AWS 서비스에 각종 기능을 내장했습니다. 예를 들어, Amazon S3에 대해 SSE(서버 측 암호화)를 구현하여 데이터를 암호화된 형태로 저장하기 쉽게 만들었습니다. 또한 흔히 SSL termination 이라고 부르는 전체 HTTPS 암호화 및 복호화 프로세스를 Elastic Load Balancing을 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 실전 연습을 통해 인시던트 대응을 정기적으로 연습한다면 아키텍처가 적기에 조사 및 복구를 수용할 수 있게 할 수 있습니다.

AWS에서는 다음과 같은 사례를 통해 효과적인 인시던트 대응을 지원합니다.

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

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 11: 인시던트에 대응하는 방법
조직의 업무 중단 가능성을 최소화할 수 있도록 보안 인시던트를 제때 조사하고 대응하려면 철저한 준비가 필요합니다.

InfoSec 팀에게 신속하게 액세스를 부여할 수 있는 절차를 마련하고 인스턴스 격리와 과학 수사를 위해 데이터 및 상태 캡처를 자동화합니다.

주요 AWS 서비스

보안에 필수적인 AWS 서비스은(는) AWS Identity and Access Management(IAM), 사용자에 대해 AWS 서비스와 리소스에 대한 액세스 권한을 안전하게 제어할 수 있게 해 줍니다.이며, 다음 서비스 및 기능이 보안의 다섯개 영역을 지원합니다.

- 자격 증명 및 액세스 관리: IAM은 사용자의 AWS 서비스와 리소스에 대한 액세스 권한을 안전하게 제어할 수 있게 해 줍니다. MFA는 사용자 액세스에 대해 추가 보호 계층을 추가합니다. AWS Organizations는 여러 AWS 계정에 대한 정책을 중앙 집중식으로 관리하고 시행할 수 있게 해 줍니다.
- 탐지 제어: AWS CloudTrail은 AWS API 호출을 기록하고 AWS Config는 AWS 리소스 및 구성에 대한 자세한 인벤토리를 제공합니다. Amazon GuardDuty는 악성 또는 인증되지 않은

동작을 지속적으로 모니터링하는 관리형 위협 탐지 서비스입니다. Amazon CloudWatch는 CloudWatch 이벤트를 트리거하여 보안 응답을 자동화할 수 있는 AWS 리소스 모니터링 서비스입니다.

- 인프라 보호: Amazon Virtual Private Cloud(Amazon VPC)에서는 사용자가 정의한 가상 네트워크로 AWS 리소스를 시작할 수 있습니다. Amazon CloudFront는 DDoS 완화를 위해 AWS Shield와 통합되어 뷰어에게 데이터, 동영상, 애플리케이션 및 API를 안전하게 전달하는 글로벌 콘텐츠 전송 네트워크입니다. AWS WAF는 Amazon CloudFront 또는 Application Load Balancer에 배포되는 웹 애플리케이션 방화벽으로, 일반적인 웹 도용에서 웹 애플리케이션을 보호합니다.
- 데이터 보호: ELB, Amazon Elastic Block Store(Amazon EBS), Amazon S3 및 Amazon Relational Database Service(Amazon RDS)등의 서비스에는 암호화 기능이 포함되어 있으므로 전송 및 저장 상태의 데이터를 보호해 줍니다. Amazon Macie는 민감한 데이터를 자동으로 발견, 분류 및 보호하며 AWS Key Management Service(AWS KMS)는 암호화에 사용되는 키를 쉽게 만들고 제어할 수 있게 합니다.
- 인시던트 대응: IAM을 사용하여 인시던트 대응 팀 및 대응 도구에 적절한 권한을 부여해야 합니다. Amazon CloudFormation을 사용하면 조사를 수행할 신뢰할 수 있는 환경 또는 안전한 공간을 만들 수 있습니다. Amazon CloudWatch Events를 사용하면 AWS Lambda를 포함한 자동 응답을 트리거하는 규칙을 만들 수 있습니다.

리소스

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

설명서

- AWS Cloud Security
- AWS Compliance
- AWS Security Blog

백서

- Security Pillar
- AWS Security Overview
- AWS Security Best Practices
- AWS Risk and Compliance

동영상

- AWS Security State of the Union
- Shared Responsibility Overview

안정성

안정성 원칙에는 인프라나 서비스의 시스템 장애를 복구하고, 컴퓨팅 리소스를 동적으로 확보하여 수요에 대응하거나, 구성 오류나 일시적 네트워크 문제와 같은 장애를 완화하는 시스템의 성능입니다 이(가) 포함됩니다.

안정성 부문에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 안정성 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 안정성에 대한 다섯개의 설계 원칙이 있습니다.

- 복구 절차 테스트: 온프레미스 환경에서는 특정한 시나리오에서 시스템이 작동한다는 것을 입증하기 위해 테스트를 수행하는 경우가 많으며, 일반적으로 복구 전략을 검증하기 위해 테스트하지는 않습니다. 클라우드에서는 시스템의 장애 과정을 테스트하고 복구 절차를 검증할 수 있습니다. 자동화를 이용하여 다양한 장애를 시뮬레이션하거나 예전에 장애로 이어졌던 시나리오를 재현할 수 있습니다. 이렇게 해서 드러난 장애 경로를 테스트하고 실제 장애 시나리오로 이어지기 전에 문제를 해결함으로써 테스트를 거치지 않은 구성 요소의 장애 위험을 줄일 수 있습니다.
- 장애 자동 복구: 시스템의 핵심 성능 지표(KPI)를 모니터링하다가 임계값을 넘어서면 자동화를 트리거할 수 있습니다. 이를 통해 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.
- 수평적 확장으로 시스템 전체 가용성 증대: 큰 리소스 하나를 작은 리소스 여러 개로 대체하여 한 가지 장애가 전체 시스템에 미치는 영향을 줄일 수 있습니다. 요청을 더 작은 리소스 여러 개로 분산시키면 공통의 장애 지점을 공유하지 않게 됩니다.
- 용량 추정 불필요: 시스템에 대한 수요가 해당 시스템의 용량을 넘어서는 리소스 포화 상태는 온프레미스 시스템에서 흔히 발생하는 장애의 원인입니다(서비스 거부 공격의 대상). 클라우드에서는 수요 및 시스템 사용량을 모니터링하고 리소스 추가 또는 제거를 자동화함으로써 프로비저닝 과다 또는 부족 현상 없이 최적의 수준으로 수요를 충족할 수 있습니다.
- 자동화 변경 사항 관리: 인프라에 대한 변경이 자동화를 사용하여 이루어져야 합니다. 관리해야 할 변경 사항은 자동화의 대상입니다.

정의

클라우드에는 안정성에 대한 셋개의 모범 사례 영역이 있습니다.

- 기반
- 변경 관리

- 장애 관리

시스템 안정성을 얻기 위해서는 잘 계획된 기반 위에 모니터링을 실시하고, 수요 또는 요구 변화를 처리하기 위한 메커니즘을 갖춰야 합니다. 또한 장애를 탐지하고 자동으로 해결될 수 있도록 시스템을 설계해야 합니다.

모범 사례

기반

시스템을 설계할 때는 먼저 안정성을 좌우하는 기반 요구 사항부터 갖춰야 합니다. 예를 들어, 데이터 센터의 네트워크 대역폭을 충분히 확보해야 합니다. 특정 프로젝트의 범위를 넘어선다는 이유로 이러한 요구 사항을 간과하는 경우도 있습니다. 이렇게 되면 안정적인 시스템을 제공하는 능력이 상당히 저하될 수 있습니다. 온프레미스 환경의 경우, 이러한 요구 사항에 따른 종속성으로 인해 리드 시간이 길어질 수 있으므로 결국 초기 계획에 이를 반영해야 합니다.

그러나 AWS에는 이러한 기반 요구 사항이 대부분 이미 통합되어 있거나 필요에 따라 적용할 수 있습니다. 클라우드는 기본적으로 한계가 없도록 설계되어 있으므로 충분한 네트워킹 및 컴퓨팅 파워에 대한 요구는 AWS의 책임입니다. 고객은 스토리지 디바이스의 크기 등 리소스 크기와 할당 비율을 필요에 따라 자유롭게 변경할 수 있습니다.

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다. (안정성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.).

REL 1: 서비스 한도를 관리하는 방법
기본 서비스 한도는 사용자가 필요 이상으로 많은 리소스를 실수로 프로비저닝하는 것을 방지합니다. 또한 서비스 악용으로부터 보호하기 위해 얼마나 자주 API 작업을 호출 할 수 있는지에 대한 한도가 있습니다. AWS Direct Connect를 사용하는 경우 각 연결에서 전송할 수 있는 데이터 양에 한도가 적용됩니다. AWS Marketplace 애플리케이션을 사용하는 경우에는 애플리케이션의 제한을 파악해야 합니다. 타사 웹 서비스 또는 SaaS(Software-as-a-Service)를 사용하는 경우에는 해당 서비스의 한도도 파악해야 합니다.
REL 2: 네트워크 토폴로지를 관리하는 방법
애플리케이션은 기존 데이터 센터 인프라, 공개적으로 액세스 가능한 퍼블릭 클라우드 인프라, 비공개 주소가 지정된 퍼블릭 클라우드 인프라 등 하나 이상의 환경에 있을 수 있습니다. 시스템 내/시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 주소 관리, 이름 확인과 같은 네트워크 고려 사항은 클라우드의 리소스를 활용하기 위한 기본 사항입니다.

AWS는 실수로 인한 리소스 과다 프로비저닝을 방지하기 위해 서비스 한도(팀에서 요청할 수 있는 각각의 리소스 수 상한선)를 설정하고 있습니다. 이러한 한도를 모니터링하고 비즈니스상 필요에 따라 변경하기 위한 거버넌스 및 프로세스를 마련해 두어야 합니다. 클라우드 도입 과정에서 기존 온프레미스 리소스와의 통합을 계획해야 할 수 있습니다(하이브리드 접근 방식). 하이브리드 모델에서는 시간을 두고 완벽한 클라우드 방식으로 점진적으로 이전할 수 있습니다. 따라서 AWS 리소스와 온프레미스 리소스가 네트워크 토폴로지에 따라 상호 작용하는 방식을 반드시 설계해 두어야 합니다.

변경 관리

변경 사항이 시스템에 미치는 영향을 알고 있으면 적극적인 계획 수립이 가능하며, 모니터링을 통해 용량 문제 또는 SLA 위반으로 이어질 수 있는 동향을 신속하게 파악할 수 있습니다. 기존 환경에서는 변경 제어 프로세스를 수작업으로 처리하는 경우가 많았고, 변경 담당자와 변경 시기를 효과적으로 관리하기 위해서는 반드시 감사 부서와 철저히 공조해야 했습니다.

AWS를 이용하면 시스템 동작을 모니터링하고 KPI 대응을 자동화할 수 있습니다. 예를 들어, 서버 시스템을 추가하면 사용자 수가 늘어납니다. 시스템 변경 권한을 가진 사용자를 관리하고 이러한 변경 기록을 감사할 수 있습니다.

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

REL 3: 시스템이 수요 변화에 어떻게 대응하고 있습니까?

확장 가능형 시스템에는 리소스를 자동으로 유연하게 추가하거나 제거할 수 있어 특정 시점의 요구에 근접하게 대응합니다.

REL 4: 리소스를 모니터링하는 방법

로그와 지표는 워크로드의 상태를 파악할 수 있는 유용한 도구입니다. 로그 및 지표를 모니터링하여 임계값을 초과하거나 중요한 이벤트가 발생하면 알림을 보내도록 워크로드를 구성할 수 있습니다. 낮은 성능 임계값을 초과하거나 장애가 발생할 때는 워크로드가 그에 대응하여 자동으로 자체 복구되거나 확장되도록 설계하는 것이 가장 좋습니다.

REL 5: 변경 사항을 어떻게 수행하십니까?

환경에 제어되지 않는 변경 사항이 생기면 그 영향을 예측하기가 어렵습니다. 워크로드와 운영 환경에서 확인된 소프트웨어를 실행하고 예측 가능한 방법으로 패치를 적용하거며 예측가능한 방식으로 교체할 수 있도록 하기 위하여 프로비저닝된 리소스와 워크로드에 대한 변경 사항은 제어되어야 합니다.

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

장애 관리

통상적인 수준의 복잡한 시스템에는 장애가 발생하기 마련입니다. 이러한 장애를 파악하는 방법과 대응 방법, 재발 방지 방법 등을 알아 둘 필요가 있습니다.

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

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

REL 6: 데이터는 어떻게 백업합니까?
데이터, 애플리케이션 및 운영 환경(애플리케이션이 구성된 운영 체제)을 백업하여 평균 복구 시간(MTTR) 및 목표 복구 시점(RPO) 요구 사항을 충족하십시오.
REL 7: 구성 요소 장애 시 시스템에서 대처하는 방법
워크로드에 명시적으로든 암시적으로든 고가용성과 낮은 평균 복구 시간(MTTR)에 대한 요구 사항이 있는 경우 복원성을 갖도록 워크로드를 설계하고 이러한 워크로드를 배포하여 중단에 대처하십시오.
REL 8: 복원성은 어떻게 테스트하고 있습니까?
워크로드의 복원성을 테스트하면 프로덕션 환경에서만 나타나는 잠재적인 버그를 찾는 데 도움이 됩니다. 이러한 테스트를 정기적으로 수행하십시오.
REL 9: 재해 복구를 계획하는 방법
데이터를 백업 방식으로 복원해야 하는 경우 재해 복구(DR)가 필수적입니다. 이 데이터의 목표, 리소스, 위치 및 기능은 RTO 및 RPO 목표와 일치하도록 정의하고 실행해야 합니다.

정기적으로 데이터를 백업하고 백업 파일을 테스트하여 논리적 오류와 물리적 오류를 모두 복구할 수 있는지 확인하십시오. 빈번한 시스템 자동 테스트를 통해 장애 원인을 파악하고 복구 방식을 살펴보는 것이 장애 관리의 열쇠입니다 정기 일정에 따라 이 테스트를 수행하고, 중요한 시스템 변경 이후에도 이 테스트가 트리거되는지 확인해야 합니다. 복구 시간 목표(RTO), 복구 시점 목표(RPO) 등의 KPI를 적극적으로 추적하여 장애 테스트 시나리오 등에서 시스템의 복원력을 평가합니다. KPI를 추적하면 단일 장애 지점을 파악 및 완화하는 데 도움이 됩니다. 목표는 시스템 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.

주요 AWS 서비스

안정성에 필수적인 AWS 서비스은(는) Amazon CloudWatch, 런타임 지표를 모니터링합니다.이며, 다음 서비스 및 기능이 안정성의 셋개 영역을 지원합니다.

- 기반: AWS IAM은 사용자의 AWS 서비스와 리소스에 대한 액세스 권한을 안전하게 제어할 수 있게 해 줍니다. Amazon VPC를 통해 AWS 클라우드의 격리된 프라이빗 영역을 프로비저닝하고, 가상 네트워크에서 AWS 리소스를 실행할 수 있습니다. AWS Trusted Advisor는 서비스 한도에 대한 가시성을 제공합니다. AWS Shield는 AWS에서 실행되는 웹 애플리케이션을 보호하는 디도스(DDoS) 보호 서비스입니다.
- 변경 관리: AWS CloudTrail은 계정에 대한 AWS API 호출을 기록하고 로그 파일을 감사할 수 있도록 사용자에게 전달합니다. AWS Config는 AWS 리소스 및 구성에 대한 자세한 목록 정보를 제공하고, 구성 변경 사항을 지속적으로 기록합니다. Amazon Auto Scaling은 배포된 워크로드에 대한 자동화된 수요 관리를 제공하는 서비스입니다. Amazon CloudWatch는 사용자 지정 지표를 비롯하여 지표에 대한 알림 기능을 제공합니다. 또한 Amazon CloudWatch에는 리소스의 로그 파일을 집계하는 데 사용할 수 있는 로깅 기능이 있습니다.

- 장애 관리: AWS CloudFormation은 AWS 리소스의 템플릿을 만들어 순서에 따라 예측 가능한 방식으로 프로비저닝할 수 있는 템플릿을 제공합니다. Amazon S3는 백업을 유지할 수 있는 내구성이 뛰어난 서비스를 제공합니다. Amazon Glacier는 내구성이 뛰어난 아카이브를 제공합니다. AWS KMS는 많은 AWS 서비스와 통합되는 안정적인 키 관리 시스템을 제공합니다.

리소스

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

설명서

- Service Limits
- Service Limits Reports Blog
- Amazon Virtual Private Cloud
- AWS Shield
- Amazon CloudWatch
- Amazon S3
- AWS KMS

백서

- Reliability Pillar
- Backup Archive and Restore Approach Using AWS
- Managing your AWS Infrastructure at Scale
- AWS Disaster Recovery
- AWS Amazon VPC Connectivity Options

동영상

- How do I manage my AWS service limits?
- Embracing Failure: Fault-Injection and Service Reliability

제품

- AWS Premium Support
- Trusted Advisor

성능 효율성

성능 효율성 원칙에는 컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력입니다 이(가) 포함됩니다.

성능 효율성 부문에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적인 가이드는 성능 효율성 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 성능 효율성에 대한 다섯개의 설계 원칙이 있습니다.

- 고급 기술의 대중화: 기술에 대한 지식과 복잡성을 클라우드 업체가 제공하는 서비스로 극복하면서, 구현하기 어려운 기술도 쉽게 사용할 수 있습니다. 새로운 기술을 호스팅하고 실행하는 방법을 IT 팀에서 직접 배우는 대신 서비스 방식으로 간편하게 이용하면 됩니다. 예를 들어 NoSQL 데이터베이스, 미디어 트랜스코딩, 기계 학습 등은 아직 기술 커뮤니티 전반에 고르게 확산되지 않은 전문 지식이 요구되는 기술입니다. 클라우드에서는 이러한 기술을 서비스 방식으로 제공하기 때문에 팀은 리소스 프로비저닝 및 관리가 아닌 제품 개발에 집중하면서 서비스를 활용할 수 있습니다.
- 몇 분 만에 전 리전으로 프로비저닝: 클릭 몇 번으로 세계 곳곳의 여러 리전에 시스템을 손쉽게 배포할 수 있습니다. 이를 통해 지연 시간을 줄이고 최소 비용으로 고객에게 더 나은 사용 환경을 제공할 수 있습니다.
- 서버리스 아키텍처 사용: 클라우드에서 서버리스 아키텍처를 이용하면 서버를 실행하고 유지하는 등의 전통적인 컴퓨팅 관리 업무를 할 필요가 없습니다. 예를 들면 스토리지 서비스에 정적 웹 사이트 역할을 맡기면 웹 서버가 필요 없고, 이벤트 서비스를 통해 코드를 호스팅할 수 있습니다. 이렇게 하면 서버 관리에 따르는 운영 부담이 줄어들 뿐 아니라, 이러한 관리형 서비스가 클라우드 규모에서 작동하므로 트랜잭션 비용을 낮출 수 있습니다.
- 테스트 횟수 증가: 자동화할 수 있는 가상 리소스를 활용하며 여러 가지 인스턴스, 스토리지 또는 구성에 대한 비교 테스트를 신속하게 수행할 수 있습니다.
- 기계적 동조: 보관 대상에 가장 적합한 기술 접근 방식을 사용합니다. 예를 들어 데이터베이스 또는 스토리지 접근 방식을 선택할 때 데이터 액세스 패턴을 고려합니다.

정의

클라우드에는 성능 효율성에 대한 넷개의 모범 사례 영역이 있습니다.

- 선택
- 검토
- 모니터링
- 트레이드오프

고성능 아키텍처 선택 시 데이터 기반 접근 방식을 취합니다. 상위 수준 설계에서 리소스 유형의 선택 및 구성에 이르기까지 아키텍처의 모든 측면에 대한 데이터를 수집합니다. 주기적으로 선택 사항을 검토함으로써 계속 진화하는 AWS 클라우드를 최대한 활용할 수 있습니다. 모니터링은 예상 성능과의 차이를 인지하고 관련 조치를 취할 수 있게 해 줍니다. 마지막으로 아키텍처는 압축 또는 캐싱을 사용하거나 일관성 요구 사항을 완화하는 등 트레이드오프를 통해 성능을 개선할 수 있습니다.

모범 사례

선택

특정 시스템에 대한 최적의 솔루션은 워크로드에 따라 달라지며, 흔히 여러 접근 방식이 복합적으로 사용됩니다. 설계가 잘된 시스템은 여러 개의 솔루션을 사용하고 다양한 특성을 사용하여 성능을 높입니다.

AWS에서는 리소스가 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 보다 쉽게 요구 사항과 일치하는 접근 방식을 찾을 수 있을 뿐만 아니라, 온프레미스 인프라에서는 쉽게 얻을 수 없는 다양한 스토리지 옵션을 찾을 수 있습니다. 예를 들어 Amazon DynamoDB와 같은 관리형 서비스는 완전관리형 NoSQL 데이터베이스로, 어떤 규모에서도 지연 시간이 한 자릿수 밀리초 단위로 매우 짧습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다. (성능 효율성 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.).

PERF 1: 최고의 성능을 발휘하는 아키텍처를 선택하기 위하여 어떤 방식을 사용하십니까?

워크로드에서 최적의 성능을 얻으려면 여러 가지 방식이 필요합니다. 설계가 잘된 시스템은 여러 개의 솔루션을 사용하고 다양한 특성을 사용하여 성능을 높입니다.

아키텍처에 대한 패턴 및 구현을 선택할 때는 최적의 솔루션을 위한 데이터 기반 접근 방식을 사용합니다. AWS 솔루션스 아키텍트, AWS 참조 아키텍처 및 AWS 파트너 네트워크(APN) 파트너는 이제까지 구축한 지식을 바탕으로 사용자가 아키텍처를 선택하는 과정을 도울 수 있습니다. 하지만 아키텍처를 최적화하기 위해서는 벤치마킹 또는 로드 테스트를 통해 획득한 데이터가 필요합니다.

아키텍처는 여러 아키텍처 접근 방식(예: 이벤트 기반, ETL 또는 파이프라인)을 결합하게 될 가능성이 높습니다. 아키텍처 구현에는 아키텍처 성능 최적화에 적합한 AWS 서비스를 사용하게 됩니다. 다음 단원에서는 네 가지의 고려해야 할 리소스 유형, 즉 컴퓨팅, 스토리지, 데이터베이스 및 네트워크에 대해 살펴봅니다.

컴퓨팅

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

AWS에서는 인스턴스, 컨테이너, 함수 등 세 가지 형태의 컴퓨팅이 제공됩니다.

- 인스턴스는 가상화된 서버이며, 따라서 버튼을 클릭하거나 API 호출을 통해 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 크기로 제공하며 솔리드 스테이트 드라이브(SSD)와 그래픽 처리 장치(GPU)를 비롯한 매우 다양한 기능을 제공합니다.
- 컨테이너는 애플리케이션 및 종속성을 리소스가 격리된 프로세스에서 실행할 수 있는 운영 체제 가상화 방식입니다.
- 함수는 실행하려는 코드에서 실행 환경을 추상화합니다. 예를 들어, AWS Lambda를 이용하면 인스턴스를 실행하지 않고도 코드를 실행할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 2: 컴퓨팅 솔루션 선택 방법

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

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

스토리지

특정 시스템에 대한 최적의 스토리지 솔루션은 액세스 방식 종류(블록, 파일, 객체), 액세스 패턴(랜덤 또는 순차), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적) 및 가용성과 내구성 제약 사항에 따라 다릅니다. 설계가 잘 된 시스템은 여러 개의 스토리지 솔루션을 사용하고 다양한 특성을 사용하여 성능을 향상시킵니다.

AWS에서 스토리지는 가상화되며 다양한 유형으로 제공됩니다. 이는 스토리지 방식을 요구에 보다 밀접하게 맞출 수 있게 하며, 온프레미스 인프라에서는 쉽게 얻을 수 없는 다양한 스토리지 옵션을 제공합니다. 예를 들어, Amazon S3는 99.999999999%의 내구성을 제공하도록 설계되었습니다. 또한 마그네틱 하드 디스크 드라이브(HDD)에서 솔리드 스테이트 드라이브(SSD)로 변경할 수 있으며, 몇 초 안에 한 인스턴스에서 다른 인스턴스로 가상 드라이브를 쉽게 이동할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 3: 스토리지 솔루션 선택 방법

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

스토리지 솔루션을 선택할 때는 액세스 패턴과 일치하도록 선택하는 것이 원하는 성능을 구현하는 데 매우 중요합니다.

데이터베이스

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

Amazon RDS는 완전관리형 관계형 데이터베이스를 제공합니다. Amazon RDS를 이용하면 가동을 중단하는 경우가 거의 없이 데이터베이스의 컴퓨팅과 스토리지 리소스를 확장할 수 있습니다. Amazon DynamoDB는 어떤 규모에서도 지연 시간이 한 자릿수 밀리초 단위로 매우 짧은 완전관리형 NoSQL 데이터베이스입니다. Amazon Redshift는 페타바이트 규모의 관리형 데이터 웨어하우스로, 성능 또는 용량을 변경해야 할 경우 노드 수 또는 유형을 변경할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 4: 데이터베이스 솔루션 선택 방법

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

한 워크로드의 데이터베이스 접근 방식(RDBMS, NoSQL)은 시스템의 성능 효율에 큰 영향을 미치지만, 이는 보통 데이터 기반 접근 방식보다는 조직적 기본 사항에 따라 선택됩니다. 스토리지와 마찬가지로 워크로드의 액세스 패턴을 고려하는 것이 중요하며, 다른 비데이터베이스 솔루션이 보다 효율적으로 문제(예: 검색 엔진 또는 데이터 웨어하우스 사용)를 해결할 수 있는지 여부도 고려해야 합니다.

네트워크

특정 시스템에 맞는 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항 등에 따라 다양합니다. 사용자 또는 온프레미스 리소스와 같은 물리적 제약은 엣지 기술 또는 리소스 배치를 통해 오프셋될 수 있는 위치 옵션을 구동시킵니다.

AWS에서 네트워킹은 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 네트워킹 방식을 요구에 보다 밀접하게 맞출 수 있습니다. AWS는 네트워크 트래픽을 최적화하는 제품 기능(예: 향상된 네트워킹, Amazon EBS 최적화 인스턴스, Amazon S3 전송 가속 기능, 동적 Amazon CloudFront)을 제공합니다. 또한 AWS는 네트워크 거리 또는 지터를 줄이기 위한 네트워킹 기능(예: Amazon Route53 지연 시간 기반 라우팅, Amazon VPC 엔드포인트 및 AWS Direct Connect)도 제공합니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 5: 네트워킹 솔루션 구성 방법

시스템에 맞는 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항 등에 따라 다양합니다. 사용자 또는 온프레미스 리소스와 같은 물리적 제약은 엣지 기술 또는 리소스 배치를 통해 상쇄될 수 있는 위치 옵션을 구동시킵니다.

네트워크 솔루션을 선택할 때는 위치를 고려해야 합니다. AWS를 사용하면 리소스를 사용할 위치 가까이 해당 리소스가 배치되도록 선택하여 거리를 줄일 수 있습니다. 리전, 배치 그룹 및 엣지 로케이션을 활용하여 성능을 현저히 개선할 수 있습니다.

검토

솔루션 설계 시 선택할 수 있는 옵션은 한정되어 있습니다. 그러나 시간이 지나면 아키텍처의 성능을 향상시킬 수 있는 새로운 기술과 접근 방식을 사용할 수 있게 됩니다.

AWS를 사용하면 고객의 요구를 충족하기 위해 지속적인 혁신의 혜택을 받을 수 있습니다. AWS는 정기적으로 새로운 리전, 엣지 로케이션, 서비스 및 기능을 출시합니다. 이들 모두는 아키텍처의 성능 효율성을 확실히 개선할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 6: 새 릴리스를 활용하기 위해 워크로드를 개선하는 방법

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

어떤 조건이 아키텍처 성능을 제약하는지 이해하면 해당 제약 조건을 완화할 수 있는 릴리스를 찾아볼 수 있습니다.

모니터링

아키텍처를 구현한 후에는 고객이 인지하기 전에 모든 문제를 해결할 수 있도록 성능을 모니터링해야 합니다. 임계값을 초과할 경우 경보가 생성되도록 모니터링 측정치를 사용해야 합니다. 경보는 성능이 불량한 구성 요소를 해결하기 위한 자동 작업을 트리거할 수 있습니다.

Amazon CloudWatch는 모니터링 기능 및 알림 경보 전송 기능을 제공합니다. 자동화를 이용하면 Amazon Kinesis, Amazon Simple Queue Service(Amazon SQS) 및 AWS Lambda를 통해 작업을 트리거하여 성능 문제를 해결할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 7: 예상된 성능이 발휘되도록 리소스를 모니터링하는 방법

시스템 성능은 시간이 지남에 따라 저하될 수 있습니다. 시스템 성능을 모니터링하여 성능 저하 상태를 식별하고 운영체제 또는 애플리케이션 부하와 같은 내부 또는 외부 요인을 해소합니다.

정책 오차 또는 데이터가 과도하게 발생하지 않도록 하는 것이 효과적인 모니터링 솔루션의 핵심입니다. 자동 트리거는 수동 작업으로 인한 오류를 방지하고 문제 해결 시간을 단축시킬 수 있습니다. 프로덕션 환경에서 시뮬레이션을 통해 경보 솔루션이 올바로 문제를 인지하는지 테스트하는 실전 연습을 계획하십시오.

트레이드오프

솔루션을 설계할 때는 최적의 접근 방식을 선택할 수 있도록 트레이드오프를 고려해야 합니다. 상황에 따라 일관성, 내구성 및 공간을 위해 시간 또는 지연 시간을 희생함으로써 보다 높은 성능을 제공할 수 있습니다.

AWS를 사용하면 몇 분 내에 전 세계의 여러 위치에 리소스를 배포하여 최종 사용자에게 더욱 가깝게 다가갈 수 있습니다. 또한 데이터베이스 시스템과 같은 정보 스토어에 읽기 전용 복제본을 동적으로 추가하여 기본 데이터베이스의 부하를 줄일 수 있습니다. 또한 AWS는 인 메모리 스토어 또는 캐시를 제공하는 Amazon ElastiCache와 같은 캐싱 솔루션과 정적 콘텐츠의 사본을 최종 사용자에게 더 가깝게 캐싱하는 Amazon CloudFront를 제공합니다. Amazon DynamoDB Accelerator(DAX)는 Amazon DynamoDB 앞에 연속 읽기/연속 쓰기 분산 캐싱 계층을 제공하여 동일한 API를 지원하면서도 캐시 내에 있는 엔터티에 대해 1밀리초 미만의 지연 시간을 제공합니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 8: 성능 향상을 위해 트레이드오프를 어떻게 사용하고 있습니까?

솔루션을 설계할 때 트레이드오프를 적극적으로 고려하면 최적의 접근 방식을 선택할 수 있습니다. 일관성, 내구성, 공간을 시간과 지연 시간으로 바꾸어 성능을 수시로 향상시킬 수 있습니다.

트레이드오프는 아키텍처의 복잡성을 증가시킬 수 있으며 측정 가능한 이점이 달성되는지 확인하기 위한 로드 테스트가 필요합니다.

주요 AWS 서비스

성능 효율성에 필수적인 AWS 서비스은(는) Amazon CloudWatch, 리소스와 시스템을 모니터링하여 전반적인 성능 및 운영 상태를 확인할 수 있도록 합니다.이며, 다음 서비스 및 기능이 성능 효율성의 넷개 영역을 지원합니다.

- 선택
  - 컴퓨팅: Auto Scaling은 수요를 충족하고 응답 속도를 유지할 수 있는 충분한 인스턴스를 보장해 줍니다.

  - 스토리지: Amazon EBS는 사용 사례에 맞춰 최적화할 수 있는 광범위한 스토리지 옵션(예: SSD 및 프로비저닝된 IOPS(초당 입/출력 작업 수)을 제공합니다. Amazon S3는 서버리스 콘텐츠 전송을 제공하며 Amazon S3 Transfer Acceleration은 장거리에서 파일을 빠르고 쉽고 안전하게 전송할 수 있게 해 줍니다.
  - 데이터베이스: Amazon RDS는 사용 환경에 맞춰 최적화할 수 있게 해 주는 다양한 데이터베이스 기능(프로비저닝된 IOPS, 읽기 전용 복제본 등)을 제공합니다. Amazon DynamoDB는 어떤 규모에서도 한 자릿수 밀리초 단위로 매우 짧은 지연 시간을 제공합니다.
  - 네트워크: Amazon Route53은 지연 시간 기반 라우팅을 제공합니다. Amazon VPC 엔드포인트 및 AWS Direct Connect는 네트워크 거리 또는 지터를 줄일 수 있습니다.
- 검토: AWS 웹사이트의 AWS 블로그와 새로운 기능 섹션은 새롭게 출시되는 기능과 서비스에 대해 알아보기 위한 리소스입니다.
- 모니터링: Amazon CloudWatch는 기존의 모니터링 솔루션과 통합할 수 있고 AWS Lambda와 함께 사용하여 작업을 트리거할 수 있는 측정치, 경보 및 알림을 제공합니다.
- 트레이드오프: Amazon ElastiCache, Amazon CloudFront 및 AWS Snowball은 성능 개선을 위해 사용할 수 있는 서비스입니다. Amazon RDS의 읽기 전용 복제본을 사용하면 읽기 중심의 워크로드를 확장할 수 있습니다.

리소스

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

설명서

- Amazon S3 Performance Optimization
- Amazon EBS Volume Performance

백서

- Performance Efficiency Pillar

동영상

- AWS re:Invent 2016: Scaling Up to Your First 10 Million Users (ARC201)
- AWS re:Invent 2017: Deep Dive on Amazon EC2 Instances

비용 최적화

비용 최적화 원칙에는 시스템을 실행하여 최저 가격으로 비즈니스 가치를 제공할 수 있는 기능입니다 이(가) 포함됩니다.

비용 최적화 부문에서는 설계 원리 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 규범적 지침은 비용 최적화 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 비용 최적화에 대한 다섯개의 설계 원칙이 있습니다.

- 소비 모델 도입: 정교한 예측 기능을 사용하지 않아도 필요한 컴퓨팅 리소스에만 비용을 지불하고 비즈니스 요구 사항에 따라 사용량을 늘리거나 줄입니다. 예를 들어 개발 및 테스트 환경은 주로 주중 근무일 중에 하루 8시간 동안만 사용됩니다. 사용되지 않는 동안 이러한 리소스를 중단하여 잠재적으로 75%의 비용을 절감할 수 있습니다(40시간과 168시간의 차이).
- 전반적인 효율성 측정: 워크로드의 비즈니스 결과 및 워크로드 제공과 관련된 비용을 측정합니다. 이 측정 기능을 사용하면 늘어난 성과와 절감한 비용을 확인할 수 있습니다.
- 데이터 센터 운영에 비용 투자 불필요: AWS에서 서버를 랙에 설치하고 쌓아 올리고 서버에 전원을 공급하는 힘든 작업을 처리하기 때문에 IT 인프라보다 고객과 조직 프로젝트에 주력할 수 있습니다.
- 지출 분석 및 기여: 클라우드를 사용하면 손쉽게 시스템의 사용량과 비용을 정확하게 식별할 수 있어 개별 워크로드 소유자가 IT 비용에 대한 투명한 기여도를 확인할 수 있습니다. 이는 ROI(투자 대비 수익률)를 측정하고 워크로드 소유자가 리소스를 최적화하고 비용을 절감하는 기회를 얻을 수 있도록 해 줍니다.
- 관리형 및 애플리케이션 수준 서비스를 사용하여 소유권 비용 절감: 클라우드에서 관리형 및 애플리케이션 수준 서비스는 이메일 전송 또는 데이터베이스 관리와 같은 작업의 서버 유지 관리에 대한 작업 부담을 덜어줍니다. 관리형 서비스가 클라우드 규모에서 운영되기 때문에 트랜잭션 또는 서비스당 비용을 절감할 수 있습니다.

정의

클라우드에는 비용 최적화에 대한 넷개의 모범 사례 영역이 있습니다.

- 지출 인식
- 비용 효율적인 리소스
- 수요와 공급 일치
- 시간 경과에 따른 최적화

다른 부문에서와 같이 비용 최적화 부문에서도 절충이 필요한 요소를 고려해야 합니다. 출시 시간 또는 비용을 최적화하려는 경우를 예로 들어 보겠습니다. 선결제 비용 최적화에 투자하는 것보다 출시 시간을 단축하거나 새로운 기능을 배포하거나 단순히 납기를 준수하는 등 속도를 우선적으로 고려하는 것이 가장 좋은 경우도 있습니다. 시간이 지남에 따라 가장 비용 최적화된 워크로드 벤치마킹에 시간을 쓰기보다, “만약을 대비해” 과잉 지출을 하려는 유혹은 항상 존재하기 때문에 경험적인 데이터가 아니라 급하게 설계 결정이 내려지는 경우가 있습니다. 이는 종종 현저히 오버프로비저닝되고 최적화가 부족한 배포로 이어지게 되고, 이로 인해 수명 주기 전반이 정적으로 유

지됩니다. 다음 섹션에는 초기 및 진행 중 배포 비용 최적화에 대한 기술과 전략적 가이드가 나와 있습니다.

모범 사례

지출 인식

클라우드 덕분에 늘어난 유연성과 민첩성은 혁신과 빠른 개발 및 배포를 촉구합니다. 이는 하드웨어 사양을 식별하고, 가격 견적을 협상하고, 주문 번호를 관리하고, 배송을 예약한 후 리소스 배포하기와 같은 온프레미스 인프라 프로비저닝과 연관된 시간 및 수동 프로세스를 제거합니다. 하지만 사용이 편리하고 온디맨드 용량이 사실상 무제한으로 제공되면 지출을 새로운 방식으로 고려해야 합니다.

많은 비즈니스는 다양한 팀에서 운영하는 여러 시스템으로 구성되어 있습니다. 개별 조직 또는 제품 소유자에게 리소스 비용을 부여하는 기능은 효율적인 사용 행동 양식으로 이어지고 낭비되는 요소를 줄여 줍니다. 또한 정확한 비용 기여도는 진짜 수익을 내는 제품을 확인하고 예산을 어디에 할당할지에 대해 보다 정보를 근거로 한 결정을 내릴 수 있게 해 줍니다.

AWS에서는 Cost Explorer를 사용하여 지출을 추적하고 정확히 지출된 부분에 대한 인사이트를 확보할 수 있습니다. AWS 예산을 사용하여 사용량이나 비용이 예측된 선상에 없을 경우 알림을 전송할 수 있습니다. 리소스에 태그 지정을 사용하여 비즈니스와 조직 정보를 사용량 및 비용에 적용할 수 있습니다. 이는 조직 관점에서 최적화에 대한 추가 인사이트를 제공합니다.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다. (비용 최적화 관련 질문 및 모범 사례 목록은 부록을 참조하십시오.).

COST 1: 사용량 관리 방법
목표 달성 과정에서 발생하는 비용을 적정 수준으로 유지하는 정책과 메커니즘을 설정합니다. '견제와 균형' 방식을 도입하면 비용을 과도하게 지출하지 않고 획기적인 방식으로 목표를 달성할 수 있습니다.
COST 2: 사용량과 비용을 모니터링하는 방법
비용을 모니터링하고 적절하게 할당하기 위한 정책 및 절차를 구성합니다. 이렇게 하면 이 워크로드의 비용 효율성을 측정하고 개선할 수 있습니다.
COST 3: 리소스 폐기 방법
프로젝트 시작부터 마지막까지의 전체 과정에서 변경 제어 및 리소스 관리를 구현합니다. 그러면 사용되지 않은 리소스를 종료하여 낭비되는 리소스를 줄일 수 있습니다.

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

태그가 지정된 리소스와 엔터티 수명 주기 추적(직원, 프로젝트)을 결합하면 조직에 더 이상 가치를 제공하지 않으므로 폐기해야 할 분리된 리소스 또는 프로젝트를 식별할 수 있습니다. 결제 알림을 설정하여 예측된 과다 지출을 알릴 수 있으며, AWS 월 사용량 계산기를 사용하면 데이터 전송 비용을 계산할 수 있습니다.

비용 효율적인 리소스

워크로드에 적절한 인스턴스와 리소스 사용은 비용 절감의 핵심 요소입니다. 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 사항에 가장 적합한 방식으로 EC2 및 다른 서비스의 인스턴스를 획득하도록 유연하고 비용 효율적인 요금 옵션을 매우 다양하게 제공합니다. 온디맨드 인스턴스의 경우 최소 약정이 필요 없으며 시간 단위로 컴퓨팅 파워 비용을 지불할 수 있습니다. 예약 인스턴스의 경우에는 용량을 예약할 수 있으며, 온디맨드 요금이 최대 75%까지 할인됩니다. 스팟 인스턴스를 사용하면 사용하지 않는 Amazon EC2 용량을 활용하고 최대 90% 할인된 온디맨드 요금을 제공할 수 있습니다. 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 완전히 해소할 수 있으며, Amazon Aurora on RDS를 활용하면 값비싼 데이터베이스 라이선싱 비용을 해소할 수 있습니다.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 4: 서비스 선택 시의 비용 평가 방법
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 사용하여 관리 및 운영 고정 비요을 많이 줄이거나 없앨 수 있으며, 응용 프로그램 및 비즈니스 관련 활동을 수행할 수 있습니다.
COST 5: 리소스 유형과 크기 선택 시 비용 목표를 충족하는 방법
진행 중인 작업에 대해 적절한 리소스 크기를 선택해야 합니다. 가장 비용 효율적인 유형 및 크기를 선택하면 리소스 낭비를 최소화할 수 있습니다.
COST 6: 요금 모델을 사용하여 비용을 절감하는 방법
리소스가 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.
COST 7: 데이터 전송 요금 부과 계획 방법
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.

수요와 공급 일치

수요에 대한 공급을 최적으로 일치시키면 워크로드에 대한 최저 비용이 제공되지만 프로비저닝 시간 및 개별 리소스 장애를 대비하여 충분한 추가 공급도 필요합니다. 수요가 고정되거나 가변적일 수 있어서, 관리가 중요한 비용 요소가 되지 않도록 측정 및 자동화가 필요합니다.

AWS에서는 수요와 일치하도록 리소스를 자동으로 프로비저닝할 수 있습니다. Auto Scaling 및 수요, 버퍼, 시간 기반 접근 방식을 활용하면 필요한 만큼 리소스를 추가하고 제거할 수 있습니다. 수요의 변화를 예측할 수 있을 경우 비용을 보다 많이 절감하고 리소스가 워크로드 요구 사항과 일치하도록 할 수 있습니다.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 8: 수요와 일치하도록 리소스를 공급하는 방법

비용과 성능을 적절하게 절충한 워크로드에서는 비용을 결제한 모든 리소스가 사용되는지 확인하고, 사용률이 매우 낮은 인스턴스가 없도록 해야 합니다. 사용률 지표가 매우 높거나 낮은 리소스가 있으면 조직의 운영 비용이 증가하거나(사용률이 너무 높아 성능이 저하됨), 과도한 프로비저닝으로 인해 AWS에 지출한 금액이 낭비됩니다.

수요에 대해 공급이 일치하도록 설계할 경우 새 리소스를 프로비저닝하는 데 소요되는 시간과 사용량 패턴에 대해 적극적으로 고민해야 합니다.

시간 경과에 따른 최적화

AWS가 새로운 서비스와 기능을 출시하면 계속해서 가장 비용 효율적인 방식을 취하도록 기존 설계상 결정을 검토하는 것이 가장 좋습니다. 요구 사항이 변경되면 더 이상 필요하지 않은 리소스, 전체 서비스 및 시스템을 적극적으로 폐기하십시오.

AWS의 관리형 서비스는 워크로드를 상당히 최적화할 수 있으므로 사용 가능한 새 관리형 서비스와 기능을 파악해야 합니다. 예를 들어 Amazon RDS 데이터베이스를 운영하는 것이 Amazon EC2에서 자체 데이터베이스를 운영하는 것보다 저렴할 수 있습니다.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 9: 새로운 서비스를 어떻게 평가하십니까?

AWS는 새로운 서비스와 기능을 출시하므로, 기존 아키텍처 결정을 검토하여 최고의 비용 효율성을 유지하는 것이 좋습니다.

정기적으로 배포를 검토할 때 최신 서비스가 비용을 절약하는 데 어떻게 도움이 될 수 있는지 평가하십시오. 예를 들어 Amazon Aurora on RDS를 사용하면 관계형 데이터베이스의 비용을 줄일 수 있습니다.

주요 AWS 서비스

비용 최적화에 필수적인 도구은(는) 비용 탐색기, 조직 전반에 걸쳐 전체 워크로드 및 워크로드 사용량을 확인할 수 있습니다.이며, 다음 서비스 및 기능이 비용 최적화의 넷개 영역을 지원합니다.

- 지출 인식: AWS Cost Explorer를 사용하면 사용량에 대한 세부 정보를 확인하고 추적할 수 있습니다. AWS Budgets는 사용량이나 지출이 실제 또는 예측 예산 값을 초과했는지 여부를 알려줍니다.
- 비용 효율적인 리소스: 예약 인스턴스 추천 사항에 대해 Cost Explorer를 사용하고 시간이 지남에 따라 AWS 리소스에 대한 비용 지출 패턴을 살펴볼 수 있습니다. Amazon CloudWatch 및 Trusted Advisor를 사용하면 리소스의 크기를 올바르게 조정할 수 있습니다. Amazon Aurora on RDS를 사용하여 데이터 라이선싱 비용을 제거할 수 있습니다. AWS Direct Connect 및 Amazon CloudFront는 데이터 전송을 최적화하는 데 사용할 수 있습니다.
- 수요와 공급 일치: Auto Scaling을 활용하면 과도한 지출 없이 수요가 일치하도록 리소스를 추가하거나 제거할 수 있습니다.
- 시간 경과에 따른 최적화: AWS 웹사이트의 AWS 뉴스 블로그와 새로운 기능 섹션은 새롭게 출시되는 기능과 서비스에 대해 알아보기 위한 리소스입니다. AWS Trusted Advisor는 AWS 환경을 점검하고 사용하지 않거나 유휴 상태인 리소스를 제거하거나 예약 인스턴스 용량에 주력하여 비용을 절감할 기회를 파악하게 해 줍니다.

리소스

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

설명서

- Analyzing Your Costs with Cost Explorer
- AWS Cloud Economics Center
- AWS Detailed Billing Reports

백서

- Cost Optimization Pillar

동영상

- Cost Optimization on AWS

도구

- AWS Total Cost of Ownership (TCO) Calculators
- AWS Simple Monthly Calculator

검토 프로세스

아키텍처 검토는 자세한 분석을 장려하는 접근을 통해 일관적인 방식으로 수행되어야 합니다. 감사가 아니라 대화로 진행되는 가벼운 프로세스(며칠이 아닌 몇 시간)여야 합니다. 아키텍처를 검토하는 목적은 해결해야 할 영역이나 개선해야 할 부분을 파악하는 것입니다. 검토 결과는 워크로드를 사용하는 고객의 경험을 개선해야 하는 일련의 작업입니다.

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

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

변경하기 어려운 단방향 방식[1] 을 방지할 수 있도록 가동하기 전 설계 단계 초반에 제품 수명 주기의 주요 이정표에서 검토를 적용해야 합니다. 그 이후 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 우수한 청결한 관행을 따라야 합니다. 중요한 아키텍처 변경을 수행할 때에는 Well-Architected 검토를 비롯한 일련의 청결한 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: “암호화 활성화 여부”)

검토를 마친 후에는 비즈니스 상황에 따라 우선 순위를 지정할 수 있는 문제 목록을 보유해야 합니다. 또한 이러한 문제가 팀의 일상적인 업무에 미치는 영향을 고려하고자 할 수도 있습니다. 이러

[1] 대부분의 결정을 내릴 때는 번복 가능한 두 가지 옵션 중에서 선택을 하게 됩니다. 이러한 결정 과정에서는 간단한 프로세스를 사용할 수 있습니다. 단방향 방식은 번복하기 어렵거나 불가능하기 때문에 결정을 내리기 전에 더 많은 검사를 진행해야 합니다.

한 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, 새로운 팀이 처음에 반감을 가질 수 있다는 점을 확인하게 될 수 있습니다. 다음은 몇 가지 이의 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 언급됨)
  - 이 이벤트는 이동할 수 없습니다. 아키텍처에 대한 위험을 파악하지 않은 채 그대로 진행하고 싶으십니까? 이러한 모든 문제를 해결하지 못하더라도, 구체화하는 경우 해당 문제를 처리하기 위한 지침서를 가질 수 있습니다.
- "We don't want others to know the secrets of our solution implementation!"
  - Well-Architected 프레임워크의 질문을 팀에게 제시하면 어떠한 질문에서도 상업적 또는 기술적 독점 정보를 확인할 수는 없음을 알 수 있습니다.

조직 내 팀과 여러 검토를 수행하면서 주제별 문제를 식별할 수 있습니다. 예를 들어 여러 팀이 특정 기반 또는 주제에 대한 문제를 갖는 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 클라우드상의 안정적이고 안전하며 효율적이고 경제적인 시스템을 설계하고 운영하기 위한 다섯 가지 주요 기반의 설계 모범 사례를 제공합니다. 이 프레임워크에서는 기존 아키텍처 또는 제안된 아키텍처를 검토할 수 있는 몇 가지 질문과 각 기반에 대한 AWS 모범 사례를 제시합니다. 아키텍처에 이 프레임워크를 사용하면 기능적 요구 사항에 초점을 둔 안정적이고 효율적인 시스템을 구축할 수 있습니다.

기고자

다음은 본 문서 작성에 도움을 준 개인 및 조직입니다.

- "Fitz" Philip Fitzsimons: 선임 선임 관리자, Amazon Web Services
- Brian Carlson: Well-Architected 운영 부문 담당자, Amazon Web Services
- Ben Potter: Well-Architected 보안 부문 담당자, Amazon Web Services
- Rodney Lester: Well-Architected 안정성 부문 담당자, Amazon Web Services
- John Ewart: Well-Architected 성능 부문 담당자, Amazon Web Services
- Nathan Besh: Well-Architected 비용 부문 담당자, Amazon Web Services
- Jon Steele: 선임 기술 계정 관리자, Amazon Web Services
- Ryan King: 기술 프로그램 관리자, Amazon Web Services
- Erin Rifkin: 선임 제품 관리자, Amazon Web Services
- Max Ramsay: 수석 보안 솔루션스 아키텍트, Amazon Web Services
- Scott Paddock: 보안 솔루션스 아키텍트, Amazon Web Services
- Callum Hughes: 솔루션스 아키텍트, Amazon Web Services

참고 문헌

AWS Well-Architected Partner program

AWS Well-Architected Tool

AWS Well-Architected homepage

Cost Optimization Pillar whitepaper

Operational Excellence Pillar whitepaper

Performance Efficiency Pillar whitepaper

Reliability Pillar whitepaper

Security Pillar whitepaper

문서 수정

표 2.

주요 개정 사항:

날짜	설명
2019년 7월	AWS Well-Architected Tool, AWS Well-Architected Labs에 대한 링크 및 AWS Well-Architected 파트너 추가, 여러 언어 버전의 프레임워크를 지원하는 몇 가지 수정 사항.
2018년 11월	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 6월	업데이트를 통해 질문 텍스트를 더 단순하게 작성하고 답변을 표준화했으며 가독성을 개선했습니다.
2017년 11월	운영 우수성을 콘텐츠 앞부분으로 옮겨 다시 작성했으며 이에 따라 다른 콘텐츠가 구성됨. AWS의 발전을 반영하기 위해 다른 기반을 새로 고침.
2016년 11월	운영 우수성 기반을 포함하도록 프레임워크를 업데이트하고, 중복을 줄이기 위해 다른 기반을 개정 및 업데이트했으며, 수천 명의 고객과 검토하여 획득한 내용을 통합.
2015년 11월	최신 Amazon CloudWatch Logs 정보로 부록 업데이트.
2015년 10월	초판 발행.

부록: 질문 및 모범 사례

운영 우수성

준비

OPS 1 우선 순위를 결정하는 방법

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

모범 사례:

- 외부 고객 요구 평가:: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 외부 고객 요구 충족을 위한 운영 작업을 중점적으로 진행할 영역을 결정합니다. 이렇게 하면 비즈니스 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
- 내부 고객 요구 평가:: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 내부 고객 요구 충족을 위한 운영 작업을 중점적으로 진행할 영역을 결정합니다. 이렇게 하면 비즈니스 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
- 규정 준수 요구 사항 평가:: 규정 준수 요구 사항 및 산업 표준과 같은 외부 요인을 평가하여 특정 작업을 반드시/집중적으로 수행해야 할 수 있는 의무 사항이나 지침을 파악해야 합니다. 규정 준수 요구 사항이 확인되지 않으면 적절한 판단에 따라 해당 요구 사항을 결정해야 합니다.
- 위협 환경 평가:: 운영 작업을 집중적으로 수행할 분야를 결정할 때 해당 영향을 포함할 수 있도록 업무상의 위협 요소(예: 경쟁, 업무상의 위험/책임, 운영상의 위험, 정보 보안 위협)를 평가합니다.
- 장단점 평가: 운영 작업을 집중적으로 수행할 분야를 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 상충하는 이해 관계의 장단점을 평가합니다. 예를 들어 비용 최적화보다 새 기능의 신속한 출시를 우선적으로 수행할 수 있습니다.
- 이점 및 위험 관리: 운영 작업을 집중적으로 수행할 분야를 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 이점과 위험을 관리합니다. 예를 들어 중요한 새 기능을 고객에게 제공하려는 경우에는 해결되지 않은 문제가 있는 시스템을 배포하는 것이 유용할 수 있습니다.

OPS 2 워크로드 상태를 파악할 수 있도록 워크로드를 설계하는 방법

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

모범 사례:

- 애플리케이션 원격 측정 구현:: 애플리케이션 코드를 계측하여 내부 상태 및 비즈니스 성과 달성 관련 정보를 내보냅니다. 대기열 길이, 오류 메시지, 응답 시간 등의 정보를 내보낼 수 있습니다. 이 정보를 사용하여 응답이 필요한 경우를 확인합니다.
- 워크로드 원격 측정 구현 및 구성:: 내부 상태와 현재 상태 관련 정보를 내보내도록 워크로드를 설계 및 구성합니다. API 콜 현황, http 상태 코드, 스케일링 이벤트 등의 정보를 내보낼 수 있습니다. 이 정보를 사용하면 응답이 필요한 경우를 확인할 수 있습니다.
- 사용자 활동 원격 측정 구현:: 애플리케이션 코드를 계측하여 사용자 활동 관련 정보를 내보냅니다. 클릭 스트림 또는 시작/중단/완료된 트랜잭션 등의 정보를 내보낼 수 있습니다. 이 정보를 사용하면 애플리케이션 사용 방법과 사용 패턴을 파악하고 응답이 필요한 경우를 확인할 수 있습니다.
- 종속성 원격 측정 구현:: 워크로드가 사용하는 리소스의 상태 관련 정보를 내보내도록 워크로드를 설계 및 구성합니다. 이러한 리소스의 예로는 외부 데이터베이스, DNS, 네트워크 연결 등이 있습니다. 이 정보를 사용하여 응답이 필요한 경우를 확인합니다.
- 트랜잭션 추적 기능 구현:: 워크로드 전반에 걸친 트랜잭션 흐름 관련 정보를 내보내도록 애플리케이션 코드를 구현하고 워크로드 구성 요소를 구성합니다. 이 정보를 사용하면 응답이 필요한 경우를 확인하고 문제의 근본 원인을 파악할 수 있습니다.

OPS 3 결함을 줄이고 문제를 쉽게 해결하고 프로덕션 환경으로의 흐름을 개선하는 방법

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

모범 사례:

- 버전 관리 사용:: 버전 관리를 사용하면 변경 사항과 릴리스를 추적할 수 있습니다.
- 변경 사항 테스트 및 확인:: 변경 사항을 테스트하고 확인하면 오류를 제한하고 감지할 수 있습니다. 그리고 테스트를 자동화하면 수동 프로세스에서 발생하는 오류와 테스트해야 하는 작업량을 줄일 수 있습니다.
- 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.
- 빌드 및 배포 관리 시스템 사용:: 빌드 및 배포 관리 시스템을 사용합니다. 이러한 시스템에서는 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업량을 줄일 수 있습니다.
- 패치 관리 수행:: 패치 관리를 수행하면 기능을 확인하고, 문제를 해결하고, 거버넌스 규정 준수 상태를 유지할 수 있습니다. 그리고 패치 관리를 자동화하면 수동 프로세스에서 발생하는 오류와 패치를 위한 작업량을 줄일 수 있습니다.
- 설계 표준 공유:: 여러 팀이 모범 사례를 공유하면 표준에 대한 인지도를 높이고 개발 작업의 이점을 극대화할 수 있습니다.
- 코드 품질 개선을 위한 사례 구현:: 코드 품질을 개선하고 결함을 최소화하는 사례를 구현합니다. 테스트 중심 개발, 코드 검토, 표준 도입 등을 예로 들 수 있습니다.
- 여러 환경 사용:: 여러 환경을 사용하여 워크로드를 실험, 개발 및 테스트합니다. 제어 수준을 계속 높이면서 워크로드의 정상 작동 신뢰도를 유지합니다.
- 작게 자주 발생하고 되돌릴 수 있는 변경 내용 적용:: 되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 영향과 범위가 감소합니다. 이렇게 하면 문제를 더 빠르고 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용할 수 있습니다.
- 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.

OPS 4 배포 위험을 완화하는 방법

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:

- 부적절한 변경을 수행한 경우의 계획 수립:: 변경을 수행했는데 적절한 성과를 달성하는 데 도움이 되지 않는다면 알려진 정상 상태로 되돌릴 수 있는 계획을 세우거나 프로덕션 환경에서 관련 문제를 해결합니다. 이와 같이 준비를 하면 문제에 더욱 신속하게 대응함으로써 복구 시간을 단축할 수 있습니다.
- 변경 사항 테스트 및 확인:: 수명 주기의 모든 단계에서 변경 사항을 테스트하고 결과를 검증하여 새 기능을 확인하고 배포 실패의 영향과 위험을 최소화합니다.
- 배포 관리 시스템 사용:: 배포 관리 시스템을 사용하여 변경을 추적하고 구현합니다. 이렇게 하면 수동 프로세스에서 발생하는 오류와 변경 사항 배포를 위한 작업을 줄일 수 있습니다.
- 제한된 배포를 사용하여 테스트:: 전체 배포를 진행하기 전에 기존 시스템과 함께 제한된 배포를 사용해 테스트를 진행하여 원하는 결과를 달성할 수 있는지를 확인합니다. 예를 들어 Canary 배포 테스트나 원박스 배포를 사용합니다.
- 병렬 환경을 사용하여 배포:: 병렬 환경에 변경 사항을 구현한 다음 새 환경으로 이전합니다. 배포가 정상 완료되었음이 확인될 때까지는 이전 환경을 유지합니다. 이렇게 하면 만일의 경우 이전 환경을 롤백할 수 있으므로 복구 시간을 최소화할 수 있습니다.
- 작게 자주 발생하고 되돌릴 수 있는 변경 내용 배포:: 되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 범위가 감소합니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다.
- 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.
- 테스트 및 롤백 자동화:: 배포된 환경의 테스트를 자동화하여 원하는 성과를 달성할 수 있는지를 확인합니다. 원하는 결과를 달성할 수 없는 경우에는 알려진 정상 상태로 롤백하는 과정을 자동화하면 수동 프로세스에서 발생하는 오류를 줄이고 복구 시간을 최소화할 수 있습니다.

OPS 5 워크로드를 지원할 준비가 되었는지 확인하는 방법

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

모범 사례:

- 직원의 역량 확보:: 운영상의 요구 사항에 대해 지원하기 위해 적절한 수의 숙련된 인력이 있는지 확인하는 메커니즘을 확보합니다. 효과적인 지원을 유지하기 위해 필요한 경우 직원을 교육하고 직원의 역량을 조정합니다.
- 일관된 방식으로 운영 준비 검토:: 워크로드를 운영할 준비가 되었는지를 일관된 방식으로 검토합니다. 검토에서는 최소한 팀 및 워크로드의 운영 준비 상태와 보안 고려 사항을 파악해야 합니다. 코드에서 검토 활동을 구현하고 해당하는 경우 이벤트 대응 과정에서 자동화된 검토를 트리거하면 일관성을 유지하고, 실행 속도를 높이고, 수동 프로세스에서 발생하는 오류를 줄일 수 있습니다.
- 실행서(Runbook)를 사용하여 절차 수행:: 실행서는 특정 결과를 달성하기 위한 문서화된 절차입니다. 절차를 실행서로 문서화하면 적절하게 파악한 이벤트에 일관된 방식으로 신속하게 대응할 수 있습니다. 실행서를 코드로 구현하고, 해당하는 경우 이벤트 대응 과정에서 실행서 실행을 트리거하면 일관성을 유지하고, 대응 속도를 높이고, 수동 프로세스에서 발생하는 오류를 줄일 수 있습니다.
- 플레이북을 사용하여 문제 파악:: 플레이북은 문제 조사를 위한 문서화된 프로세스입니다. 플레이북에 조사 프로세스를 문서화하면 장애 발생 시나리오에 일관적이고 빠르게 대응할 수 있습니다. 플레이북을 코드로 구현하고, 해당하는 경우 이벤트 대응 과정에서 플레이북 실행을 트리거하면 일관성을 유지하고, 대응 속도를 높이고, 수동 프로세스에서 발생하는 오류를 줄일 수 있습니다.
- 정보에 입각하여 시스템 및 변경 사항 배포 결정 내리기:: 워크로드를 지원할 수 있는 팀의 능력과 워크로드의 거버넌스 준수 여부를 평가합니다. 배포의 이점을 기준으로 하여 이러한 평가를 수행해 시스템 또는 변경 사항을 프로덕션 환경으로 전환할지 여부를 결정합니다. 이점과 위험을 파악하면 정보에 입각한 결정을 내릴 수 있습니다.

운영

OPS 6 워크로드 상태를 파악하는 방법

워크로드 지표를 정의, 파악 및 분석하면 워크로드 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.

모범 사례:

- 핵심 성과 지표 파악:: 원하는 비즈니스 성과와 고객 결과를 기준으로 하여 핵심 성과 지표(KPI)를 확인합니다. 그리고 KPI를 평가하여 워크로드의 성공 여부를 결정합니다.
- 워크로드 지표 정의:: KPI 달성 여부를 측정하는 데 사용할 워크로드 지표와 워크로드 상태를 측정하는 데 사용할 워크로드 지표를 정의합니다. 그런 다음 해당 지표를 평가해 워크로드에서 적절한 성과를 달성할 수 있는지를 확인하고 워크로드의 상태를 파악합니다.
- 워크로드 지표 수집 및 분석:: 지표를 정기적으로 사전 예방 차원에서 점검하여 추세를 확인하고 어느 부분에 적절한 대응이 필요한지를 파악합니다.
- 워크로드 지표 기준 설정:: 지표의 기준을 설정해 성능이 기준보다 높은/낮은 구성 요소를 확인하고 각 구성 요소의 성능을 비교할 수 있는 기준으로 필요한 값을 제공합니다.
- 워크로드의 예상 활동 패턴 파악:: 워크로드 활동 패턴을 설정하여 워크로드의 동작이 필요한 값의 범위를 벗어나는 경우를 확인합니다. 그러면 필요시에 적절하게 대응을 할 수 있습니다.
- 워크로드 성과가 위험한 상태이면 알림 생성:: 워크로드 성과가 위험한 상태이면 알림을 생성합니다. 그러면 필요시에 적절하게 대응을 할 수 있습니다.
- 워크로드 이상이 감지되면 알림 생성:: 워크로드에서 이상 상태가 감지되면 알림을 생성합니다. 그러면 필요시에 적절하게 대응을 할 수 있습니다.
- 성과 달성 여부와 KPI 및 지표의 효율성 확인: : 워크로드 운영을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지를 확인할 수 있으며 업무 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.

OPS 7 운영 상태를 파악하는 방법

운영 지표를 정의, 파악 및 분석하면 운영 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.

모범 사례:

- 핵심 성과 지표 파악:: 원하는 비즈니스 성과와 고객 결과를 기준으로 하여 핵심 성과 지표(KPI)를 확인합니다. 그리고 KPI를 평가하여 운영의 성공 여부를 결정합니다.
- 운영 지표 정의:: KPI 달성 여부를 측정하는 데 사용할 운영 지표와 운영 상태를 측정하는 데 사용할 운영 지표를 정의합니다. 그런 다음 해당 지표를 평가해 운영 과정에서 적절한 성과를 달성할 수 있는지를 확인하고 운영 상태를 파악합니다.
- 운영 지표 수집 및 분석:: 지표를 정기적으로 사전 예방 차원에서 점검하여 추세를 확인하고 어느 부분에 적절한 대응이 필요한지를 파악합니다.
- 운영 지표 기준 설정:: 지표의 기준을 설정해 성능이 기준보다 높은/낮은 프로세스를 확인하고 각 프로세스의 성능을 비교할 수 있는 기준으로 필요한 값을 제공합니다.
- 운영의 예상 활동 패턴 파악:: 지표의 기준을 설정하여 비교의 기준으로 필요한 값을 제공합니다.
- 운영 성과가 위험한 상태이면 알림 생성:: 운영 성과가 위험한 상태이면 알림을 생성합니다. 그러면 필요시에 적절하게 대응을 할 수 있습니다.
- 운영 이상이 감지되면 알림 생성:: 운영에서 이상 상태가 감지되면 알림을 생성합니다. 그러면 필요시에 적절하게 대응을 할 수 있습니다.
- 성과 달성 여부와 KPI 및 지표의 효율성 확인: : 운영 활동을 실무 수준에서 확인할 수 있는 보기를 생성합니다. 그러면 요구를 충족하고 있는지를 확인할 수 있으며 업무 목표 달성을 위해 개선해야 하는 영역을 파악할 수 있습니다. 또한 KPI와 지표의 효율성을 확인하고 필요한 경우 KPI/지표를 수정합니다.

OPS 8 워크로드 및 운영 이벤트를 관리하는 방법

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

모범 사례:

- 이벤트, 인시던트 및 문제 관리 프로세스 사용:: 관찰되는 이벤트, 개입이 필요한 이벤트(인시던트), 개입이 필요하며 반복되거나 현재 해결할 수 없는 이벤트(문제)를 처리하기 위한 프로세스를 마련합니다. 이러한 프로세스를 사용하면 적시에 적절한 대응을 보장하여 비즈니스와 고객에 대한 해당 이벤트의 영향을 완화할 수 있습니다.
- 근본 원인 분석 프로세스(RCA): 재발을 제한하거나 방지하기 위한 완화책을 개발하고 빠르고 효과적인 대응을 위한 절차를 개발할 수 있도록 이벤트의 근본 원인을 식별하고 문서화하는 프로세스를 마련합니다. 적절한 경우 근본 원인을 알리고 목표 대상에게 맞춤화된 프로세스를 마련합니다.
- 알림별 프로세스 마련:: 경계심을 갖는 이벤트가 있는 경우, 특정하게 식별된 소유자를 지정함과 동시에 명확하게 정의된 대응 방법(실행서 또는 지침서)을 마련합니다. 이렇게 하면 운영 이벤트에 빠르고 효과적으로 대응할 수 있으며 중요하지 않은 알림 때문에 실행 가능한 이벤트를 제대로 확인하지 못하는 상황을 방지할 수 있습니다.
- 비즈니스 영향을 기반으로 운영 이벤트의 우선 순위 지정:: 여러 이벤트에 대해 조치를 취해야 할 때는 실무에 가장 큰 영향을 주는 이벤트를 먼저 해결해야 합니다. 예를 들어 이러한 영향에는 생명 또는 부상, 재정적 손실 또는 평판이나 신뢰의 손상이 포함될 수 있습니다.
- 에스컬레이션 경로 정의:: 에스컬레이션을 트리거하는 요소와 에스컬레이션 절차를 포함한 에스컬레이션 경로를 실행서와 플레이북에 정의합니다. 운영 이벤트에 즉시 효율적으로 대응할 수 있도록 각 작업의 소유자를 구체적으로 명시합니다.
- 푸시 알림 활성화: 사용자가 사용 중인 서비스가 이벤트의 영향을 받을 때와 정상 작동 상태로 되돌아갈 때 사용자에게 이메일이나 SMS 등을 통해 직접 알립니다. 그러면 사용자가 적절한 조치를 취할 수 있습니다.
- 대시보드를 통해 상태 전달:: 목표 대상(예: 내부 기술 팀, 리더십 및 고객)에게 맞춤화된 대시보드를 제공하여 비즈니스의 현재 운영 상태를 전달하고 관심 있는 지표를 제공합니다.
- 이벤트 대응 자동화:: 이벤트 대응을 자동화하면 수동 프로세스에서 발생하는 오류를 줄일 수 있으며 일관된 방식으로 즉시 대응을 할 수 있습니다.

개선

OPS 9 운영을 개선하는 방법

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

모범 사례:

- 지속적인 개선을 위한 프로세스 마련:: 개선 기회를 정기적으로 평가하고 우선 순위를 지정해 가장 큰 이점을 얻을 수 있는 영역에서 작업을 중점적으로 수행합니다.
- 피드백 루프 구현:: 개선이 필요한 영역과 문제를 확인할 수 있도록 절차와 워크로드에 피드백 루프를 포함합니다.
- 개선 추진 요인 정의:: 개선 기회를 평가하고 우선 순위를 지정할 수 있도록 개선 추진 요인을 파악합니다.
- 분석 정보 확인:: 여러 부문의 팀 및 비즈니스 소유자와 함께 분석 결과와 응답을 검토합니다. 이러한 검토에서는 개선 가능성을 공통적으로 파악하고, 추가적인 영향을 확인하고, 조치 과정을 결정할 수 있습니다. 필요에 따라 대응 내용을 조정합니다.
- 운영 지표 검토 수행:: 다양한 실무 영역의 여러 팀 구성원들과 함께 운영 지표 후행 분석을 정기적으로 수행합니다. 이러한 검토에서는 개선 기회와 진행 가능한 조치 과정을 파악하고 파악한 내용을 공유할 수 있습니다.
- 파악한 내용(Lessons Learned) 문서화 및 공유:: 운영 활동 실행 과정에서 파악한 내용을 문서화 및 공유하여 내부적으로, 그리고 여러 팀 간에 사용할 수 있도록 하십시오.
- 개선을 위한 시간 할애:: 프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

보안

자격 증명 및 액세스 관리

SEC 1 자격 증명 및 인증을 관리하는 방법

자격 증명 및 인증 메커니즘에는 워크로드에서 직접적 또는 간접적으로 액세스 권한을 부여하는 암호, 토큰 및 키가 포함됩니다. 우발적 사용 또는 악의적 사용의 위험을 줄일 수 있게 적절한 메커니즘을 사용하여 자격 증명을 보호해야 합니다.

모범 사례:

- 자격 증명 및 액세스 관리 요구 사항 정의:: 조직/법적/규정 준수 요구 사항을 충족할 수 있는 자격 증명 및 액세스 관리 구성을 정의해야 합니다.
- AWS 루트 사용자 보호:: MFA를 적용하고, 액세스 키를 제공하지 않고, AWS 루트 사용자를 사용하는 범위를 제한하는 방식으로 루트 사용자를 보호하면 AWS 계정을 보호할 수 있습니다.
- Multi-Factor Authentication(MFA) 사용 옵션 적용:: 액세스를 추가로 제어하기 위해 소프트웨어 또는 하드웨어 메커니즘을 사용하여 Multi-Factor Authentication(MFA)을 적용합니다.
- 액세스 제어 적용 자동화:: 자동화된 도구를 사용하고 비정상적인 액세스를 보고하여 액세스 제어를 적용합니다. 이렇게 하면 자격 증명 관리 요구 사항을 유지 관리할 수 있습니다.
- 중앙 집중식 연동 공급자 통합:: 연동 자격 증명 공급자 또는 디렉터리 서비스와 통합해 중앙 위치에서 모든 사용자를 인증합니다. 이렇게 하면 여러 자격 증명을 사용해야 하는 경우가 감소하며 복잡한 관리 작업도 줄어듭니다.
- 암호 요구 사항 적용:: 무작위 대입 공격 및 기타 암호 공격으로부터 보호하기 위해 암호의 최소 길이, 복잡성 및 재사용 관련 정책을 적용합니다.
- 정기적인 자격 증명 교체:: 권한이 부여되지 않은 시스템이나 사용자가 이전 자격 증명을 사용할 위험을 줄일 수 있도록 자격 증명을 정기적으로 교체합니다.
- 주기적인 자격 증명 감사:: 자격 증명을 감사해 MFA 등의 정의된 제어 기능이 적용되고 정기적으로 교체되며, 액세스 수준이 적합한지를 확인합니다.

SEC 2 인적 액세스를 제어하는 방법

무단 액세스의 영향과 관련 위험을 줄이려면 정의된 비즈니스 요구 사항에 맞는 제어 기능을 구현하여 인적 액세스를 제어해야 합니다. 이러한 제어는 AWS 계정의 권한 있는 사용자/관리자와 애플리케이션 최종 사용자에게 모두 적용됩니다.

모범 사례:

- 인적 액세스 요구 사항 정의:: 불필요한 권한에서 오는 위험을 줄일 수 있도록 직무 기능을 기준으로 하여 사용자의 액세스 요구 사항을 명확하게 정의합니다.
- 최소 권한 부여:: 무단 액세스 위험을 줄일 수 있도록 정의한 최소 권한만 사용자에게 부여합니다.
- 개인에게 고유한 자격 증명 할당:: 사용자를 구분하고 추적 가능하도록 하기 위하여 사용자 간에 자격 증명을 공유하지 않도록 합니다.
- 사용자 수명 주기를 기준으로 자격 증명 관리:: 사용자 수명 주기와 액세스 관리를 통합합니다. 예를 들어 사용자가 퇴사하거나 사용자의 역할이 변경되면 사용되지 않는/불필요한 자격 증명이 취소되도록 사용자를 폐기합니다.
- 자격 증명 관리 자동화:: 자격 증명 관리를 자동화하여 최소 권한을 적용하고 사용되지 않는 자격 증명을 비활성화합니다. 사용자 수명 주기 감사, 보고, 관리를 자동화합니다.
- 역할 또는 연동을 통한 액세스 권한 부여:: IAM 사용자 또는 정적 액세스 키 대신 IAM 역할을 사용하여 안전한 계정 간 액세스 권한 및 연동 사용자의 액세스를 허용합니다.

SEC 3 프로그래밍 방식 액세스를 제어하는 방법

적절하게 정의/제한/분리된 액세스 권한을 통해 프로그래밍 방식 액세스 또는 자동화된 액세스를 제어하면 무단 액세스 위험을 줄일 수 있습니다. 프로그래밍 방식 액세스에는 워크로드 내부의 액세스 및 AWS 관련 리소스 액세스가 포함됩니다.

모범 사례:

- 프로그래밍 방식 액세스 요구 사항 정의:: 불필요한 권한에서 오는 위험을 줄일 수 있도록 자동화된 액세스 또는 프로그래밍 방식 액세스에 대한 액세스 요구 사항을 명확하게 정의합니다.
- 최소 권한 부여:: 무단 액세스 위험을 줄일 수 있도록 자동화된 액세스 또는 프로그래밍 방식 액세스에 대해 정의한 최소 권한만 부여합니다.
- 자격 증명 관리 자동화:: 자격 증명 관리를 자동화하여 최소 권한을 적용하고 사용되지 않는 자격 증명을 비활성화합니다. 또한 동적 인증 감사, 보고 및 관리를 자동화합니다.
- 각 구성 요소에 고유한 자격 증명 할당:: 사용자를 구분하고 추적할 수 있도록 구성 요소 간에 자격 증명을 공유하지 않도록 합니다. 예를 들어 AWS Lambda 함수와 EC2 인스턴스에 각기 다른 IAM 역할을 사용합니다.
- 역할 또는 연동을 통한 액세스 권한 부여:: IAM 사용자 또는 정적 액세스 키 대신 IAM 역할 또는 연동을 사용하여 안전한 프로그래밍 방식 액세스를 허용합니다.
- 동적 인증 구현:: 서버나 시스템에서 자격 증명을 동적으로 가져오며 자주 교체됩니다.

탐지 제어

SEC 4 보안 이벤트를 감지하고 조사하는 방법

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

모범 사례:

- 로그의 요구 사항 정의:: 조직/법률/규정 준수 요구 사항을 충족할 수 있도록 로그 보존 및 액세스 제어를 위한 요구 사항을 정의합니다.
- 지표의 요구 사항 정의:: 지표를 수집하고 기준을 정의하면 잠재적 보안 위협 관련 정보를 파악할 수 있습니다.
- 알림의 요구 사항 정의:: 알림을 수신해야 하는 사람과 수신자가 수신한 알림에 대해 수행해야 하는 작업을 정의합니다.
- 서비스 및 애플리케이션 로깅 구성:: 워크로드 전반에 걸쳐 애플리케이션 로그, AWS 서비스 로그 및 리소스 로그를 포함한 로깅을 구성합니다.
- 중앙에서 로그 분석:: 모든 로그를 중앙 위치에 수집한 다음 자동으로 분석하여 악의적인 활동이나 보안 침해를 나타내는 정보나 이상 상태를 감지합니다.
- 주요 지표 관련 알림 자동화:: 보안 관련 지표와 이벤트를 포함한 주요 지표를 모니터링하고 임계값을 기준으로 자동화된 알림을 트리거해야 합니다.
- 조사 프로세스 개발:: 인시던트 대응 프로세스의 에스컬레이션 경로를 비롯하여 여러 이벤트 유형을 조사하는 프로세스를 개발합니다.

SEC 5 새로운 보안 위협을 방어하는 방법

AWS 및 업계의 최신 모범 사례와 위협 관련 정보를 지속적으로 확인하면 새로운 위험을 파악할 수 있습니다. 그러면 워크로드를 보호하는 데 적합한 제어 기능을 확인하여 우선 순위를 지정하고 구현하는 보안 위협 모델을 생성할 수 있습니다.

모범 사례:

- 조직/법률/규정 최신 준수 요구 사항 파악:: 보안 태세를 조정할 때 참조할 수 있는 조직/법률/규정 최신 준수 요구 사항을 파악합니다.
- 최신 보안 모범 사례 파악:: AWS 및 업계의 보안 관련 최신 모범 사례를 파악하면 워크로드를 더욱 효율적으로 보호할 수 있습니다.
- 최신 보안 위협 정보 파악:: 최신 보안 위협 정보를 확인하여 공격 벡터를 파악합니다. 이렇게 하면 감지 및 예방 제어 기능을 구현할 수 있습니다.
- 정기적으로 새 보안 서비스 및 기능 평가:: 위협의 위험을 제한하는 새 기능을 비롯하여 AWS 및 APN 파트너가 제공하는 보안 서비스를 평가합니다.
- 보안 위협 모델을 사용하여 위험 정의 및 우선 순위 지정:: 보안 위협 모델을 사용하여 가장 최근에 등록된 잠재적 위협을 파악하고 유지 관리합니다. 또한 위협 우선 순위를 지정하고 대응을 위한 보안 태세를 조정합니다.
- 새로운 보안 서비스 및 기능 구현:: 워크로드를 보호하는 데 사용할 수 있는 제어를 구현하는 보안 서비스와 기능을 도입합니다.

인프라 보호

SEC 6 네트워크 보호 방법

퍼블릭 및 프라이빗 네트워크에는 내/외부 네트워크 기반 위협으로부터 네트워크를 보호할 수 있는 다계층 방어가 필요합니다.

모범 사례:

- 네트워크 보호 요구 사항 정의:: 조직/법적/규정 준수 요구 사항을 충족할 수 있는 네트워크 보호용 제어를 정의합니다.
- 노출 제한:: 필요한 최소 액세스 권한만 허용하는 방식을 통해 내부 네트워크와 인터넷에 대한 워크로드 노출을 제한합니다.
- 구성 관리 자동화:: 인적 오류를 줄이기 위해 구성 관리 서비스 또는 도구를 사용하여 보안 구성을 자동으로 적용하고 확인합니다.
- 네트워크 보호 자동화:: 위협 정보 및 이상 상태 감지 결과에 따라 자체 방어 네트워크를 제공하는 보호 메커니즘을 자동화합니다.
- 검사 및 보호 구현:: 위협으로부터 보호하기 위해 웹 애플리케이션 방화벽 등을 사용하여 애플리케이션 수준에서 트래픽을 검사하고 필터링합니다.
- 모든 계층에서 트래픽 제어:: 수신 및 송신 트래픽을 모두 제어하는 기능(데이터 손실 방지 포함)을 적용합니다. Amazon Virtual Private Cloud(VPC)의 경우 이러한 기능에는 보안 그룹, 네트워크 ACL, 서브넷 등이 포함됩니다. AWS Lambda는 트래픽 제어를 위해 프라이빗 VPC에서 실행하는 것이 좋습니다.

SEC 7 컴퓨팅 리소스를 보호하는 방법

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

모범 사례:

- 컴퓨팅 보호 요구 사항 정의:: 조직/법적/규정 준수 요구 사항을 충족할 수 있는 컴퓨팅 리소스용 보호 제어 기능을 정의합니다.
- 취약성 검사 및 패치:: 코드베이스와 인프라의 취약성을 자주 검색하고 패치 하여 새 위협으로부터 워크로드를 보호합니다.
- 구성 관리 자동화:: 인적 오류를 줄이기 위해 구성 관리 서비스 또는 도구를 사용하여 보안 구성을 자동으로 적용하고 확인합니다.
- 컴퓨팅 보호 자동화:: 알 수 없는 위협으로부터 워크로드를 보호하는 기능(예: 가상 패치 사용)을 포함한 침입 방지 기능을 통해 방어를 자동화합니다.
- 공격 대상 영역 축소:: EC2 운영 체제 강화, 컨테이너 및 서버리스 리소스 구성 등의 작업을 통해 공격 대상 영역을 축소하여 노출을 제한합니다.
- 관리형 서비스 구현:: 보안 유지 관리 작업을 줄일 수 있도록 Amazon RDS, AWS Lambda, Amazon ECS 등 리소스를 관리하는 서비스를 구현합니다.

데이터 보호

SEC 8 데이터 분류 방법

분류는 중요도 수준을 기반으로 데이터를 분류하는 방법을 제공하여 적절한 보호 및 보존 제어를 결정하도록 합니다.

모범 사례:

- 데이터 분류 요구 사항 정의:: 조직/법적/규정 준수 요구 사항을 충족할 수 있도록 데이터 분류 요구 사항을 정의합니다.
- 데이터 보호 제어 정의:: 분류 수준에 따라 데이터를 보호합니다. 예를 들어 중요한 데이터는 추가 제어 기능을 사용해 보호하고 공개적으로 액세스 가능한 데이터는 모범 사례를 사용해 보호합니다.
- 데이터 식별 기능 구현:: 쉽게 식별 가능한 지표를 사용해 데이터를 분류합니다. 예를 들어 Amazon S3 버킷 및 객체에는 버킷의 데이터를 분류하는 태그를 사용합니다.
- 식별 및 분류 자동화:: 인적 오류의 위험을 줄이기 위해 데이터 식별 및 분류 작업을 자동화합니다.
- 데이터 유형 파악:: 조직/법적/규정 준수 요구 사항을 충족하는 제어 기능을 구현할 수 있도록 워크로드의 데이터 유형을 파악합니다.

SEC 9 저장된 데이터 보호 방법

무단 액세스 또는 손실 위험을 줄이기 위해 암호화를 비롯한 제어를 구현하고 요구 사항을 정의하여 저장된 데이터를 보호합니다.

모범 사례:

- 저장된 데이터 관리 및 보호 요구 사항 정의:: 조직/법률/규정 준수 요구 사항을 충족할 수 있도록 암호화 및 데이터 보존 등의 저장된 데이터 관리 및 보호 요구 사항을 정의합니다.
- 보안 키 관리 구현:: 암호화 키는 안전하게 저장해야 하며 엄격하게 액세스를 제어하면서 교체해야 합니다. 예를 들어 AWS Key Management Service와 같은 키 관리 서비스를 사용할 수 있습니다. 각 데이터 분류 수준 및 보존 요구 사항을 구분할 수 있도록 여러 키를 사용할 수 있습니다.
- 저장 데이터 암호화 적용:: 저장된 데이터를 보호하기 위해 최신 표준 및 모범 사례에 따라 정의된 암호화 요구 사항을 적용합니다.
- 액세스 제어 적용:: 저장된 데이터를 보호할 수 있도록 백업, 격리, 버전 관리 등의 메커니즘과 최소 권한을 사용하여 액세스 제어를 적용합니다. 그리고 공개적으로 액세스 가능하도록 설정할 데이터를 고려합니다.
- 사람들이 데이터에 쉽게 액세스할 수 없도록 유지하는 메커니즘 제공:: 모든 중요한 데이터에 직접 액세스할 수 없도록 유지합니다. 예를 들어 데이터 스토어에 직접 액세스를 허용하는 대신 대시보드를 제공하고, 데이터를 간접적으로 관리할 수 있는 도구를 제공합니다.

SEC 10 전송 중인 데이터 보호 방법

무단 액세스 또는 노출 위험을 줄이기 위해 암호화를 비롯한 제어를 구현하고 요구 사항을 정의하여 전송 중인 데이터를 보호합니다.

모범 사례:

- 전송 중인 데이터 보호 요구 사항 정의:: 조직/법률/규정 준수 요구 사항을 충족할 수 있도록 데이터 분류에 따라 암호화 표준 등의 전송 중인 데이터 보호 요구 사항을 정의합니다. 모든 트래픽을 암호화/인증하고 최신 표준과 암호화를 적용하는 것이 모범 사례입니다.
- 보안 키 및 인증서 관리 구현:: 암호화 키와 인증서를 안전하게 저장하고 엄격하게 액세스를 제어하면서 교체해야 합니다. 예를 들어 AWS Certificate Manager와 같은 인증서 관리 서비스를 사용할 수 있습니다.
- 전송 중 데이터 암호화 적용: 조직/법률/규정 준수 요구 사항을 충족하기 위해 최신 표준 및 모범 사례에 따라 정의된 암호화 요구 사항을 적용합니다.
- 데이터 유출 감지 자동화:: 도구 또는 감지 메커니즘을 사용하여 정의된 경계 외부로 데이터를 이동하려는 시도를 직접 감지합니다. 예를 들어 알 수 없는 호스트로 데이터를 복사하는 데이터베이스 시스템을 감지할 수 있습니다.
- 네트워크 통신 인증:: 데이터 변조나 손실 위험을 줄일 수 있도록 TLS(전송 계층 보안) 또는 IPsec 등의 프로토콜을 사용해 통신 자격 증명을 확인합니다.

인시던트 대응

SEC 11 인시던트에 대응하는 방법

조직의 업무 중단 가능성을 최소화할 수 있도록 보안 인시던트를 제때 조사하고 대응하려면 철저한 준비가 필요합니다.

모범 사례:

- 주요 직원과 외부 리소스 파악:: 조직의 인시던트 대응을 지원할 수 있는 내/외부 직원과 리소스를 파악합니다.
- 도구 파악:: 조직의 인시던트 대응을 지원할 수 있는 AWS, 파트너 및 오픈 소스 도구를 파악합니다.
- 인시던트 대응 계획 개발:: 워크로드와 조직에 발생할 가능성이 가장 높은 시나리오부터 시작하여 인시던트 대응 계획을 생성합니다. 내/외부의 통신 및 에스컬레이션 방법도 포함해야 합니다.
- 억제 기능 자동화:: 대응 시간과 조직에 대한 영향을 줄일 수 있도록 인시던트 억제를 자동화합니다.
- 포렌식 기능 파악:: 외부 전문가를 비롯하여 사용 가능한 포렌식 조사 기능을 파악합니다.
- 액세스 권한 사전 프로비저닝:: 인시던트에 적절하게 대응할 수 있도록 보안 담당자가 올바른 액세스 권한을 AWS에 사전 프로비저닝하도록 합니다.
- 도구 사전 배포:: 인시던트에 적절하게 대응할 수 있도록 보안 담당자가 적절한 도구를 AWS에 사전 배포하도록 합니다.
- 게임 데이 실행:: 인시던트 대응 게임 데이(시뮬레이션)를 정기적으로 연습하고, 파악한 내용을 계획에 통합하고, 대응 방식과 계획을 지속적으로 개선합니다.

안정성

기반

REL 1 서비스 한도를 관리하는 방법

기본 서비스 한도는 사용자가 필요 이상으로 많은 리소스를 실수로 프로비저닝하는 것을 방지합니다. 또한 서비스 악용으로부터 보호하기 위해 얼마나 자주 API 작업을 호출 할 수 있는지에 대한 한도가 있습니다. AWS Direct Connect를 사용하는 경우 각 연결에서 전송할 수 있는 데이터 양에 한도가 적용됩니다. AWS Marketplace 애플리케이션을 사용하는 경우에는 애플리케이션의 제한을 파악해야 합니다. 타사 웹 서비스 또는 SaaS(Software-as-a-Service)를 사용하는 경우에는 해당 서비스의 한도도 파악해야 합니다.

모범 사례:

- 한도를 인지하되 추적하지는 않음:: 한도가 있다는 것은 알지만 한도를 추적하고 있지는 않습니다.
- 한도 모니터링 및 관리:: 잠재 사용량을 평가하고, 리전별 한도를 적절히 증가한 후 계획에 따른 사용량 증가분이 반영되도록 하십시오.
- 한도 자동 모니터링 및 관리 사용:: 임계값에 근접했을 때 알림을 받을 수 있는 도구를 구현합니다. 그러면 한도 증가 요청을 자동화할 수 있을 때까지 배포 메커니즘을 사용해 담당 그룹에 알림을 보낼 수 있습니다.
- 아키텍처를 통해 고정된 서비스 한도 수용:: 변경할 수 없는 서비스 한도 및 이러한 문제를 해결하는 아키텍쳐를 알고 있어야 합니다.
- 장애 조치를 수용할 수 있도록 현재의 서비스 한도와 최대 사용치 간의 여유가 충분한지 확인합니다.: 리소스는 장애가 발생하더라도 정상적으로 종료할 때까지는 한도 계산에 계속 포함될 수 있습니다. 그러므로 장애 발생 리소스가 종료되기 전까지를 수용할 수 있는 모든 장애 발생 리소스와 교체용 리소스를 합산한 한도를 적용해야 합니다. 이 차이를 계산할 때는 가용 영역 장애를 고려해야 합니다.
- 모든 관련 계정 및 리전에서 서비스 한도 관리:: AWS 계정이나 AWS 리전을 여러 개 사용하는 경우 프로덕션 워크로드를 실행하는 모든 환경이 동일한 한도를 가지도록 요청해야 합니다.

REL 2 네트워크 토폴로지를 관리하는 방법

애플리케이션은 기존 데이터 센터 인프라, 공개적으로 액세스 가능한 퍼블릭 클라우드 인프라, 비공개 주소가 지정된 퍼블릭 클라우드 인프라 등 하나 이상의 환경에 있을 수 있습니다. 시스템 내/시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 주소 관리, 이름 확인과 같은 네트워크 고려 사항은 클라우드의 리소스를 활용하기 위한 기본 사항입니다.

모범 사례:

- 온프레미스 환경과 퍼블릭 클라우드의 프라이빗 주소 간에 고가용성 연결 사용:: 별도로 배포된 프라이빗 IP 주소 공간 간에 AWS Direct Connect(DX) 회선 및 여러 VPN 터널을 사용합니다. 가용성을 높이려는 경우에는 여러 DX 위치를 사용합니다. 여러 AWS 리전을 사용하는 경우 2개 이상의 리전에서 여러 DX 위치가 필요할 수도 있습니다. VPN을 위한 AWS Marketplace 어플라이언스를 평가해야 할 수 있습니다. AWS Marketplace 어플라이언스를 사용하는 경우 다른 가용 영역에서 고가용성을 위해 중복 인스턴스를 배포합니다.
- 워크로드 사용자를 위한 고가용성 네트워크 연결 사용:: 고가용성 DNS, CloudFront, API 게이트웨이, 로드 밸런싱 및 역방향 프록시를 애플리케이션의 퍼블릭 엔드포인트로 사용합니다. 로드 밸런싱 또는 프록시를 위해 AWS Marketplace 어플라이언스를 평가해야 할 수 있습니다.
- 연결된 여러 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위 적용:: VPN을 통해 피어링되거나 연결된 경우 각 VPC의 IP 범위가 충돌하지 않아야 합니다. 온프레미스 환경 및 기타 클라우드 공급자에 개인 프라이빗 연결에 대해서도 동일합니다. 필요한 경우 프라이빗 IP 범위를 할당할 수 있어야 합니다.
- 확장 및 가용성을 위한 IP 서브넷 할당 계정 확인:: 개별 Amazon VPC IP 주소 범위는 가용 영역의 서브넷에 IP 주소를 할당하고 추후 확장을 고려하는 등 애플리케이션의 요구 사항을 수용할 수 있도록 충분히 커야 합니다. 여기에는 로드 밸런서, AWS Lambda 함수, EC2 인스턴스 및 컨테이너 기반 애플리케이션이 포함됩니다. 또한 향후 확장을 위해 일부 IP 주소를 사용할 수 있도록 보전해야 합니다.

변경 관리

REL 3 시스템이 수요 변화에 어떻게 대응하고 있습니까?

확장 가능형 시스템에는 리소스를 자동으로 유연하게 추가하거나 제거할 수 있어 특정 시점의 요구에 근접하게 대응합니다.

모범 사례:

- 워크로드를 확장 또는 축소할 때 리소스 자동 조달:: Amazon S3, Amazon CloudFront, Amazon Auto Scaling, AWS Lambda 등의 자동 확장되는 서비스를 사용합니다. 타사 도구 및 AWS SDK를 사용하여 크기 조정을 자동화할 수도 있습니다.
- 워크로드 내에서 서비스 부족이 감지되면 리소스 조달:: 가용성에 영향이 있는 경우 리소스 확장을 수동으로 수행합니다.
- 워크로드에 조만간 더 많은 리소스가 필요해질 가능성이 감지되면 리소스를 수동으로 조달합니다.:: 필요에 따라 연산 및 저장 기능을 수동으로 확장합니다.
- 워크로드 로드 테스트:: 확장 작업이 워크로드의 필요 사항을 충족시키는지를 측정하기 위한 로드 테스트 방식을 채택하십시오.

REL 4 리소스를 모니터링하는 방법

로그와 지표는 워크로드의 상태를 파악할 수 있는 유용한 도구입니다. 로그 및 지표를 모니터링하여 임계값을 초과하거나 중요한 이벤트가 발생하면 알림을 보내도록 워크로드를 구성할 수 있습니다. 낮은 성능 임계값을 초과하거나 장애가 발생할 때는 워크로드가 그에 대응하여 자동으로 자체 복구되거나 확장되도록 설계하는 것이 가장 좋습니다.

모범 사례:

- 모든 계층에서 워크로드 모니터링:: Amazon CloudWatch 또는 타사 도구를 사용하여 워크로드의 모든 단계를 모니터링하십시오. Personal Health Dashboard로 AWS 서비스를 모니터링하십시오.
- 모니터링 결과를 기반으로 알림 전송:: 중요한 이벤트가 발생할 경우 이를 알아야 할 조직으로 알림이 전달됩니다.
- 이벤트 자동 대응 수행:: 이벤트가 감지되면 자동으로 실패한 구성 요소를 교체하는 등의 조치를 취합니다.
- 정기 검토 진행:: 중요한 이벤트 및 변경 사항을 중심으로 워크로드 모니터링 결과를 자주 검토하여 아키텍처 및 구현을 확인합니다.

REL 5 변경 사항을 어떻게 수행하십니까?

환경에 제어되지 않는 변경 사항이 생기면 그 영향을 예측하기가 어렵습니다. 워크로드와 운영 환경에서 확인된 소프트웨어를 실행하고 예측 가능한 방법으로 패치를 적용하거며 예측가능한 방식으로 교체할 수 있도록 하기 위하여 프로비저닝된 리소스와 워크로드에 대한 변경 사항은 제어되어야 합니다.

모범 사례:

- 계획된 방식으로 변경 사항 배포:: 배포 및 패치 적용 과정에서 문서화된 프로세스를 따릅니다.
- 자동화 기능을 사용하여 변경 사항 배포:: 배포 활동 및 패치가 자동으로 수행됩니다.

장애 관리

REL 6 데이터는 어떻게 백업합니까?

데이터, 애플리케이션 및 운영 환경(애플리케이션이 구성된 운영 체제)을 백업하여 평균 복구 시간(MTTR) 및 목표 복구 시점(RPO) 요구 사항을 충족하십시오.

모범 사례:

- 백업해야 하는 모든 데이터를 확인하고 백업을 수행하거나 원본에서 데이터 재현:: Amazon S3, Amazon EBS 스냅샷 또는 타사 소프트웨어를 사용하여 중요한 데이터를 백업합니다. RPO 충족을 위해 원본에서 데이터를 재현할 수 있는 경우에는 백업이 필요하지 않을 수도 있습니다.
- 자동으로 데이터 백업을 수행하거나 원본에서 자동으로 데이터 재현:: AWS 기능(예: Amazon RDS 및 Amazon EBS의 스냅샷, Amazon S3의 버전 등), AWS Marketplace 솔루션 또는 타사 솔루션을 사용하여 백업 또는 원본으로부터의 재현을 자동화합니다.
- 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행:: 복구 테스트를 통해 백업 프로세스 구현이 목표 복구 시간(RTO) 및 목표 복구 시점(RPO)을 충족하는지 확인합니다.
- 백업을 안전하게 보호하고 암호화하거나, 재현에 필요한 데이터의 경우 안전한 공간에 보관되고 이용 가능한지 확인:: AWS IAM과 같은 인증 및 권한 부여 서비스를 통해 액세스를 감지하고, 암호화를 사용하여 데이터 무결성 위반을 감지합니다.

REL 7 구성 요소 장애 시 시스템에서 대처하는 방법

워크로드에 명시적으로든 암시적으로든 고가용성과 낮은 평균 복구 시간(MTTR)에 대한 요구 사항이 있는 경우 복원성을 갖도록 워크로드를 설계하고 이러한 워크로드를 배포하여 중단에 대처하십시오.

모범 사례:

- 워크로드의 모든 계층을 모니터링하여 장애 감지:: 시스템의 상태를 지속적으로 모니터링하고 성능 저하 및 전체 장애를 보고합니다.
- 약결합(Loosely coupled) 종속성 구현:: 대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 약결합(Loosely coupled)됩니다.
- 해당하는 하드 종속성을 소프트 종속성으로 변환하는 정상적인 성능 저하 구현:: 구성 요소의 종속성이 비정상 상태이더라도 구성 요소 자체는 비정상으로 보고되지 않습니다. 계속해서 제한된 방식으로 요청을 처리할 수 있습니다.
- 워크로드의 전체 또는 일부분에 단일 위치가 필요한 기술 제약 조건이 있는 경우 전체 복구 자동화:: 워크로드의 요소는 하나의 가용 영역이나 데이터 센터에서만 실행할 수 있으므로 복구 목표를 정의하여 워크로드의 전체 재구축을 구현해야 합니다.
- 여러 위치로 워크로드 배포:: 여러 가용 영역 및 AWS 리전(예: DNS, ELB, Application Load Balancer 및 API Gateway)에 워크로드를 분산합니다. 필요에 따라 다양한 위치를 사용할 수 있습니다.
- 모든 계층에서 복구 자동화:: 장애를 감지하면 자동 기능을 사용하여 개선 작업을 수행합니다.
- 가용성에 영향을 주는 이벤트 발생 시 알림 전송:: 중요한 이벤트가 감지되면 문제가 자동으로 복구되었더라도 알림이 전송됩니다.

REL 8 복원성은 어떻게 테스트하고 있습니까?

워크로드의 복원성을 테스트하면 프로덕션 환경에서만 나타나는 잠재적인 버그를 찾는 데 도움이 됩니다. 이러한 테스트를 정기적으로 수행하십시오.

모범 사례:

- 예기치 않은 장애 발생 시 플레이북 사용:: 예기치 않은 장애 시나리오에서는 플레이북을 사용하여 근본 원인을 파악하고 해당 장애를 방지하거나 해결하기 위한 전략 결정을 지원할 수 있습니다.
- 근본 원인 분석(RCA) 수행 및 결과 공유:: 중요한 이벤트를 기반으로 시스템 오류를 검토하여 아키텍처를 평가하고 근본 원인을 식별합니다. 필요한 경우 다른 관계자들에게 이러한 원인을 전달하는 방법을 마련합니다.
- 오류를 삽입하여 복원력 테스트:: 장애를 정기적으로 테스트하여 장애 경로의 커버리지를 확인합니다.
- 정기적으로 게임 데이 진행:: 게임 데이를 통해 실제 장애 시나리오에 참여할 직원들과 장애 절차를 정기적으로 연습합니다.

REL 9 재해 복구를 계획하는 방법

데이터를 백업 방식으로 복원해야 하는 경우 재해 복구(DR)가 필수적입니다. 이 데이터의 목표, 리소스, 위치 및 기능은 RTO 및 RPO 목표와 일치하도록 정의하고 실행해야 합니다.

모범 사례:

- 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의:: 워크로드에는 복구 시간 목표(RTO) 및 복구 시점 목표(RPO)가 있습니다.
- 복구 목표 달성을 위해 정의된 복구 전략 사용:: 목표를 달성하기 위한 재해 복구(DR) 전략을 정의해야 합니다.
- 재해 복구 구현을 테스트하여 구현 확인:: DR에 대한 장애 조치를 정기적으로 테스트하여 RTO와 RPO를 충족하는지 확인합니다.
- 모든 변경 수행 시 구성 변화 관리:: AMI 및 시스템 구성 상태가 DR 사이트 또는 리전에서 최신 상태로 유지되는지 여부 및 AWS 서비스의 제한 사항을 확인합니다.
- 복구 자동화:: AWS 또는 타사 도구를 사용하여 시스템 복구를 자동화합니다.

성능 효율성

선택

PERF 1 최고의 성능을 발휘하는 아키텍처를 선택하기 위하여 어떤 방식을 사용하십니까?

워크로드에서 최적의 성능을 얻으려면 여러 가지 방식이 필요합니다. 설계가 잘된 시스템은 여러 개의 솔루션을 사용하고 다양한 특성을 사용하여 성능을 높입니다.

모범 사례:

- 사용 가능한 서비스 및 리소스 파악:: AWS에서 사용 가능한 폭넓은 서비스와 리소스에 대한 정보를 파악합니다. 워크로드와 관련이 있는 서비스 및 구성 옵션을 확인하고, 이러한 옵션을 사용해 성능을 최적화할 수 있는 방법을 파악합니다.
- 아키텍처를 선택하는 프로세스 정의:: AWS와 관련된 내부의 기존 경험과 지식을 사용하거나 게시된 사용 사례, 관련 설명서 또는 백서 등의 외부 리소스를 사용하여 리소스 및 서비스를 선택하는 프로세스를 정의합니다. 워크로드에서 사용될 수 있는 여러 서비스의 실험과 벤치마킹을 수행해 볼 수 있는 프로세스를 예로 들 수 있습니다.
- 결정 과정에서 비용 또는 예산 고려:: 워크로드에는 운영 시에 사용 가능한 예산이 책정되는 경우가 많으며, 예산은 효율적인 운영을 위한 중요 요소입니다. 예상 리소스 요구량을 기준으로 리소스 유형과 크기를 선택할 때는 내부 비용 제어 기능을 사용하고 예산을 고려해야 합니다.
- 정책 또는 참조 아키텍처 사용:: 내부 정책 또는 기존 참조 아키텍처를 사용하여 워크로드에 가장 적합한 아키텍처를 선택합니다. 워크로드에 가장 적절한 서비스와 구성을 평가하면 성능과 효율성을 극대화할 수 있습니다.
- AWS 또는 APN 파트너의 지침 사용:: 결정 과정에서 솔루션스 아키텍트 등의 AWS 리소스나 APN 파트너를 활용할 수 있습니다. 이러한 리소스를 활용하면 아키텍처를 검토하고 제안 개선 사항을 파악할 수 있으므로 성능 수준을 최적화할 수 있습니다.
- 기존 워크로드 벤치마크:: 기존 워크로드의 성능 벤치마크를 수행하여 AWS에서 워크로드의 성능을 파악합니다. 이러한 벤치마크에서 수집된 데이터를 사용하면 아키텍처를 원활하게 결정할 수 있습니다.
- 워크로드 부하 테스트:: 다양한 리소스 유형 및 크기를 사용하여 AWS에 최신 버전의 시스템을 배포하고, 모니터링 기능을 사용하여 병목 현상이나 초과 용량을 파악할 수 있는 성능 지표를 파악합니다. 성능을 기준으로 하여 선택한 아키텍처 및 리소스를 설계하거나 개선할 때 이 정보를 사용합니다.

PERF 2 컴퓨팅 솔루션 선택 방법

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

모범 사례:

- 사용 가능한 컴퓨팅 옵션 평가:: 사용 가능한 컴퓨팅 관련 옵션의 성능 특성을 확인하고 파악합니다. 인스턴스, 컨테이너 및 함수의 작동 방식과 이러한 컴퓨팅 기능이 워크로드에 제공하는 장단점을 확인합니다.
- 사용 가능한 컴퓨팅 구성 옵션 파악:: 다양한 옵션을 통해 워크로드를 보완할 수 있는 방식과 시스템에 가장 적합한 구성 옵션을 파악합니다. 이러한 옵션의 예로는 인스턴스 패밀리, 크기, 기능(GPU, I/O), 함수 크기, 컨테이너 인스턴스, 단일/다중 테넌시 등이 있습니다.
- 컴퓨팅 관련 지표 수집:: 시스템 성능을 가장 효율적으로 파악하는 방법 중 하나는 다양한 리소스의 실제 사용률을 기록하고 추적하는 것입니다. 이 데이터를 시스템에 다시 공급하면 리소스 요구 사항을 더 정확하게 확인할 수 있습니다.
- 적절하게 크기를 조정하여 필요한 구성 확인:: 워크로드의 다양한 성능 특성, 그리고 이러한 특성과 메모리/네트워크/CPU 사용량 간의 관계를 분석합니다. 워크로드 프로필에 가장 적합한 리소스를 선택할 때 이 데이터를 사용할 수 있습니다. 예를 들어 데이터베이스 등의 메모리를 많이 사용하는 워크로드는 r 인스턴스 패밀리를 사용하면 가장 효율적으로 처리할 수 있습니다. 반면 버스팅 워크로드의 경우 Amazon Elastic Container Service 등의 탄력적 컨테이너 시스템을 사용하면 더 효율적일 수 있습니다.
- 사용 가능한 리소스 탄력성 사용:: AWS에서는 변화하는 수요를 충족할 수 있도록 AWS Auto Scaling, Amazon Elastic Container Service, AWS Lambda 등의 다양한 메커니즘을 통해 리소스를 유연하게 동적으로 확장하거나 축소할 수 있습니다. 컴퓨팅 관련 지표를 함께 활용하는 경우 워크로드가 이러한 변화에 자동으로 대응하여 목표를 달성하는 데 가장 적합한 리소스 세트를 활용할 수 있습니다.
- 지표를 기준으로 컴퓨팅 요구 재평가:: 시스템 수준 지표를 사용하여 시간별 워크로드 동작 및 요구 사항을 파악합니다. 사용 가능한 리소스를 이러한 요구 사항과 비교해 워크로드 요구를 평가한 다음, 워크로드 프로필에 가장 적합하도록 컴퓨팅 환경을 변경합니다. 예를 들어 시스템을 계속 사용하다 보면 초기 예상보다 메모리가 더 많이 사용될 수 있으므로 다른 인스턴스 패밀리나 크기로 전환하면 성능과 효율성이 모두 개선될 수 있습니다.

PERF 3 스토리지 솔루션 선택 방법

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

모범 사례:

- 스토리지 특성 및 요구 사항 파악:: Amazon S3, Amazon EBS, Amazon Elastic File System(Amazon EFS), Amazon EC2 인스턴스 스토어 등 워크로드에 가장 적합한 서비스를 선택하는 데 필요한 다양한 특성(예: 공유 가능 여부, 파일 크기, 캐시 크기, 액세스 패턴, 지연 시간, 처리량, 데이터 지속성)을 파악합니다.
- 사용 가능한 구성 옵션 평가:: 다양한 특성과 구성 옵션, 그리고 이러한 특성/구성과 스토리지의 관계를 평가합니다. PIOPS, SSD, 마그네틱 스토리지, Amazon S3, Amazon Glacier 또는 휘발성 스토리지를 사용하는 위치와 방법을 파악하여 워크로드의 성능과 스토리지 공간을 최적화합니다.
- 액세스 패턴과 지표를 기준으로 결정 내리기:: 워크로드가 데이터에 액세스하는 방식을 고려하여 스토리지 시스템을 선택하고 구성합니다. 액세스 패턴에 가장 적합한 캐싱 서비스 또는 인스턴스를 선택하거나, Amazon S3 또는 DynamoDB에 데이터를 저장할 때 최적의 키 분산 방식을 활용하거나, 스토리지 볼륨을 스트라이핑하거나, 시스템 측정값에 따라 데이터를 분할하는 등의 방식을 통해 성능을 개선합니다. Amazon S3 등의 객체 스토리지 또는 Amazon Elastic Block Store 등의 블록 스토리지를 선택하여 스토리지 효율성을 높입니다. 선택한 스토리지 옵션을 데이터 액세스 패턴과 일치하도록 구성합니다.

PERF 4 데이터베이스 솔루션 선택 방법

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

모범 사례:

- 데이터 특성 파악:: 워크로드에 포함된 데이터의 다양한 특성을 파악합니다. 예를 들어 워크로드에 트랜잭션이 필요한지 여부, 워크로드가 데이터와 상호 작용하는 방식, 성능 요구 사항 등을 확인할 수 있습니다. 이 데이터를 사용하여 가장 우수한 성능을 제공하는 워크로드용 데이터베이스 방식(예: 관계형 데이터베이스, NoSQL, 데이터 웨어하우스, 인 메모리 스토리지)을 선택합니다.
- 사용 가능한 옵션 평가:: 워크로드 스토리지 메커니즘 선택 프로세스의 일환으로 사용 가능한 서비스 및 스토리지 옵션을 평가합니다. 그리고 데이터 저장용으로 지정된 서비스나 시스템을 사용하는 방법과 시기를 파악합니다. 또한 PIOP, 메모리/컴퓨팅 리소스, 캐싱 등 데이터베이스 성능이나 효율성을 더욱 최적화할 수 있는 사용 가능한 구성 옵션을 확인합니다.
- 데이터베이스 성능 지표 수집 및 기록:: 데이터베이스 성능과 관련된 성능 측정값을 기록하는 도구, 라이브러리 및 시스템을 사용합니다. 예를 들어 초당 트랜잭션 수, 속도가 느린 쿼리 또는 데이터베이스 액세스 시에 발생하는 시스템 지연 시간을 측정할 수 있습니다. 이 데이터를 사용하면 데이터베이스 시스템의 성능을 파악할 수 있습니다.
- 액세스 패턴을 기준으로 데이터 스토리지 선택:: 워크로드의 액세스 패턴을 기준으로 하여 사용할 서비스와 기술을 결정합니다. 예를 들어 해당하는 경우에는 더 높은 처리량을 제공하지만 최종 처리량은 일정한 키-값 저장소 또는 트랜잭션을 수행해야 하는 워크로드에는 관계형 데이터베이스를 활용할 수 있습니다.
- 액세스 패턴 및 지표를 기준으로 데이터 스토리지 최적화:: 성능을 최대한 높이려면 데이터 저장 또는 쿼리 방식을 최적화하는 성능 특성과 액세스 패턴을 사용합니다. 인덱싱, 키 분산, 데이터 웨어하우스 설계, 캐싱 전략 등의 최적화가 시스템 성능이나 전반적인 효율성에 영향을 주는 방식을 측정합니다.

PERF 5 네트워킹 솔루션 구성 방법

시스템에 맞는 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항 등에 따라 다양합니다. 사용자 또는 온프레미스 리소스와 같은 물리적 제약은 엣지 기술 또는 리소스 배치를 통해 상쇄될 수 있는 위치 옵션을 구동시킵니다.

모범 사례:

- 네트워킹이 성능에 영향을 주는 방식 파악:: 네트워크 관련 결정 사항이 워크로드 성능에 영향을 주는 방식을 분석하고 파악합니다. 예를 들어 네트워크 지연 시간은 사용자 환경에 영향을 주는 경우가 많으므로 잘못된 프로토콜을 사용하는 경우 오버헤드가 너무 많아져 네트워크 용량이 고갈될 수 있습니다.
- 사용 가능한 제품 옵션 파악:: EC2 인스턴스 네트워크 기능, 강화된 네트워킹, Amazon EBS 최적화 인스턴스, Amazon S3 Transfer Acceleration, Amazon CloudFront를 사용한 동적 콘텐츠 전송 등 네트워크 관련 성능을 최적화하는 데 사용할 수 있는 서비스 수준 기능을 파악합니다.
- 사용 가능한 네트워킹 기능 평가:: 성능을 개선해 주는 AWS의 네트워킹 기능을 평가합니다. 테스트, 지표 및 분석을 통해 이러한 기능의 영향을 측정할 수 있습니다. 예를 들어 Amazon Route 53 지연 시간 기반 라우팅, Amazon VPC 종단점 또는 AWS Direct Connect 등의 사용 가능한 네트워크 수준 기능을 활용하여 지연 시간, 네트워크 거리 또는 지터를 줄일 수 있습니다.
- 최소한의 네트워크 ACL 사용:: 요구 사항을 충족하면서 ACL 수를 최소화하는 방식으로 네트워크를 설계합니다. ACL이 너무 많으면 네트워크 성능이 저하되어 시스템 성능이나 효율성이 낮아질 수 있습니다.
- 암호화 오프로딩 및 로드 밸런싱 활용:: 성능을 개선하고 트래픽을 효율적으로 관리/라우팅할 수 있도록 로드 밸런싱을 사용하여 암호화 종료(TLS)를 오프로드합니다. 워크로드가 AWS에서 제공하는 탄력성을 활용할 수 있도록 여러 리소스나 서비스에 트래픽을 분산시킵니다.
- 성능을 개선할 수 있는 네트워크 프로토콜 선택:: 시스템과 네트워크 간의 통신용 프로토콜을 선택할 때는 해당 프로토콜이 워크로드 성능에 영향을 주는 방식에 따라 프로토콜을 결정합니다.
- 네트워크 요구 사항에 따라 위치 선택:: AWS 리전, 가용 영역, 배치 그룹, 엣지 로케이션 등의 사용 가능한 위치 옵션을 활용해 네트워크 지연 시간을 줄이거나 처리량을 늘립니다.
- 지표를 기준으로 네트워크 구성 최적화:: 수집 및 분석한 데이터를 사용하여 정보에 입각해 네트워크 구성 최적화 관련 결정을 내립니다. 이러한 변경의 영향을 측정한 다음 영향 측정값을 활용해 이후 결정을 내립니다.

검토

PERF 6 새 릴리스를 활용하기 위해 워크로드를 개선하는 방법

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

모범 사례:

- 새 리소스 및 서비스를 지속적으로 파악:: 새 서비스, 설계 패턴 또는 제품 오퍼링을 사용할 수 있게 되면 성능을 개선할 수 있는 방식을 평가합니다. 임시 평가, 내부 논의 또는 외부 분석을 통해 워크로드의 효율성이나 성능을 개선할 수 있는 방법을 고려합니다.
- 워크로드 성능 개선을 위한 프로세스 정의:: 새 서비스, 설계 패턴, 리소스 유형 및 구성을 사용할 수 있게 되면 평가를 하기 위한 프로세스를 정의합니다. 예를 들어 새 인스턴스 오퍼링에 대해 기존 성능 테스트를 실행하여 해당 오퍼링 사용 시에 성능 또는 효율성이 개선되는 부분을 확인할 수 있습니다.
- 장기적으로 워크로드 성능 개선:: 조직에서 평가 프로세스를 통해 수집된 정보를 사용해 새 서비스나 리소스를 사용할 수 있게 되면 적극적으로 도입함으로써 워크로드의 성능이나 효율성을 개선합니다.

모니터링

PERF 7 예상된 성능이 발휘되도록 리소스를 모니터링하는 방법

시스템 성능은 시간이 지남에 따라 저하될 수 있습니다. 시스템 성능을 모니터링하여 성능 저하 상태를 식별하고 운영체제 또는 애플리케이션 부하와 같은 내부 또는 외부 요인을 해소합니다.

모범 사례:

- 성능 관련 지표 기록:: Amazon CloudWatch, 타사 서비스 또는 자체 관리형 모니터링 도구를 사용하여 성능 관련 지표를 기록합니다. 예를 들어 데이터베이스 트랜잭션, 속도가 느린 쿼리, I/O 지연 시간, HTTP 요청 처리량, 서비스 지연 시간 또는 기타 주요 데이터를 기록할 수 있습니다.
- 이벤트 또는 인시던트 발생 시의 지표 분석:: 이벤트나 인시던트에 대응하는 과정에서 모니터링 대시보드나 보고서를 사용해 이벤트/인시던트의 영향을 파악하고 진단합니다. 이러한 대시보드나 보고서에서는 필요한 수준의 성능을 제공하지 못하는 워크로드 부분을 파악할 수 있습니다.
- 워크로드 성능 측정을 위한 KPI 설정:: 시스템이 예상한 성능을 제공하는지 여부를 나타내는 KPI를 파악합니다. 예를 들어 API 기반 워크로드는 전반적인 성능의 지표로 전체 응답 지연 시간을 사용할 수 있으며, 전자상거래 사이트는 구매 건 수를 KPI로 사용하도록 선택할 수 있습니다.
- 모니터링을 사용하여 경보 기반 알림 생성:: 직접 정의한 성능 관련 KPI를 사용해 측정값이 예상 경계를 벗어나면 경보를 자동으로 생성하는 모니터링 시스템을 사용합니다.
- 일정한 간격으로 지표 검토:: 주기적인 유지 관리의 일환으로 또는 이벤트나 인시던트 대응 과정에서 수집된 지표를 검토합니다. 이러한 검토를 진행하면 문제를 해결하는 데 반드시 필요했던 지표, 그리고 문제를 확인/해결/방지하는 데 도움이 되었던 지표(추적한 경우)를 파악할 수 있습니다.
- 사전 모니터링 및 경보 생성:: KPI를 모니터링 및 알림 시스템과 함께 사용하여 성능 관련 문제를 사전에 해결합니다. 가능한 경우 경보를 사용해 문제 해결을 위한 자동화된 작업을 트리거합니다. 그리고 자동으로 대응할 수 없는 경우에는 대응 가능한 구성 요소로 경보를 에스컬레이션합니다. 예를 들어 필요한 KPI 값을 예측하고 해당 값이 특정 임계값을 초과하는 경우 경보를 생성할 수 있는 시스템이나, KPI가 필요한 값의 범위를 벗어나는 경우 배포를 자동으로 중지하거나 롤백할 수 있는 도구로 경보를 에스컬레이션할 수 있습니다.

트레이드오프

PERF 8 성능 향상을 위해 트레이드오프를 어떻게 사용하고 있습니까?

솔루션을 설계할 때 트레이드오프를 적극적으로 고려하면 최적의 접근 방식을 선택할 수 있습니다. 일관성, 내구성, 공간을 시간과 지연 시간으로 바꾸어 성능을 수시로 향상시킬 수 있습니다.

모범 사례:

- 성능이 가장 중요한 영역 파악:: 워크로드 성능을 개선하면 고객 환경이나 효율성을 개선할 수 있는 영역을 파악/확인합니다. 예를 들어 고객 상호 작용이 많이 수행되는 웹 사이트에서는 Amazon CloudFront 등의 엣지 서비스를 사용하여 콘텐츠 전송 위치를 고객과 더 가까운 곳으로 이동하면 성능을 개선할 수 있습니다.
- 설계 패턴 및 서비스 파악:: 워크로드 성능 개선에 도움이 되는 다양한 설계 패턴과 서비스를 조사하고 파악합니다. 분석의 일부분으로 성능 개선을 위해 절충할 수 있는 요소를 파악합니다. 예를 들어 Amazon ElastiCache를 사용하면 데이터베이스 시스템의 로드를 줄일 수 있습니다. 하지만 안전한 캐싱을 구현하기 위한 엔지니어링을 수행해야 하거나, 특정 영역에서 최종 일관성 개념을 도입해야 할 수 있습니다.
- 트레이드오프가 고객 및 효율성에 주는 영향 파악:: 성능 관련 개선 사항을 평가할 때는 선택한 옵션이 고객 및 워크로드 효율성에 미치는 영향을 고려합니다. 예를 들어 Amazon DynamoDB 등의 키-값 스토리지 사용 시 시스템 성능이 크게 향상되는 경우에는 Amazon DynamoDB의 최종 일관성 특성이 고객에게 줄 수 있는 영향도 평가해야 합니다.
- 성능 개선의 영향 측정:: 성능을 개선하기 위한 변경을 수행할 때는 수집된 지표와 데이터를 평가하여 워크로드, 구성 요소 및 고객에 대한 성능 개선의 영향을 확인합니다. 이 측정을 수행하면 절충을 통한 성능 개선을 파악할 수 있으며 부정적인 부작용 발생 여부를 확인할 수 있습니다.
- 다양한 성능 관련 전략 사용:: 해당하는 경우 다양한 전략을 활용하여 성능을 개선합니다. 예를 들어 데이터 캐싱 등의 전략을 사용해 과도한 네트워크 또는 데이터베이스 호출을 방지하고, 데이터베이스 엔진용 읽기 전용 복제본을 사용해 읽기 속도를 높이고, 가능한 경우 데이터 샤딩/압축을 수행하여 데이터 볼륨을 줄이고, 제공되는 결과를 버퍼링/스트리밍하여 차단을 방지하는 등의 전략을 사용할 수 있습니다.

비용 최적화

지출 인식

COST 1 사용량 관리 방법

목표 달성 과정에서 발생하는 비용을 적정 수준으로 유지하는 정책과 메커니즘을 설정합니다. '견제와 균형' 방식을 도입하면 비용을 과도하게 지출하지 않고 획기적인 방식으로 목표를 달성할 수 있습니다.

모범 사례:

- 조직 요구 사항에 따라 정책 개발:: 조직에서 리소스를 관리하는 방법을 정의하는 정책을 개발합니다. 정책에는 리소스 수명 주기 동안의 생성, 수정, 폐기를 비롯한 리소스와 워크로드의 비용 측면을 포함해야 합니다. 또한 워크로드의 목표와 비용 목표도 개발합니다.
- 계정 구조 구현:: 조직에 적합한 계정 구조를 구현합니다. 그러면 조직 전체에서 비용을 쉽게 할당하고 관리할 수 있습니다.
- 그룹 및 역할 구현:: 정책에 맞는 그룹과 역할을 구현하고 개발, 테스트, 프로덕션 등의 각 그룹에서 인스턴스와 리소스를 생성/수정/폐기할 수 있는 사람을 제어합니다. 이 모범 사례는 AWS 서비스와 타사 솔루션에 모두 적용됩니다.
- 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 IAM 정책을 사용하여 리전 또는 리소스 유형 액세스를 제어할 수 있습니다.
- 프로젝트 수명 주기 추적:: 프로젝트, 팀 및 환경의 수명 주기를 추적, 측정 및 감사하여 불필요한 리소스 사용 및 이에 대한 비용 지출을 방지합니다.

COST 2 사용량과 비용을 모니터링하는 방법

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

모범 사례:

- AWS 비용 및 사용 보고서 구성:: 자세한 사용 및 결제 정보를 캡처하도록 AWS 비용 및 사용 보고서를 구성합니다.
- 비용 기여 범주 파악:: 조직 내에서 비용을 할당하는 데 사용할 수 있는 조직 범주를 파악합니다.
- 조직 지표 설정:: 이 워크로드에 필요한 조직 지표를 설정합니다. 워크로드 지표의 예로는 생성된 고객 보고서 또는 고객에게 제공된 웹 페이지 등이 있습니다.
- 태그 지정 정의 및 구현:: 조직, 워크로드 속성 및 비용 할당 범주를 기준으로 하여 태그 지정 스키마를 정의하고 모든 리소스에 대해 태그 지정을 구현합니다.
- 결제 및 비용 관리 도구 구성:: 조직 정책에 맞게 AWS Cost Explorer 및 AWS Budgets를 구성합니다.
- 비용 최적화 관련 보고 및 알림:: 목표를 기준으로 하여 비용 및 사용량 알림을 제공하도록 AWS Budgets를 구성합니다. 정기 회의를 진행하여 이 워크로드의 비용 효율성을 분석하고 비용을 인지하는 조직 문화를 장려합니다.
- 비용 사전 모니터링:: 이 워크로드의 비용을 사전에 모니터링하기 위한 도구 및 대시보드를 구현합니다. 알림 수신 시 비용 및 범주만 확인해서는 안 됩니다. 이렇게 하면 긍정적인 추세를 파악하여 조직 전반에서 해당 방식을 사용하도록 장려할 수 있습니다.
- 워크로드 지표를 기준으로 비용 할당:: 지표 또는 비즈니스 성과를 기준으로 이 워크로드의 비용을 할당하여 워크로드 비용 효율성을 측정합니다. 분석 정보 및 결제 처리 기능을 제공할 수 있는 Amazon Athena를 사용하여 AWS 비용 및 사용 보고서를 분석하는 프로세스를 구현합니다.

COST 3 리소스 폐기 방법

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

모범 사례:

- 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.
- 폐기 프로세스 구현:: 분리된 리소스를 파악하고 폐기하는 프로세스를 구현합니다.
- 계획되지 않은 방식으로 리소스 폐기:: 계획되지 않은 방식으로 리소스를 폐기합니다. 이러한 폐기는 대개 정기 감사 등의 이벤트를 통해 트리거되며 보통 수동으로 수행합니다.
- 리소스 자동 폐기:: 중요하지 않은 리소스, 필수가 아닌 리소스 또는 사용률이 낮은 리소스를 파악하고 폐기하는 과정에서 워크로드가 리소스 종료를 정상적으로 처리하도록 설계합니다.

비용 효율적인 리소스

COST 4 서비스 선택 시의 비용 평가 방법

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 사용하여 관리 및 운영 고정 비요을 많이 줄이거나 없앨 수 있으며, 응용 프로그램 및 비즈니스 관련 활동을 수행할 수 있습니다.

모범 사례:

- 조직의 비용 요구 사항 파악:: 팀 구성원과 협의하여 이 워크로드의 비용 최적화와 기타 부문(예: 성능, 안정성)의 적절한 절충 수준을 정의합니다.
- 이 워크로드의 모든 구성 요소 분석:: 현재 크기나 비용에 관계없이 모든 워크로드 구성 요소를 분석해야 합니다. 검토 작업은 현재 비용과 예상 비용 등 제공될 수 있는 이점을 반영해야 합니다.
- 각 구성 요소의 철저한 분석 수행:: 조직에서 발생하는 각 구성 요소의 전반적인 비용을 확인합니다. 그런 다음 운영 및 관리 비용을 감안하여 총 소유 비용을 파악합니다(특히 관리형 서비스를 사용하는 경우). 검토 작업은 분석에 소요되는 시간 대비 구성 요소 비용 등의 제공될 수 있는 이점을 반영해야 합니다.
- 조직의 우선 순위에 따라 비용을 최적화하려는 이 워크로드의 구성 요소 선택:: 모든 구성 요소를 선택할 때는 비용을 고려해야 합니다. 이 과정에서는 Amazon RDS, Amazon DynamoDB, Amazon SNS, Amazon SES 등의 애플리케이션 수준 서비스와 관리형 서비스를 사용할 수 있습니다. 컴퓨팅 구성 요소의 경우에는 서버리스 서비스와 컨테이너를 사용합니다(예: AWS Lambda, 정적 웹 사이트용 Amazon S3, Amazon ECS). 그리고 오픈 소스 소프트웨어 또는 라이선스 요금이 없는 소프트웨어를 사용하여 라이선스 비용을 최소화합니다. 예를 들어 컴퓨팅 워크로드에 Amazon Linux를 사용하거나, Amazon Aurora로 데이터베이스를 마이그레이션할 수 있습니다.
- 시간별로 사용량이 달라지는 경우 비용 분석 수행:: 워크로드는 시간이 지나면 변경될 수 있으며 사용 수준이 달라지면 비용 효율성이 높아지는 서비스나 기능도 있습니다. 예상 사용량에 따라 시간별로 각 구성 요소 분석을 수행하면 수명 주기 내내 이 워크로드의 비용 효율성을 유지할 수 있습니다.

COST 5 리소스 유형과 크기 선택 시 비용 목표를 충족하는 방법

진행 중인 작업에 대해 적절한 리소스 크기를 선택해야 합니다. 가장 비용 효율적인 유형 및 크기를 선택하면 리소스 낭비를 최소화할 수 있습니다.

모범 사례:

- 비용 모델링 수행:: 조직 요구 사항을 파악하고 워크로드 및 각 워크로드 구성 요소의 비용 모델링을 수행합니다. 그리고 예상되는 다양한 부하에서 워크로드의 벤치마크 활동을 수행하여 비용을 비교합니다. 모델링 작업은 소요되는 시간 대비 구성 요소 비용 등의 제공될 수 있는 이점을 반영해야 합니다.
- 예측 결과에 따라 리소스 유형 및 크기 선택: 워크로드 및 리소스 특성(예: 컴퓨팅, 메모리, 처리량, 쓰기 집약형)을 기준으로 리소스 크기나 유형을 평가합니다. 일반적으로는 온프레미스 버전과 같은 워크로드의 이전 버전, 설명서 또는 워크로드와 관련된 기타 정보 출처를 사용해 리소스 사용량을 예측합니다.
- 지표를 기준으로 리소스 유형 및 크기 선택: 현재 실행 중인 워크로드의 지표를 사용하여 비용을 최적화하기에 적합한 크기와 유형을 선택합니다. Amazon EC2, Amazon DynamoDB, Amazon EBS (PIOPS), Amazon RDS, Amazon EMR 및 네트워킹과 같은 서비스의 처리량, 크기 및 스토리지를 적절하게 프로비저닝합니다. 자동 조정 등의 피드백 루프나 수동 크기 조정 프로세스를 통해 이 프로비저닝을 수행할 수 있습니다.

COST 6 요금 모델을 사용하여 비용을 절감하는 방법

리소스가 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.

모범 사례:

- 요금 모델 분석 수행:: AWS Cost Explorer의 예약 인스턴스 권장 사항 기능을 사용하여 워크로드 분석을 수행합니다.
- 적용 범위 비율이 낮은 여러 요금 모델 구현:: 워크로드에서 적용 범위 비율이 낮은 예약 용량, 스팟 인스턴스, 스팟 블록 또는 스팟 집합(전체 권장 적용 범위의 80% 미만)을 구현합니다.
- 비용을 기준으로 리전 구현:: 리소스 요금은 리전별로 다를 수 있습니다. 따라서 리전 비용을 고려하면 이 워크로드의 전체 가격을 최저 수준으로 낮출 수 있습니다.
- 이 워크로드의 모든 구성 요소용 요금 모델 구현:: 영구적으로 실행되는 리소스의 경우 예약 용량의 적용 범위 비율이 높습니다(구현되는 권장 적용 범위의 80% 이상). 짧은 기간 동안만 사용할 용량은 스팟 인스턴스, 스팟 블록 또는 스팟 집합을 사용하도록 구성됩니다. 온디맨드 옵션은 예약 용량을 사용할 만큼 충분히 길게 실행되지 않으며(리소스 유형에 따라 1년 중 대개 25~75%에 해당하는 기간에 실행됨) 중단할 수 없는 단기 워크로드에만 사용됩니다.

COST 7 데이터 전송 요금 부과 계획 방법

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:

- 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.
- 데이터 전송 비용을 최적화할 구성 요소 선택:: 모든 구성 요소를 선택해야 하며, 데이터 전송 비용을 줄이도록 아키텍처를 설계해야 합니다. 이 과정에서는 WAN 최적화, 다중 AZ 구성 등의 구성 요소를 사용할 수 있습니다.
- 데이터 전송 비용을 줄이기 위한 서비스 구현:: 데이터 전송 비용을 줄이기 위한 서비스를 구현합니다. 예를 들어 Amazon CloudFront 등의 CDN을 사용해 최종 사용자에게 콘텐츠를 전송하거나, Amazon ElastiCache를 사용하여 계층을 캐시하거나, VPN 대신 AWS Direct Connect를 사용해 AWS에 연결할 수 있습니다.

수요와 공급 일치

COST 8 수요와 일치하도록 리소스를 공급하는 방법

비용과 성능을 적절하게 절충한 워크로드에서는 비용을 결제한 모든 리소스가 사용되는지 확인하고, 사용률이 매우 낮은 인스턴스가 없도록 해야 합니다. 사용률 지표가 매우 높거나 낮은 리소스가 있으면 조직의 운영 비용이 증가하거나(사용률이 너무 높아 성능이 저하됨), 과도한 프로비저닝으로 인해 AWS에 지출한 금액이 낭비됩니다.

모범 사례:

- 워크로드 수요 분석 수행:: 시간별 워크로드 수요를 분석합니다. 이 분석에서는 시기별 추세를 파악하고 전체 워크로드 수명 주기 동안의 작동 상태를 정확하게 반영해야 합니다. 분석 작업은 소요되는 시간 대비 워크로드 비용 등의 제공될 수 있는 이점을 반영해야 합니다.
- 사후 대응식으로/계획 없이 리소스 프로비저닝:: 리소스 수준은 수요에 따라 바뀌지만 프로비저닝은 계획 없이 대개 수동으로 진행되며, 워크로드가 변경되거나 문제가 되는 이벤트가 발생하면 트리거됩니다. 반면 리소스 공급 방식은 변경되는 속도가 느리기 때문에 일반적으로 리소스가 너무 많거나 적게 프로비저닝됩니다.
- 동적으로 리소스 프로비저닝:: 리소스가 계획된 방식으로 프로비저닝됩니다. 이 경우 리소스는 자동 조정 등의 방식을 통해 수요를 기반으로 프로비저닝되거나, 버퍼를 기반으로 프로비저닝되거나(수요가 시간별로 분산되며 전체 리소스 사용량이 감소함), 시간을 기반으로 프로비저닝됩니다(수요를 예측할 수 있으며 시간에 따라 리소스가 제공됨). 이러한 방법을 사용하면 너무 많거나 적게 프로비저닝되는 리소스의 양을 최소화할 수 있습니다.

시간 경과에 따른 최적화

COST 9 새로운 서비스를 어떻게 평가하십니까?

AWS는 새로운 서비스와 기능을 출시하므로, 기존 아키텍처 결정을 검토하여 최고의 비용 효율성을 유지하는 것이 좋습니다.

모범 사례:

- 비용 최적화 기능 설정:: 조직 전체에서 비용과 사용량을 정기적으로 검토하는 팀을 구성합니다.
- 워크로드 검토 프로세스 개발:: 워크로드 검토 기준과 프로세스를 정의하는 프로세스를 개발합니다. 검토 작업은 제공될 수 있는 이점을 반영해야 합니다. 예를 들어 핵심 워크로드와 비용이 전체 결제 비용의 10%보다 많은 워크로드는 분기별로 검토하고, 비용이 전체 결제 비용의 10% 미만인 워크로드는 연 1회 검토할 수 있습니다.
- 계획 없이 서비스 검토 및 구현:: 계획되지 않은 방식으로 새로운 서비스를 도입합니다.
- 정기적으로 이 워크로드 검토 및 분석:: 정의된 프로세스에 따라 기존 워크로드를 정기적으로 검토합니다.
- 신규 서비스 릴리스를 적용하여 최신 상태 유지:: 정기적으로 전문가 또는 APN 파트너의 상담을 받아 더 저렴한 가격을 제공하는 서비스와 기능을 고려합니다. AWS 블로그 및 기타 정보 출처도 검토합니다.