AWS Well-Architected 프레임워크

2020년 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)사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여주며, (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 솔루션즈 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	효과적인 개발 및 워크로드 실행을 지원하고, 작업에 대한 인사이트를 얻고, 지원 프로세스 및 절차를 지속적으로 개선하여 비즈니스 가치를 제공할 수 있는 능력입니다.
보안	보안 원칙에는 클라우드 기술을 활용하여 보안을 강화하고 데이터, 시스템 및 자산을 보호하는 능력이 포함됩니다.
안정성	워크로드의 의도한 기능이 수행될 것으로 예상되는 시기에 이 기능을 올바르고 일관적으로 수행하는 기능을 말하며 총 수명 주기에 걸쳐 워크로드를 운영 및 테스트할 수 있는 기능이 포함됩니다.
성능 효율성	컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 변화에 따라 이러한 효율성을 유지하는 기능이 포함됩니다.
비용 최적화	시스템을 실행하여 최저 가격으로 비즈니스 가치를 제공할 수 있는 기능이 포함됩니다.

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 소유권의 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), 솔루션즈 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. 팀 내 표준 충족 확인에 대한 감사 등을 할 때 의사결정 권한을 분산한다면 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 전문가를 배치합니다. 둘째로, 해당 사례를 실제로 적용하기 위한 메커니즘을 구현합니다.[2] 표준 충족 여부를 확인하는 자동화된 검사를 수행합니다. 이러한 분산된 접근 방식은 Amazon 리더십 원칙을 통해 뒷받침되며 Working backward을 수행하는 모든 역할에서 문화를 만듭니다.[3] 문화를 수립합니다. 고객 중심의 팀은 고객의 요구 사항에 대응하여 제품을 개발합니다.

아키텍처 측면에서, 이는 모든 팀이 아키텍처를 생성하고 모범 사례를 따르는 기능을 갖추고 있음을 의미합니다. AWS는 새로운 팀이 이러한 기능을 확보하거나 기존 팀이 기준을 높이도록 돕기 위해, 설계를 검토하고 AWS 모범 사례가 무엇인지 이해하는 데 도움을 주는 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, 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-Architected는 내부 검토 프로세스를 고객 중심으로 구현한 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 믿습니

[1] [2]작업 수행 방식, 프로세스, 표준 및 용인된 규범

[3] “ 의도가 좋다고 효과도 좋은 것은 아닙니다. 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.

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

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

- 필요 용량에 대한 추측 불필요: 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 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.
- 실전 테스트를 통한 개선: 프로덕션 환경에서 이벤트를 시뮬레이션하기 위한 실전 테스트를 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 효과적인 개발 및 워크로드 실행을 지원하고, 작업에 대한 인사이트를 얻고, 지원 프로세스 및 절차를 지속적으로 개선하여 비즈니스 가치를 제공할 수 있는 능력입니다. 이(가) 포함됩니다.

운영 우수성 원칙에서는 설계 원리 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 선제적 가이드는 운영 우수성 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 운영 우수성 에 대한 다섯개의 설계 원칙이 있습니다.

- 코드를 통한 운영: 애플리케이션 코드를 위해 사용하였던 엔지니어링 원칙을 클라우드에서 인프라를 포함한 환경에 적용할 수 있습니다. 클라우드에서는 전체 워크로드(애플리케이션, 인프라)를 코드로 정의하고 코드와 함께 업데이트할 수 있고 운영 절차를 코드로 구현하고 이벤트에 대응하여 이를 트리거하면 실행을 자동화할 수 있습니다. 코드로 운영을 수행하면 인적 오류가 제한되고 이벤트에 대한 일관된 대응이 가능합니다.
- 자주 발생하고 되돌릴 수 있는 사소한 변경 내용 적용: 요소를 정기적으로 업데이트할 수 있도록 워크로드를 설계하고, 실패할 경우 되돌릴 수 있는 작은 증분으로 변경 내용을 적용합니다(가능하면 고객에게 영향을 주지 않도록).
- 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 확인합니다. 정기적인 게임 데이를 준비하여 시뮬레이션된 이벤트에 대한 팀의 대응 및 워크로드를 테스트합니다.
- 모든 운영상 실패로부터 학습: 모든 운영상 이벤트 및 실패로부터 파악한 내용을 통해 개선합니다. 팀 전반 및 조직 전체에서 파악한 내용을 공유합니다.

정의

클라우드에는 운영 우수성 에 대한 넷개의 모범 사례 영역이 있습니다.

- 조직
- 준비
- 운영
- 개선

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

워크로드 운영에 내재된 위험이 있을 수 있습니다. 이러한 위험을 파악하고 정보에 근거하여 프로덕션 환경에 적용할지 여부를 결정해야 합니다. 그리고 팀에서 워크로드를 지원할 수 있어야 합니다. 원하는 비즈니스 성과에서 도출된 비즈니스 및 운영 지표를 통해 워크로드 상태, 운영 활동, 인시던트에 대한 대응 능력을 파악할 수 있습니다. 우선순위는 비즈니스 요구 사항과 비즈니스 환경 변화에 따라 달라집니다. 이를 피드백 루프로 활용하여 조직과 워크로드 운영을 지속적으로 개선합니다.

모범 사례

조직

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

비즈니스에 대한 위협 요소(예: 비즈니스상의 위험 및 법적 책임, 정보 보안 위협)를 평가하고 위험 목록에서 이 정보를 관리합니다. 위험의 영향과 상충하는 이해 관계나 대안 사이의 장단점을 평가합니다. 예를 들어, 비용 최적화보다 새로운 기능의 시장 출시를 앞당기는 데 더 역점을 둘 수 있습니다. 아니면 리팩터링 없이 시스템 마이그레이션 작업을 간소화하기 위해 비관계형 데이터용 솔루션으로 관계형 데이터베이스를 선택할 수도 있습니다. 주력할 영역을 결정할 때 정보를 토대로

적절한 결정을 내릴 수 있도록 이점과 위험을 관리합니다. 일부 위험이나 선택은 한동안 감수할 수 있거나, 관련 위험을 완화할 수도 있겠지만, 위험을 감수할 수 없는 경우에는 위험을 해결하기 위한 조치를 취해야 합니다.

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

애플리케이션, 워크로드, 플랫폼 및 인프라 구성 요소마다 소유자가 명시되어 있고, 각 프로세스와 절차의 정의 및 실행을 담당하는 소유자가 각각 명시되어 있는지 확인합니다. 각 구성 요소, 프로세스 및 절차의 비즈니스 가치, 이러한 리소스가 배치되거나 활동이 수행되는 이유, 그러한 소유권이 존재하는 이유를 파악하면 팀원의 작업을 알 수 있습니다. 팀원이 적절하게 행동하고 책임과 소유권을 식별하는 메커니즘이 마련되도록 팀원의 책임을 명확하게 정의합니다. 혁신에 제약이 없도록 추가, 변경 및 예외를 요청하는 메커니즘을 마련합니다. 팀 간의 협력을 통해 서로를 지원하는 방법과 비즈니스 성과를 설명하는 계약을 정의합니다.

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

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

조직에 적용되는 외부 규제 또는 규정 준수 요구 사항이 있다면 팀원이 우선순위에 대한 영향을 확인할 수 있도록 AWS 클라우드 규정 준수에서 제공하는 리소스를 사용하여 관련 정보를 제공해야 합니다. Well-Architected 프레임워크에서는 학습, 평가 및 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 확장 가능한 설계를 구현하는 일관된 접근 방식을 제공합니다. AWS에서 제공하는 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 Trusted Advisor는 우선순위 결정에 도움이 될 수 있는 최적화 방안을 알려주는 핵심 검사 세트에 액세스할 수 있는 도구입니다. Business 및 Enterprise Support 고객에게는 우선순위를 더욱 자세히 결정하는 데 사용할 수 있는 추가 검사 기능이 제공됩니다. 이러한 기능을 사용하면 보안, 안정성, 성능 및 비용 최적화 영역을 중점적으로 확인할 수 있습니다.

AWS를 활용하면 선택한 방식이 워크로드에 어떤 영향을 주는지를 더 자세히 파악할 수 있도록 팀에게 AWS 및 해당 서비스 관련 정보를 제공할 수 있습니다. AWS Support에서 제공하는 리소스(AWS Knowledge Center, AWS 토론 양식 및 AWS 지원 센터) 및 AWS 설명서를 사용하여 팀에게 관련 정보를 제공해야 합니다. AWS 관련 문의 사항에 대해 지원을 받으려면 AWS 지원 센터를 통해 AWS Support에 지원을 요청하십시오. 또한 AWS는 AWS의 운영을 통해 학습한 모범 사례와 패턴을 Amazon Builders' Library에서 공유합니다. AWS 블로그 및 공식 AWS 팟캐스트에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다. AWS Training and Certification에서는 AWS 기초에 관한 자습형 디지털 과정을 통해 무료 교육을 제공합니다. 팀이 AWS 기술을 발전시켜 나갈 수 있도록 강의식 교육에 등록할 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 이 관리 기능을 확장합니다. Managed Services 공급자(예: AWS Managed Services, AWS Managed Services 파트너 또는 AWS Partner Network의 Managed Services 공급자)는 클라우드 환경을 구현하는 전문성을 제공하며, 보안 및 규정 준수 요구 사항과 비즈니스 목표를 지원합니다. Managed Services를 운영 모델에 추가하면 시간과 리소스를 절약할 수 있으며, 새로운 기술과 기능을 개발하는 대신 내부 팀을 간소화하며, 팀이 비즈니스를 차별화하는 전략적 결과에 집중할 수 있습니다.

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

OPS 1: 운영 우선순위를 결정하는 요인은 무엇입니까?

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

OPS 2: 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합니까?

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

OPS 3: 조직 문화는 비즈니스 성과를 어떻게 지원합니까?

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

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

준비

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

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

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

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

워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다. 수동 또는 자동화된 체크리스트를 비롯한 일관된 프로세스를 사용해 워크로드 또는 변경에 응답하는 준비 여부를 확인해야 합니다. 이렇게 하면 문제 해결 계획을 세워야 하는 영역도 파악할 수 있습니다. 일상 활동을 문서화한 런북과 문제 해결 프로세스를 안내하는 플레이북을 준비합니다. 이점과 위험을 파악하여 프로덕션에 변경 사항 적용에 대해 정보에 입각한 결정을 내립니다.

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

다음 질문은 운영 우수성 에 대한 이러한 고려 사항을 중점적으로 다룹니다.

OPS 4: 어떻게 운영 상태를 파악할 수 있도록 워크로드를 설계하십니까?
모든 구성 요소에서 지표, 로그, 추적 등의 내부 상태를 파악하는 데 필요한 정보를 제공하도록 워크로드를 설계합니다. 이렇게 하면 효율적으로 적절한 대응을 할 수 있습니다.
OPS 5: 귀사는 어떻게 결함을 줄이고 수정 작업을 쉽게 수행하고 프로덕션으로 이어지는 흐름을 개선하십니까?
프로덕션 환경으로 변경 사항을 전달하는 흐름을 개선할 수 있는 방식을 도입합니다. 이 방식은 리팩터링, 품질과 관련된 빠른 피드백 및 버그 수정을 지원해야 합니다. 이러한 방식을 도입하면 유용한 변경 사항을 프로덕션 환경으로 빠르게 전달할 수 있고, 문제 배포 가능성을 제한할 수 있으며, 배포 활동을 통해 발생하는 문제를 빠르게 파악하고 해결할 수 있습니다.
OPS 6: 배포 위험을 최소화하기 위해 어떻게 노력하나요?
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: 서비스 운영을 지원할 준비가 되어있는지를 어떻게 알 수 있나요?
워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다.

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

운영

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

운영 우수성을 달성하려면 효과적이고 효율적인 운영 이벤트 관리가 필요합니다. 이는 계획된 운영 이벤트 및 계획되지 않은 운영 이벤트 모두에 적용됩니다. 사전에 파악된 이벤트에 대해 런북을 작성하여 사용하고, 문제 조사 및 해결에 도움이 되는 해결책을 지원하는 데는 플레이북을 사용합니다. 비즈니스 및 고객 영향을 기반으로 이벤트 응답의 우선순위를 지정합니다. 이벤트 응답에 대해 알람이 발생하는지, 연결된 실행 프로세스가 있는지 여부를 식별된 담당자와 함께 확인합니다.

이벤트를 해결하는 데 필요한 인력을 미리 정하고 에스컬레이션 트리거를 포함하여 필요할 경우 긴급성과 영향을 기반으로 추가 인력의 참여를 유도합니다. 권한이 있는 개인을 식별하고 참여시켜 이전에 해결되지 않은 이벤트 대응에 대해 대응 과정이 비즈니스에 영향을 미쳤는지 확인합니다.

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

AWS에서는 AWS의 기본 지표 및 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 Flow Logs 등의 로그 기능을 통해 워크로드 인사이트를 제공합니다.

다음 질문은 운영 우수성 에 대한 이러한 고려 사항을 중점적으로 다룹니다.

OPS 8: 워크로드가 정상인지 어떻게 판단하나요?
워크로드 지표를 정의, 파악 및 분석하면 워크로드 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.
OPS 9: 운영 업무가 정상인지 어떻게 판단하나요?
운영 지표를 정의, 파악 및 분석하면 운영 이벤트를 확인하여 적절한 조치를 취할 수 있습니다.
OPS 10: 서비스/운영 이벤트를 어떻게 관리하나요?
이벤트로 인해 워크로드가 중단될 가능성을 최소화할 수 있도록 이벤트 대응을 위한 절차를 준비/확인합니다.

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

개선

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

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

AWS에서 Amazon S3으로 로그 데이터를 내보내거나 장기 보관을 위해 Amazon S3으로 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 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 수 있는 추세와 관심 이벤트를 찾습니다.

다음 질문은 운영 우수성 에 대한 이러한 고려 사항을 중점적으로 다룹니다.

OPS 11: 운영을 어떻게 개선 시키나요? 시간과 리소스를 할애하여 점진적 개선을 지속적으로 수행하면 운영 효율성을 높일 수 있습니다.

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

리소스

운영 우수성 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 보안 원칙에는 클라우드 기술을 활용하여 보안을 강화하고 데이터, 시스템 및 자산을 보호하는 능력이 포함됩니다.이(가) 포함됩니다.

보안 원칙은 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 규범적 지침은 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. 자격 증명 관리를 중앙 집중화하고 장기적인 정적 자격 증명에 대한 의존도를 해소하는 것을 목표로 합니다.

- 추적 기능 활성화: 실시간으로 환경에 대한 작업 및 변경 사항을 모니터링하고 알림을 전송하며 감사합니다. 로그 및 지표 수집을 시스템과 통합하여 자동으로 조사하고 조치를 취합니다.
- 모든 계층에 보안 적용: 여러 보안 제어 기능을 통해 심층 방어 방식을 적용합니다. 모든 계층(예: 네트워크 엣지, VPC, 로드 밸런싱, 모든 인스턴스 및 컴퓨팅 서비스, 운영 체제, 애플리케이션, 코드)에 적용됩니다.
- 보안 모범 사례의 자동 적용: 자동화된 소프트웨어 기반의 보안 메커니즘은 더욱 빠르게 안전한 확장 능력을 향상 시켜주며 비용 효율적입니다. 버전 제어가 가능한 템플릿에서 코드로 정의되고 관리되는 제어 기능의 구현을 비롯한 보안 아키텍처를 생성합니다.
- 전송 및 보관 중인 데이터 보호: 데이터를 민감도 수준으로 분류하고 적절한 경우 암호화, 토큰화 및 액세스 제어와 같은 메커니즘을 사용합니다.
- 사람들이 데이터에 쉽게 액세스할 수 없도록 유지: 데이터의 직접 액세스 또는 수동 처리의 필요성을 줄이거나 없애기 위한 메커니즘 및 도구를 사용합니다. 이를 통해 민감한 데이터를 처리할 때 잘못된 취급이나 수정 및 수작업으로 인한 오류의 위험을 줄일 수 있습니다.
- 보안 이벤트에 대비: 조직의 요구 사항에 부합하는 인시던트 관리 및 조사 정책과 프로세스를 마련하여 인시던트에 대비합니다. 인시던트 대응 시뮬레이션을 실행하고 자동화된 도구를 사용하여 감지, 조사 및 복구 속도를 높입니다.

정의

클라우드에는 보안에 대한 여섯개의 모범 사례 영역이 있습니다.

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

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

AWS 책임공유모델은 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 대응도 자동화되어 있습니다.

모범 사례

보안

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

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

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

SEC 1: 워크로드를 안전하게 운영하려면 어떻게 해야 합니까?

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

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, 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.

SEC 2: 사람과 시스템에 대한 자격 증명은 어떻게 관리합니까?

안전한 AWS 워크로드 운영에 접근할 때 관리해야 하는 두 가지 유형의 자격 증명이 있습니다. 액세스 권한을 관리하고 부여하는 데 필요한 자격 증명의 유형을 이해하면 적절한 자격 증명이 적절한 조건에서 적절한 리소스에 액세스할 수 있도록 보장할 수 있습니다. 인적 자격 증명: 관리자, 개발자, 운영자 및 최종 사용자가 AWS 환경 및 애플리케이션에 액세스하려면 자격 증명이 필요합니다. 이들은 조직의 구성원이거나 협업하는 외부 사용자로, 웹 브라우저, 클라이언트 애플리케이션 또는 대화형 명령줄 도구를 통해 AWS 리소스와 상호작용합니다. 시스템 자격 증명: 서비스 애플리케이션, 운영 도구 및 워크로드에서 AWS 서비스에 요청을 하려면(예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 자격 증명에는 Amazon EC2 인스턴스 또는 AWS Lambda 함수와 같이 AWS 환경에서 실행되는 시스템이 포함됩니다. 액세스가 필요한 외부 당사자의 시스템 자격 증명을 관리할 수도 있습니다. 또한 AWS 환경에 액세스해야 하는 시스템이 AWS 외부에 있을 수도 있습니다.

SEC 3: 사람과 시스템에 대한 권한은 어떻게 관리합니까?

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

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

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

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

Well-Architected 워크로드에서 로그 관리가 중요한 이유는 보안 또는 포렌식부터 규제 또는 법적 요구 사항에 이르기까지 다양합니다. 잠재적 보안 인시던트를 식별하려면 로그를 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 MFA(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 5: 네트워크 리소스는 어떻게 보호합니까?
인터넷이든 프라이빗 네트워크이든 상관없이 어떤 형태든 네트워크 연결이 있는 워크로드에는 외부 및 내부 네트워크 기반 위협으로부터 보호하기 위한 다중 방어 계층이 필요합니다.
SEC 6: 컴퓨팅 리소스를 어떻게 보호 하시나요?
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 관례부터 마련해야 합니다. 예를 들어 데이터 분류는 민감도에 따라 조직의 데이터를 구분하는 하나의 방법이고 암호화는 무단 액세스 사용자가 데이터를 해석하지 못하게 만들어 데이터를 보호하는 방법입니다. 이는 금전적 손해 방지 또는 규제 의무 준수 등 목표 달성을 뒷받침하는 중요한 도구이자 기법입니다.

AWS에서는 다음과 같은 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 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.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 7: 데이터는 어떻게 분류합니까?
분류는 적절한 보호 및 보존 제어 수준을 결정하는 데 도움이 되도록 중요도와 민감도를 기준으로 데이터를 분류하는 방법을 제공합니다.
SEC 8: 저장된 데이터는 어떻게 보호합니까?
무단 액세스 또는 취급 부주의의 위험을 줄이기 위해 여러 제어 기능을 구현하여 저장된 데이터를 보호합니다.
SEC 9: 전송 중인 데이터는 어떻게 보호합니까?
무단 액세스 또는 손실의 위험을 줄이기 위해 여러 제어 기능을 구현하여 전송 중인 데이터를 보호합니다.

AWS는 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 포렌식을 진행할 수 있습니다.

다음 질문은 보안에 대한 이러한 고려 사항을 중점적으로 다룹니다.

SEC 10: 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구합니까?

조직의 업무 중단을 최소화할 수 있도록 보안 인시던트를 제때 효고적으로 조사 및 대응하고 사후 복구하려면 철저한 준비가 필요합니다.

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

리소스

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

설명서

- AWS 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 워크로드의 의도한 기능이 수행될 것으로 예상되는 시기에 이 기능을 올바르고 일관적으로 수행하는 기능을 말하며 총 수명 주기에 걸쳐 워크로드를 운영 및 테스트할 수 있는 기능이 포함됩니다.이(가) 포함됩니다.

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

설계 원칙

클라우드에는 안정성에 대한 다섯개의 설계 원칙이 있습니다.

- 장애 자동 복구: 워크로드의 KPI(핵심 성능 지표)를 모니터링하면 임계값이 위반될 때 자동화를 트리거할 수 있습니다. 이러한 KPI는 서비스 작동의 기술적 측면이 아닌 비즈니스 가치를 측정한 값이어야 합니다. KPI를 모니터링하면 장애 추적 및 자동 알림을 지원하고, 자동화된 복구 프로세스에 따라 장애 지점을 우회하거나 복구할 수 있습니다. 보다 정교한 자동화를 구현할 경우 장애가 발생하기 전에 예측하여 해결하는 것도 가능합니다.
- 복구 절차 테스트: 온프레미스 환경에서 테스트는 워크로드가 특정 시나리오에서 작동하는 것을 증명하기 위해 시행됩니다. 일반적으로 복구 전략을 검증하기 위해 테스트하지는 않습니다. 클라우드에서는 워크로드의 장애 과정을 테스트하고 복구 절차를 검증할 수 있습니다. 자동화를 사용하여 다양한 장애를 시뮬레이션하거나 이전에 장애로 이어졌던 시나리오를 재현할 수 있습니다. 이 접근 방식은 장애 경로를 노출한 후 실제 장애 시나리오가 발생하기 전에 테스트하고 수정하여 위험을 줄일 수 있습니다.
- 수평적 확장으로 워크로드 전체 가용성 증대: 단일의 큰 리소스를 다수의 작은 리소스로 대체하여 단일 장애가 전체 워크로드에 미치는 영향을 축소합니다. 요청을 더 작은 리소스 여러 개로 분산시키면 공통의 장애 지점이 공유되지 않습니다.
- 용량 추정 불필요: 워크로드에 대한 수요가 해당 워크로드의 용량을 넘어서는 리소스 포화 상태는 온프레미스 워크로드에서 흔히 발생하는 장애의 원인입니다(서비스 거부 공격의 대상). 클라우드에서는 수요 및 워크로드 사용량을 모니터링하고 리소스 추가 또는 제거를 자동화함으로써 프로비저닝 과다 또는 부족 현상 없이 최적의 수준으로 수요를 충족할 수 있습니다. 클라우드에도 제한은 있지만 할당량을 어느 정도 제어하고 관리하는 것이 가능합니다(서비스 할당량 및 제약 조건 관리 참조).
- 자동화 변경 사항 관리: 인프라 변경은 자동화를 통해 수행되어야 합니다. 자동화 변경과 같은 변경을 관리해야 하며 이후에 이러한 변경을 추적하고 검토할 수 있습니다.

정의

클라우드에는 안정성에 대한 넷개의 모범 사례 영역이 있습니다.

- 기반
- 워크로드 아키텍처
- 변경 관리
- 장애 관리

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

모범 사례

기반

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

AWS에서는 이러한 기반에 관련된 요구 사항이 대부분 이미 통합되어 있거나 필요에 따라 적용할 수 있습니다. 클라우드는 거의 한계가 없도록 설계되었기 때문에 충분한 네트워킹 및 컴퓨팅 파워에 대한 요구 사항을 충족할 책임은 AWS에 있습니다. 따라서 고객은 리소스 크기와 할당을 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 할당량과 제약 조건은 어떻게 관리합니까?
클라우드 기반 워크로드 아키텍처에는 서비스 할당량(서비스 한도라고도 함)이라는 것이 있습니다. 이러한 할당량은 실수로 필요한 것보다 많은 리소스를 프로비저닝하는 것을 방지하고 API 작업에 대한 요청 속도를 제한하여 서비스를 침해로부터 보호하기 위해 존재합니다. 광섬유 케이블을 통해 비트를 푸시할 수 있는 속도나 물리적 디스크의 스토리지 양과 같은 리소스 제약 조건도 있습니다.
REL 2: 네트워크 토폴로지는 어떻게 계획합니까?
워크로드는 여러 환경에 존재할 수 있습니다. 여기에는 다중 클라우드 환경(퍼블릭 액세스 및 프라이빗)과 기존 데이터 센터 인프라가 포함됩니다. 따라서 시스템 내부 및 시스템 간 연결, 퍼블릭 IP 주소 관리, 프라이빗 IP 주소 관리 및 도메인 이름 확인과 같은 네트워크 고려 사항을 계획에 포함해야 합니다.

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

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

워크로드 아키텍처

안정적인 워크로드는 소프트웨어와 인프라에 대한 사전 설계 결정에서 시작됩니다. 아키텍처 선택은 모든 5가지 Well-Architected 원칙에서 워크로드 동작에 영향을 미칩니다. 안정성을 달성하려면 특정 패턴을 따라야 합니다.

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

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

REL 3: 워크로드 서비스 아키텍처는 어떻게 설계합니까?
SOA(서비스 지향 아키텍처) 또는 마이크로서비스 아키텍처를 사용하여 확장성과 안정성이 뛰어난 워크로드를 구축합니다. SOA(서비스 지향 아키텍처)는 서비스 인터페이스를 통해 소프트웨어 구성 요소를 재사용 가능하게 만드는 방식입니다. 마이크로서비스 아키텍처는 구성 요소를 더 작고 간단하게 만듭니다.
REL 4: 분산 시스템에서 장애 방지를 위한 상호 작용은 어떻게 설계합니까?
분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 여기에 나온 모범 사례는 장애를 방지하고 MTBF(평균 장애 간격)를 개선합니다.
REL 5: 분산 시스템에서 장애 완화 또는 극복을 위한 상호 작용은 어떻게 설계합니까?
분산 시스템에서 구성 요소(예: 서버 또는 서비스)는 통신 네트워크를 사용하여 상호 연결됩니다. 워크로드는 이러한 네트워크에서 데이터 손실 또는 지연 시간이 발생하더라도 안정적으로 작동해야 합니다. 분산 시스템의 구성 요소는 다른 구성 요소나 워크로드에 부정적인 영향을 미치지 않는 방식으로 작동해야 합니다. 이러한 모범 사례를 준수하면 워크로드가 스트레스 또는 장애를 견디고, 더 빠르게 이를 복구하며, 이러한 장애의 영향을 완화할 수 있습니다. 그러면 결과적으로 MTTR(평균 복구 시간)이 개선됩니다.

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.

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.

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

REL 6: 워크로드 리소스는 어떻게 모니터링합니까?
로그와 지표는 워크로드의 상태를 파악할 수 있는 유용한 도구입니다. 로그 및 지표를 모니터링하여 임계값을 초과하거나 중요한 이벤트가 발생하면 알림을 보내도록 워크로드를 구성할 수 있습니다. 모니터링을 수행하면 워크로드가 저성능 임계값을 초과하거나 장애가 발생할 때를 인식하고 이에 대응하여 자동으로 복구할 수 있습니다.
REL 7: 수요 변경에 따라 조정되도록 워크로드를 설계하려면 어떻게 해야 합니까?
확장 가능한 워크로드는 리소스를 자동으로 추가하거나 제거하여 특정 시기의 수요에 리소스 공급을 맞출 수 있는 탄력성을 제공합니다.
REL 8: 변경 사항은 어떻게 적용합니까?
새로운 기능을 배포하고 워크로드와 운영 환경에서 알려진 소프트웨어를 실행하고 예측 가능한 방식으로 패치 또는 교체할 수 있도록 하려면 변경 사항을 제어해야 합니다. 이러한 변경이 제어되지 않으면 변경의 영향을 예측하거나 변경으로 인해 발생하는 문제를 해결하기가 어려워집니다.

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

장애 관리

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

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

다음 질문은 안정성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

REL 9: 데이터는 어떻게 백업합니까?
RTO(복구 시간 목표) 및 RPO(복구 시점 목표)에 대한 요구 사항을 충족하도록 데이터, 애플리케이션 및 구성을 백업합니다.
REL 10: 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 합니까?
장애 격리 경계는 워크로드 내부 장애의 영향을 제한된 수의 구성 요소로 제한합니다. 경계 외부의 구성 요소는 장애가 발생하더라도 영향을 받지 않습니다. 다수의 장애 격리 경계를 사용하여 워크로드에 미치는 영향을 제한할 수 있습니다.
REL 11: 구성 요소 장애를 견디도록 워크로드를 설계하려면 어떻게 해야 합니까?
고가용성 및 낮은 MTTR(평균 복구 시간)이 요구되는 워크로드는 복원력을 고려하여 설계되어야 합니다.
REL 12: 안정성은 어떻게 테스트합니까?
프로덕션 환경의 스트레스에 대한 복원력을 가지도록 워크로드를 설계한 후 설계대로 작동하고 예상한 복원력을 제공하는지 확인할 수 있는 유일한 방법은 테스트입니다.
REL 13: DR(재해 복구)를 어떻게 계획합니까?
DR 전략의 시작은 백업 및 중복 워크로드 구성 요소를 갖추는 것입니다. RTO 및 RPO는 가용성 복원에 대한 목표입니다. 비즈니스 요구 사항에 따라 이러한 목표를 설정하십시오. 워크로드 리소스 및 데이터의 위치와 기능을 고려하여 이러한 목표를 충족하는 전략을 구현합니다.

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

리소스

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

설명서

- AWS Documentation
- AWS Global Infrastructure
- AWS Auto Scaling: How Scaling Plans Work
- What Is AWS Backup?

백서

- Reliability Pillar: AWS Well-Architected
- Implementing Microservices on AWS

성능 효율성

성능 효율성 원칙에는 컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 변화에 따라 이러한 효율성을 유지하는 기능이 포함됩니다. 이(가) 포함됩니다.

성능 효율성 원칙에서는 설계 원칙 개요, 모범 사례 및 질문 사항을 제공합니다. 구현 방법에 대한 규범적 지침은 성능 효율성 부문 백서에서 확인할 수 있습니다.

설계 원칙

클라우드에는 성능 효율성에 대한 다섯개의 설계 원칙이 있습니다.

- 고급 기술의 대중화: 팀에서 고급 기술을 손쉽게 구현할 수 있도록 복잡한 태스크를 클라우드 공급업체에 위임합니다. IT 팀에 새로운 기술의 호스팅 및 실행에 대해 알아볼 것을 요청하는 대신 기술을 서비스 형태로 사용하는 것을 고려해보십시오. 예를 들어 NoSQL 데이터베이스, 미디어 트랜스코딩 및 기계 학습은 모두 전문 지식이 요구되는 기술입니다. 클라우드에서는 이러한 기술이 팀에서 사용할 수 있는 서비스 형식으로 제공되므로 팀은 리소스 프로비저닝 및 관리가 아닌 제품 개발에 집중할 수 있습니다.
- 몇 분 만에 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.
- 실험 횟수 증가: 자동화할 수 있는 가상 리소스를 활용하며 여러 가지 인스턴스, 스토리지 또는 구성에 대한 비교 테스트를 신속하게 수행할 수 있습니다.
- 기계적 동조 고려: 클라우드 서비스가 어떻게 사용되는지 파악하고 항상 워크로드 목표에 가장 부합하는 기술 접근 방식을 사용합니다. 예를 들어 데이터베이스 또는 스토리지 접근 방식을 선택할 때는 데이터 액세스 패턴을 고려합니다.

정의

클라우드에는 성능 효율성에 대한 넷개의 모범 사례 영역이 있습니다.

- 선택
- 검토

- 모니터링
- 절충

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

정기적으로 선택 사항을 검토함으로써 계속 진화하는 AWS 클라우드를 최대한 활용할 수 있습니다. 모니터링을 수행하면 예상 성능과의 차이를 인식할 수 있습니다. 압축 또는 캐싱을 사용하거나 일관성 요구 사항을 완화하는 등의 절충을 통해 아키텍처를 설계하면 성능을 개선할 수 있습니다.

모범 사례

선택

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

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

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

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

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

데이터 기반 접근 방식으로 아키텍처에 적합한 패턴 및 구현을 선택하면 경제적인 솔루션을 구축할 수 있습니다. AWS 솔루션스 아키텍트, AWS 참조 아키텍처 및 APN(AWS 파트너 네트워크) 파트너는 업계 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 섹션에서는 고려할 네 가지 주요 리소스 유형(컴퓨팅, 스토리지, 데이터베이스, 네트워크)에 대해 살펴봅니다.

컴퓨팅

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

AWS에서 컴퓨팅은 인스턴스, 컨테이너, 함수의 세 가지 형태로 제공됩니다.

- 인스턴스는 버튼 또는 API 호출 한 번으로 기능을 변경할 수 있는 가상화된 서버입니다. 클라우드에서는 리소스를 한번 결정하면 그대로 고정되는 것이 아니므로 다양한 서버 유형을 시험해 볼 수 있습니다. AWS에서는 이러한 가상 서버 인스턴스를 다양한 제품군과 크기로 제공하며 SSD(Solid-State Drive)와 GPU(그래픽 처리 장치)를 비롯한 매우 다양한 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 사용자가 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 솔루션을 어떻게 선택합니까?

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

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

스토리지

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

AWS에서 스토리지는 객체, 블록 및 파일의 세 가지 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 DAS(Direct-Attached Storage) 또는 SAN(Storage Area Network)과 유사한 고가용성의 일관적이며 지연 시간이 짧은 블록 스토리지를 각 가상 호스트에 제공합니다. Amazon Elastic Block Store(Amazon EBS)는 EC2 인스턴스에서 영구 스토리지에 액세스할

수 있어야 하는 워크로드를 위해 설계되었으며, 적절한 스토리지 용량, 성능 및 비용으로 애플리케이션을 튜닝하는 데 도움이 됩니다.

- 파일 스토리지를 사용하면 여러 시스템에서 공유 파일 시스템에 액세스할 수 있습니다. Amazon Elastic File System(EFS)와 같은 파일 스토리지 솔루션은 대용량 콘텐츠 리포지토리, 개발 환경, 미디어 스토어 또는 사용자 홈 디렉터리와 같은 사용 사례에 적합합니다. Amazon FSx를 사용하면 주요 파일 시스템을 비용 효율적으로 손쉽게 시작하고 실행할 수 있으므로 널리 사용되는 오픈 소스 및 상용 라이선스 파일 시스템의 풍부한 기능 세트와 빠른 성능을 활용할 수 있습니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 3: 스토리지 솔루션을 어떻게 선택합니까?

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

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

데이터베이스

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

AWS에서는 관계형, 키-값, 문서, 인메모리, 그래프, 시계열 및 원장 데이터베이스를 비롯하여 특별히 구축된 여러 데이터베이스 엔진 중에서 선택할 수 있습니다. AWS 데이터베이스를 사용하면 서버 프로비저닝, 패치 적용, 설정, 구성, 백업 또는 복구와 같은 데이터베이스 관리 태스크에 대해 걱정할 필요가 없습니다. AWS는 자가 복구 스토리지와 Auto Scaling을 통해 클러스터를 지속적으로 모니터링하여 워크로드의 실행 상태를 유지합니다. 따라서 고객은 가치가 더 높은 애플리케이션 개발에 집중할 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.

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

네트워크

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

AWS에서 네트워킹은 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 요구 사항에 일치하는 네트워킹 방법을 보다 쉽게 찾을 수 있습니다. AWS는 네트워크 트래픽을 최적화하는 제품 기능(예: 향상된 네트워킹, Amazon EBS 최적화 인스턴스, Amazon S3 Transfer Acceleration 및 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, AWS Direct Connect 및 AWS Global Accelerator)도 제공합니다.

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 5: 네트워킹 솔루션을 구성하려면 어떻게 해야 합니까?

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

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

검토

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

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

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 6: 새 릴리스를 활용하여 워크로드를 어떻게 발전시킵니까?

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

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

모니터링

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

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

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 7: 리소스 성능을 모니터링하려면 어떻게 해야 합니까?

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

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

절충

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

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

다음 질문은 성능 효율성에 대한 이러한 고려 사항을 중점적으로 다룹니다.

PERF 8: 절충을 통해 성능을 개선하려면 어떻게 해야 합니까?

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

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

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

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

설계 원칙

클라우드에는 비용 최적화에 대한 다섯개의 설계 원칙이 있습니다.

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

정의

클라우드에는 비용 최적화에 대한 다섯개의 모범 사례 영역이 있습니다.

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

Well-Architected 프레임워크의 다른 원칙과 마찬가지로 비교 분석을 통해 하나를 선택해야 합니다. 예를 들어 출시 시간에 최적화할지 아니면 비용에 최적화할지 선택합니다. 선결제 비용 최적화에 투자하는 것보다 출시 시간을 단축하거나 새로운 기능을 배포하거나 단순히 납기를 준수하는 등 속도를 가장 높일 수 있도록 최적화하는 것이 가장 좋은 경우도 있습니다. 데이터를 고려하지 않고 급하게 설계 결정이 내려지는 경우도 있으며, 가장 비용 최적화된 구축 벤치마킹에 시간을 쓰

기보다 “만약을 대비해” 과잉 지출을 하려는 유혹은 항상 존재합니다. 이는 오버프로비저닝되고 부족하게 최적화된 배포로 이어질 수 있습니다. 하지만 온프레미스 환경에서 클라우드로 리소스를 그대로 이전한 후에 최적화를 수행해야 하는 경우에는 이러한 방식이 적절합니다. 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

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 혁신하고 구축하며, 더 민첩하고, 내부 또는 외부 요인에 적응합니다.

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

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

COST 1: 클라우드 재무 관리를 어떻게 구현합니까?
클라우드 재무 관리 시행을 통해 조직은 비용과 사용량을 최적화하고 AWS에서 규모를 확대하면서 비즈니스 가치를 실현하고 재정적 성공을 거둘 수 있습니다.

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

조직에서 비용에 대한 인식을 이행할 때 기존 프로그램과 프로세스를 바탕으로 한 개선이나 구축을 고려하십시오. 새로 구축하는 것보다 기존 프로세스와 프로그램에 추가하는 것이 훨씬 더 빠릅니다. 이를 통해 훨씬 더 빠르게 성과를 올릴 수 있습니다.

지출 및 사용량 인식

클라우드에서는 향상된 유연성과 민첩성을 바탕으로 혁신을 촉진하고 개발 및 배포를 가속화할 수 있습니다. 이는 하드웨어 사양을 식별하고, 가격 견적을 협상하고, 주문 번호를 관리하고, 배송을 예약한 후 리소스 배포하기와 같은 온프레미스 인프라 프로비저닝과 연관된 시간 및 수동 프로

세스를 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 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. 리소스에 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 예산을 통한 알림과 AWS Identity and Access Management(IAM) 및 서비스 할당량을 사용한 제어 기능을 통해 이루어집니다.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 2: 사용량을 어떻게 관리합니까?
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 3: 사용량과 비용을 어떻게 모니터링 합니까?
비용을 모니터링하고 적절하게 할당하기 위한 정책 및 절차를 구성합니다. 이렇게 하면 이 워크로드의 비용 효율성을 측정하고 개선할 수 있습니다.
COST 4: 리소스를 어떻게 폐기합니까?
프로젝트 시작부터 마지막까지의 전체 과정에서 변경 제어 및 리소스 관리를 구현합니다. 그러면 사용되지 않은 리소스를 종료하여 낭비되는 리소스를 줄일 수 있습니다.

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

비용과 사용량 보고 및 모니터링에서 적절한 수준의 세부 정보와 세분화를 사용해야 합니다. 개략적인 인사이트와 추세를 위해서는 AWS Cost Explorer에서 일 단위의 세부 수준을 사용합니다. 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 1

시간이 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 용량에 대한 비용을 지불할 수 있습니다. Savings Plans 및 예약 인스턴스는 온디맨드 요금보다 최대 75% 할인된 요금을 제공합니다. 스팟 인스턴스를 사용하면 미사용 Amazon EC2 용량을 활용할 수 있으며 온디맨드 요금보다 최대 90% 할인된 요금을 제공합니다. 스팟 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 5: 서비스를 선택할 때 비용을 어떻게 평가합니까?
Amazon EC2, Amazon EBS 및 Amazon S3는 기본 구성 AWS 서비스입니다. Amazon RDS 및 Amazon DynamoDB와 같은 관리형 서비스는 더 높은 수준이거나 애플리케이션 수준의 AWS 서비스입니다. 기본 구성 서비스와 관리형 서비스를 적절히 선택하여 이 워크로드의 비용을 최적화할 수 있습니다. 예를 들어, 관리형 서비스를 사용하여 관리 및 운영 고정 비용을 많이 줄이거나 없앨 수 있으며, 응용 프로그램 및 비즈니스 관련 활동을 수행할 수 있습니다.
COST 6: 리소스 유형, 크기 및 수 선택을 통해 비용 목표를 어떻게 달성합니까?
진행 중인 태스크에 대해 적절한 리소스 크기와 리소스 수를 선택해야 합니다. 가장 비용 효율적인 유형, 크기 및 수를 선택하여 리소스 낭비를 최소화할 수 있습니다.
COST 7: 비용 절감을 위해 가격 결정 모델을 어떻게 사용합니까?
리소스가 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.
COST 8: 데이터 전송 요금을 위한 계획은 어떻게 됩니까?
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 필요하지 않습

니다. 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.

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 9: 수요와 리소스 공급은 어떻게 관리합니까?

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

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

시간 경과에 따른 최적화

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

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

다음 질문은 비용 최적화에 대한 이러한 고려 사항을 중점적으로 다룹니다.

COST 10: 새로운 서비스를 어떻게 평가합니까?

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

리소스

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

설명서

- AWS Documentation

백서

- Cost Optimization Pillar

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.

검토는 제품 수명 주기의 주[1]요 이정표에 적용되어야 하며, 단방향 방식을 방지하기 위해 설계 단계 초기에 적용되어야 합니다. 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 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: “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 결정은 되돌릴 수 있는 양방향 방식입니다. 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, 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 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.

- Rodney Lester: 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
- Eric Pullen: Well-Architected 성능 부문 담당자, Amazon Web Services
- Seth Eliot: Well-Architected 안정성 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 솔루션즈 아키텍트, Amazon Web Services
- Scott Paddock: 보안 솔루션즈 아키텍트, Amazon Web Services
- Callum Hughes: 솔루션즈 아키텍트, Amazon Web Services

참고 문헌

AWS Cloud Compliance

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

The Amazon Builders' Library

문서 수정

표 2.

주요 개정 사항:

날짜	설명
2020년 7월	록 대다수 질문과 대답을 검토하여 재작성했습니다.
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

조직

OPS 1 운영 우선순위를 결정하는 요인은 무엇입니까?

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

모범 사례:

- 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.
- 내부 고객 요구 평가: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계자와 함께 내부 고객 요구 충족을 위한 주력할 영역을 결정합니다. 이렇게 하면 비즈니스 성과 달성에 필요한 운영 지원을 철저하게 파악할 수 있습니다.
- 거버넌스 요구 사항 평가: 조직에서 특정 사항을 준수하거나 강조하기 위해 정의한 지침이나 의무 사항을 알고 있어야 합니다. 조직 정책, 표준 및 요구 사항과 같은 내부 요인을 평가합니다. 거버넌스에 대한 변경 사항을 식별할 수 있는 메커니즘이 있는지 확인합니다. 거버넌스 요구 사항이 식별되지 않는 것으로 결론을 내릴 때에는 신중하게 판단하여 내린 결론인지 재차 확인해야 합니다.
- 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: 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.
- 장단점 평가: 주력할 영역을 결정하거나 수행할 조치를 선택할 때 정보를 토대로 결정을 내릴 수 있도록 상충하는 이해 관계나 대안 사이의 장단점을 평가합니다. 예를 들어, 비용 최적화보다 새로운 기능의 시장 출시를 앞당기는 데 더 역점을 둘 수 있습니다. 아니면 데이터 유형에 최적화된 데이터베이스로 마이그레이션하고 애플리케이션을 업데이트하는 대신, 시스템 마이그레이션 작업을 간소화하기 위해 비관계형 데이터용 솔루션으로 관계형 데이터베이스를 선택할 수도 있습니다.
- 이점 및 위험 관리: 주력할 영역을 결정할 때 정보를 토대로 적절한 결정을 내릴 수 있도록 이점과 위험을 관리합니다. 예를 들어 새 기능을 고객에게 제공하기 위해 해결되지 않은 문제가 남아 있는 워크로드를 배포하는 것이 이득일 수 있습니다. 관련 위험을 완화할 수도 있겠지만, 위험을 감수할 수 없는 경우에는 위험을 해결하기 위한 조치를 취해야 합니다.

OPS 2 비즈니스 성과를 지원하기 위해 조직을 어떻게 구성합니까?

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

모범 사례:

- 리소스 소유자 식별: 각 애플리케이션, 워크로드, 플랫폼 및 인프라 구성 요소의 소유권, 해당 구성 요소가 제공하는 비즈니스 가치, 그리고 소유권이 존재하는 이유를 파악합니다. 이러한 개별 구성 요소의 비즈니스 가치를 파악하고 이들이 비즈니스 성과를 어떻게 지원하는지 파악하면 해당 구성 요소에 적용되는 프로세스와 절차를 알 수 있습니다.
- 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.
- 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.
- 팀원이 자신의 책임을 알고 있습니다.: 역할의 책임과 비즈니스 성과에 기여하는 방법을 파악하면 작업의 우선순위와 역할이 중요한 이유를 알 수 있습니다. 이를 통해 팀원은 요구 사항을 인식하고 적절하게 대응할 수 있습니다.
- 책임과 소유권을 식별하는 메커니즘: 개인이나 팀을 식별하지 못할 경우 소유권을 할당하거나 요구 사항 충족을 위한 계획을 수립할 권한이 있는 사람에게 정해진 경로를 통해 사안을 에스컬레이션할 수 있습니다.
- 추가, 변경 및 예외를 요청하는 메커니즘: 프로세스, 절차 및 리소스의 소유자에게 요청을 보낼 수 있습니다. 이점과 위험을 평가한 후 요청이 적절한지 판단하고 정보에 입각한 의사 결정을 통해 실현 가능한 경우에 요청을 승인해야 합니다.
- 미리 정의되었거나 협상된 팀 간 책임: 팀 간에 서로 협력하고 지원하는 방식에 관한 내용을 정의하거나 협상해둡니다(예: 응답 시간, 서비스 수준 목표 또는 서비스 수준 계약). 팀의 작업이 비즈니스 성과에 미치는 영향, 그리고 다른 팀과 조직의 성과에 미치는 영향을 이해하면 작업의 우선순위를 파악하고 적절하게 대응할 수 있습니다.

OPS 3 조직 문화는 비즈니스 성과를 어떻게 지원합니까?

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

모범 사례:

- 경영진의 지원: 최고 경영진은 조직에 대한 기대치를 명확하게 설정하고 성공 여부를 평가합니다. 최고 경영진은 조직이 발전하고 모범 사례를 도입하도록 하는 동인이자 후원자이자 지지자입니다.
- 팀원은 성과가 위험한 상태일 때 조치를 취할 수 있는 권한이 있어야 합니다.: 워크로드 소유자는 성과가 위험한 상태일 때 팀원들이 응답할 수 있도록 지침과 범위를 정의했습니다. 에스컬레이션 메커니즘은 이벤트가 정의된 범위를 벗어날 경우 수행할 조치를 정할 때 사용됩니다.
- 에스컬레이션 장려: 성과 실현에 실패할 위험이 있는 경우 의사 결정권자와 이해관계자에게 문제를 에스컬레이션하는 메커니즘이 마련되어 있으며 팀원은 이를 적극적으로 활용해야 합니다. 위험을 식별하고 인시던트를 방지할 수 있도록 에스컬레이션을 조기에 자주 수행해야 합니다.
- 시기 적절하고 명확하며 실행 가능한 커뮤니케이션: 알려진 위험과 예정된 이벤트를 팀원에게 적시에 알리는 데 사용되는 메커니즘이 마련되어 있습니다. 조치가 필요한지 여부와 어떤 조치가 필요한지 판단하고 적시에 조치를 취할 수 있도록 필요한 컨텍스트, 세부 정보 및 시간(가능한 경우)이 제공됩니다. 예를 들어 소프트웨어 취약성에 대한 알림을 제공하여 패치를 신속하게 적용하도록 하거나, 예정된 판매 프로모션에 대한 알림을 제공하여 서비스 중단의 위험을 방지하기 위한 변경 동결 기능을 구현하게 할 수 있습니다.
- 실험 장려: 실험은 학습을 가속화하고 팀원의 관심과 참여를 유지합니다. 원치 않는 결과가 나와도 성공하지 못하는 경로를 알게 되었으므로 실험에는 성공한 것입니다. 원치 않는 결과가 나온 성공한 실험에 대해 팀원에게 불이익을 가하지 않습니다. 혁신과 아이디어를 실현하려면 실험이 필요합니다.
- 팀원의 기술 유지와 증진 지원 및 장려: 팀은 새로운 기술을 도입하고 워크로드 지원 책임과 수요 변화를 지원하기 위해 기술 역량을 키워야 합니다. 새로운 기술 영역의 기술 증진은 팀원의 만족도를 높이고 혁신을 뒷받침합니다. 발전하는 기술을 검증하고 인증하는 업계 자격증을 획득하고 관리하도록 팀원을 독려합니다. 지식이 효과적으로 전달되도록 하고, 제도적 지식을 갖춘 경험 많고 숙련된 직원을 잃은 경우에 중대한 영향이 발생할 위험을 줄일 수 있도록 교차 교육을 실시합니다. 학습을 위해 체계적으로 정해진 교육 시간을 제공합니다.
- 리소스 팀 적절히 관리: 워크로드 요구 사항을 지원할 수 있도록 팀원 역량을 유지하고 도구와 리소스를 제공합니다. 과중한 업무를 수행하는 팀원은 인적 오류로 인한 인시던트의 위험이 큽니다. 자주 실행하는 활동을 자동화하는 등 도구 및 리소스에 투자하면 팀의 효율성이 높아져 팀원이 더 많은 활동을 지원할 수 있게 됩니다.
- 다양한 의견을 장려하고 팀 내외에서 찾습니다.: 조직 간의 다양성을 활용하여 여러 가지 고유한 관점을 모색합니다. 이러한 관점을 통해 혁신을 증진하고, 기존의 추정 사항에 의문을 제기하며, 확증 편향의 위험을 줄일 수 있습니다. 팀 내에서 포용성, 다양성 및 접근성을 높여 유익한 관점을 확보합니다.

준비

OPS 4 어떻게 운영 상태를 파악할 수 있도록 워크로드를 설계하십니까?

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

모범 사례:

- 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 대응이 필요한 경우를 확인합니다.
- 워크로드 원격 측정 구현 및 구성: 내부 상태와 현재 상태 관련 정보를 내보내도록 워크로드를 설계하고 구성합니다. API 호출 현황, HTTP 상태 코드, 스케일링 이벤트 등의 정보를 내보낼 수 있습니다. 이 정보를 사용하면 대응이 필요한 경우를 확인할 수 있습니다.
- 사용자 활동 원격 측정 구현: 사용자 활동 관련 정보(예: 클릭 스트림 또는 시작/중단/완료된 트랜잭션)를 생성하도록 애플리케이션 코드를 설계합니다. 이 정보를 사용하면 애플리케이션 사용 방법과 사용 패턴을 파악하고 대응이 필요한 경우를 확인할 수 있습니다.
- 종속성 원격 측정 구현: 종속된 리소스의 상태에 대한 정보(도달 가능성 또는 응답 시간)를 생성하도록 워크로드를 설계하고 구성합니다. 외부 종속성의 예로는 외부 데이터베이스, DNS, 네트워크 연결 등이 있습니다. 이 정보를 사용하여 대응이 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 대응이 필요한 경우를 확인하고 문제의 원인을 파악할 수 있습니다.

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

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

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. 프로덕션 환경에 배포하는 단계에 가까워질수록 제어 수준을 높이면 배포되었을 때 워크로드가 의도한 대로 작동할 것이라는 신뢰성을 높일 수 있습니다.
- 자주 발생하고 되돌릴 수 있는 사소한 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 6 배포 위험을 최소화하기 위해 어떻게 노력하나요?

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 7 서비스 운영을 지원할 준비가 되어있는지를 어떻게 알 수 있나요?

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 요구 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.
- 런북을 사용하여 절차 수행:: 런북은 특정 결과를 달성하기 위한 문서화된 절차입니다. 절차를 런북으로 문서화하면 적절하게 파악한 이벤트에 일관된 방식으로 신속하게 대응할 수 있습니다. 런북을 코드로 구현하고, 해당하는 경우 이벤트 대응 과정에서 런북 실행을 트리거하면 일관성을 유지하고, 대응 속도를 높이고, 수동 프로세스에서 발생하는 오류를 줄일 수 있습니다.
- 플레이북을 사용하여 문제 조사: 잘 알려지지 않은 문제에 일관되고 신속하게 대응할 수 있도록 플레이북에 조사 프로세스를 문서화합니다. 플레이북은 장애 시나리오에 영향을 미치는 요인을 식별하기 위해 수행되는 사전 정의된 단계입니다. 문제가 확인되거나 에스컬레이션될 때까지 각 프로세스 단계의 결과를 사용하여 다음에 수행할 단계를 결정합니다.
- 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 8 워크로드가 정상인지 어떻게 판단하나요?

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

모범 사례:

- 핵심 성과 지표 파악: 원하는 비즈니스 성과(예: 주문율, 고객 유지율, 이익 및 운영 지출 비교)과 고객 성과(예: 고객 만족도)를 기반으로 KPI(핵심 성과 지표)를 파악합니다. 그리고 KPI를 평가하여 워크로드의 성공 여부를 결정합니다.
- 워크로드 지표 정의: KPI 달성(예: 주문하지 않은 장바구니, 제출된 주문, 비용, 가격 및 할당된 워크로드 지출)을 측정하도록 워크로드 지표를 정의합니다. 워크로드 상태(예: 인터페이스 응답 시간, 오류 발생률, 제출된 요청, 완료된 요청 및 사용률)를 측정하도록 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. 개선, 조사 및 개입을 위한 임계값을 파악합니다.
- 워크로드의 예상 활동 패턴 파악: 필요한 경우 적절히 대응할 수 있도록 비정상적인 동작을 식별할 워크로드 활동 패턴을 설정합니다.
- 워크로드 성과가 위험한 상태이면 알림 생성:: 워크로드 성과가 위험한 상태이면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.
- 워크로드 이상이 감지되면 알림 생성: 워크로드에서 이상이 감지되면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.
- 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 9 운영 업무가 정상인지 어떻게 판단하나요?

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

모범 사례:

- 핵심 성과 지표 파악: 원하는 비즈니스 성과(예: 새로운 기능 제공)와 고객 성과(예: 고객 지원 사례)를 기반으로 KPI(핵심 성과 지표)를 파악합니다. 그리고 KPI를 평가하여 운영의 성공 여부를 결정합니다.
- 운영 지표 정의: KPI 성과(예: 성공한 배포와 실패한 배포)를 측정하는 데 사용할 운영 지표를 정의합니다. 운영 활동 상태(예: 인시던트의 MTTD(평균 탐지 시간) 및 인시던트의 MTTR(평균 복구 시간))를 측정하는 데 사용할 운영 지표를 정의합니다. 그런 다음, 해당 지표를 평가해 운영 과정에서 적절한 성과를 달성할 수 있는지를 확인하고 운영 활동 상태를 파악합니다.
- 운영 지표 수집 및 분석: 지표를 정기적으로 사전 예방 차원에서 점검하여 추세를 확인하고 어느 부분에 적절한 대응이 필요한지 파악합니다.
- 운영 지표 기준 설정: 지표의 기준을 설정해 성능이 기준보다 높은/낮은 운영 활동을 확인하고 각 프로세스의 성능을 비교할 수 있는 기준으로 필요한 값을 제공합니다.
- 운영의 예상 활동 패턴 파악: 필요한 경우 적절하게 대응할 수 있도록 비정상적인 활동을 식별할 운영 활동 패턴을 설정합니다.
- 운영 성과가 위험한 상태이면 알림 생성: 운영 성과가 위험한 상태이면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.
- 운영 이상이 감지되면 알림 생성: 운영에서 이상이 감지되면 필요 시 적절히 대응할 수 있도록 알림을 생성합니다.
- 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 10 서비스/운영 이벤트를 어떻게 관리하나요?

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.
- 알림별 프로세스 마련: 경계심을 갖는 이벤트가 있는 경우, 특정하게 식별된 소유자를 지정함과 동시에 명확하게 정의된 대응 방법(런북 또는 지침서)을 마련합니다. 이렇게 하면 운영 이벤트에 빠르고 효과적으로 대응할 수 있으며 중요하지 않은 알림 때문에 실행 가능한 이벤트를 제대로 확인하지 못하는 상황을 방지할 수 있습니다.
- 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정: 여러 이벤트에 대해 조치를 취해야 할 때는 실무에 가장 큰 영향을 주는 이벤트를 먼저 해결해야 합니다. 예를 들어 이러한 영향에는 사망 또는 부상, 재정적 손실, 평판 또는 신뢰의 손상이 포함될 수 있습니다.
- 에스컬레이션 경로 정의: 에스컬레이션을 트리거하는 요소와 에스컬레이션 절차를 포함한 에스컬레이션 경로를 런북과 플레이북에 정의합니다. 운영 이벤트에 즉시 효율적으로 대응할 수 있도록 각 작업의 소유자를 구체적으로 명시합니다.
- 푸시 알림 활성화: 사용 중인 서비스가 이벤트의 영향을 받을 때와 정상 작동 상태로 다시 되돌아갈 때 사용자에게 이메일이나 SMS 등을 통해 직접 알립니다. 그러면 사용자가 적절한 조치를 취할 수 있습니다.
- 대시보드를 통한 커뮤니케이션 상태: 목표 대상(예: 내부 기술 팀, 리더십 및 고객)에게 맞춘 대시보드를 제공하여 비즈니스의 현재 운영 상태를 알리고 관심 있는 지표를 제공합니다.
- 이벤트 대응 자동화: 이벤트 대응을 자동화하면 수동 프로세스에서 발생하는 오류를 줄일 수 있으며 일관된 방식으로 즉시 대응할 수 있습니다.

개선

OPS 11 운영을 어떻게 개선 시키나요?

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

모범 사례:

- 지속적인 개선을 위한 프로세스 마련:: 개선 기회를 정기적으로 평가하고 우선순위를 지정해 가장 큰 이점을 얻을 수 있는 영역에서 작업을 집중적으로 수행합니다.
- 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 기여 요인과 교정 조치를 전달합니다.
- 피드백 루프 구현: 개선이 필요한 영역과 문제를 확인할 수 있도록 절차와 워크로드에 피드백 루프를 포함합니다.
- 지식 관리 수행: 팀원이 적시에 원하는 정보를 검색하고 액세스하며, 최신 상태의 완전한 정보인지 식별하는 메커니즘을 제공합니다. 필요한 콘텐츠, 갱신이 필요한 콘텐츠, 더 이상 참조되지 않도록 보관해야 하는 콘텐츠를 식별하는 메커니즘도 제공합니다.
- 개선 추진 요인 정의: 개선 기회를 평가하고 우선순위를 지정할 수 있도록 개선 추진 요인을 파악합니다.
- 인사이트 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 배운 내용을 공유할 수 있습니다.
- 파악한 내용 문서화 및 공유: 운영 활동 실행 과정에서 파악한 내용을 문서화하고 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

보안

SEC 1 워크로드를 안전하게 운영하려면 어떻게 해야 합니까?

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

모범 사례:

- 계정을 사용하는 별도의 워크로드: 회사의 보고 구조를 미러링하는 대신 기능 또는 공통 제어 집합을 기반으로 별도의 계정 및 그룹 계정에서 워크로드를 구성합니다. 워크로드가 증가함에 따라 조직에서 공통 가드레일을 설정할 수 있도록 보안 및 인프라를 염두에 두고 시작해야 합니다.
- 보안 AWS 계정: 예를 들어 MFA를 활성화하여 계정에 안전하게 액세스하고, 루트 사용자의 사용을 제한하며, 계정 연락처를 구성합니다.
- 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.
- 최신 보안 위협 정보 파악: 적절한 제어 수단을 정의하고 구현하는 데 도움이 되도록 최신 보안 위협 정보를 바탕으로 공격 벡터를 파악합니다.
- 최신 보안 권장 사항 파악: 워크로드의 보안 태세를 강화하기 위해 최신 AWS 및 업계 보안 권장 사항을 모두 파악합니다.
- 파이프라인에서 보안 제어의 테스트 및 검증 자동화: 빌드, 파이프라인 및 프로세스의 일부로서 테스트 및 검증되는 보안 메커니즘에 대한 보안 기준과 템플릿을 설정합니다. 도구와 자동화를 사용하여 모든 보안 제어를 지속적으로 테스트하고 검증합니다. 예를 들어 시스템 이미지와 인프라 같은 항목을 코드 템플릿으로 삼아 단계마다 설정된 기준에 따라 보안 취약성, 불규칙성, 드리프트를 검사합니다.
- 보안 위협 모델을 사용하여 위험 파악 및 우선순위 지정: 보안 위협 모델을 사용하여 가장 최근에 등록된 잠재적 위협을 파악하고 유지 관리합니다. 위협 우선순위를 지정하고 보안 제어를 조정하여 방지하고, 탐지하고, 대응합니다. 진화하는 보안 환경에 맞춰 위협 모델을 재검토하고 유지 관리합니다.
- 새로운 보안 서비스 및 기능을 정기적으로 평가 및 구현: AWS와 APN 파트너는 워크로드의 보안 태세를 개선할 수 있는 새로운 기능과 서비스를 지속적으로 릴리스합니다.

자격 증명 및 액세스 관리

SEC 2 사람과 시스템에 대한 자격 증명은 어떻게 관리합니까?

안전한 AWS 워크로드 운영에 접근할 때 관리해야 하는 두 가지 유형의 자격 증명이 있습니다. 액세스 권한을 관리하고 부여하는 데 필요한 자격 증명의 유형을 이해하면 적절한 자격 증명이 적절한 조건에서 적절한 리소스에 액세스할 수 있도록 보장할 수 있습니다. 인적 자격 증명: 관리자, 개발자, 운영자 및 최종 사용자가 AWS 환경 및 애플리케이션에 액세스하려면 자격 증명이 필요합니다. 이들은 조직의 구성원이거나 협업하는 외부 사용자로, 웹 브라우저, 클라이언트 애플리케이션 또는 대화형 명령줄 도구를 통해 AWS 리소스와 상호작용합니다. 시스템 자격 증명: 서비스 애플리케이션, 운영 도구 및 워크로드에서 AWS 서비스에 요청을 하려면(예: 데이터 읽기) 자격 증명이 필요합니다. 이러한 자격 증명에는 Amazon EC2 인스턴스 또는 AWS Lambda 함수와 같이 AWS 환경에서 실행되는 시스템이 포함됩니다. 액세스가 필요한 외부 당사자의 시스템 자격 증명을 관리할 수도 있습니다. 또한 AWS 환경에 액세스해야 하는 시스템이 AWS 외부에 있을 수도 있습니다.

모범 사례:

- 강력한 로그인 메커니즘 사용: 최소 암호 길이를 적용하고, 사용자에게 일반적인 암호나 재사용된 암호를 사용하지 않도록 교육합니다. 추가 제어 계층을 제공하기 위해 소프트웨어 또는 하드웨어 메커니즘을 사용하여 MFA(Multi-Factor Authentication)를 적용합니다.
- 임시 자격 증명 사용: 임시 자격 증명을 동적으로 획득하려면 자격 증명(ID)이 필요합니다. 인력 자격 증명의 경우 AWS Single Sign-On을 사용하거나 IAM 역할과의 연동을 사용하여 AWS 계정에 액세스합니다. 시스템 자격 증명의 경우 장기 액세스 키 대신 IAM 역할을 사용해야 합니다.
- 안전하게 보안 암호 저장 및 사용: 타사 애플리케이션에 대한 암호와 같은 보안 암호가 필요한 인력 및 시스템 자격 증명의 경우 전문적 서비스의 최신 산업 표준을 사용하여 자동 순환을 통해 보안 암호를 저장합니다.
- 중앙 집중식 자격 증명 공급자 사용: 인력 자격 증명의 경우 중앙 집중식 위치에서 자격 증명을 관리할 수 있는 자격 증명 공급자를 사용합니다. 이를 통해 단일 위치에서 액세스 권한을 생성, 관리 및 취소할 수 있으므로 액세스를 더 쉽게 관리할 수 있습니다. 이렇게 하면 여러 자격 증명에 대한 요구 사항이 줄어들고 HR 프로세스와 통합할 기회를 얻을 수 있습니다.
- 정기적으로 자격 증명 감사 및 교체: 임시 자격 증명을 사용할 수 없으며 장기 자격 증명이 필요한 경우 자격 증명을 감사하여 정의된 제어(예: MFA)가 적용되고 정기적으로 교체되며 적절한 액세스 수준을 보유하는지 확인합니다.
- 사용자 그룹 및 속성 활용: 공통 보안 요구 사항이 있는 사용자들을 자격 증명 공급자가 정의한 그룹에 배치하고, 액세스 제어에 사용할 수 있는 사용자 속성(예: 부서 또는 위치)이 정확하고 업데이트되었는지 확인하는 메커니즘을 적절히 설정합니다. 개별 사용자가 아닌 이러한 그룹 및 속성을 사용하여 액세스를 제어합니다. 이를 통해 사용자의 액세스 권한을 변경해야 할 때 여러 개별 정책을 업데이트하는 대신 사용자의 그룹 멤버십 또는 속성을 한 번 변경하여 중앙에서 액세스를 관리할 수 있습니다.

SEC 3 사람과 시스템에 대한 권한은 어떻게 관리합니까?

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

모범 사례:

- 액세스 요구 사항 정의: 관리자, 최종 사용자 또는 기타 구성 요소는 워크로드의 각 구성 요소 또는 리소스에 액세스해야 합니다. 누가 혹은 무엇이 각 구성 요소 또는 리소스에 액세스할 수 있는지 명확하게 정의한 다음 적절한 자격 증명 유형과 인증 및 권한 부여 방법을 선택해야 합니다.
- 최소 권한 액세스 부여: 특정 조건에서 특정 AWS 리소스에 대한 특정 작업과 관련한 액세스를 허용하여 자격 증명에 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 개발자를 그룹에서 제거했을 때 액세스 정책을 변경할 필요 없이 해당 그룹을 액세스 제어에 사용한 모든 리소스에서 이 개발자에 대한 액세스가 취소됩니다.
- 긴급 액세스 프로세스 설정: 가능성은 작지만 자동화된 프로세스 또는 파이프라인에 문제가 발생할 때 워크로드에 긴급 액세스할 수 있게 하는 프로세스입니다. 이 긴급 액세스 프로세스를 통해 최소 권한 액세스를 사용할 수 있습니다. 그러나 사용자가 적절한 수준의 액세스 권한이 필요할 때는 해당 권한을 얻을 수 있습니다. 예를 들어 관리자가 사용자의 액세스 권한 요청을 확인하고 승인하는 프로세스를 설정할 수 있습니다.
- 지속적으로 권한 축소: 팀 및 워크로드가 필요한 액세스 권한을 결정할 때 더 이상 사용하지 않는 권한을 제거하고 최소 권한을 부여하기 위한 검토 프로세스를 설정합니다. 사용하지 않는 자격 증명 및 권한을 지속적으로 모니터링하고 줄입니다.
- 조직에 대한 권한 가드레일 정의: 조직의 모든 자격 증명에 대한 액세스를 제한하는 공통 제어를 설정합니다. 예를 들어 특정 AWS 리전에 대한 액세스를 제한하거나 중앙 보안 팀에 사용되는 IAM 역할과 같은 공통 리소스를 운영자가 삭제하지 못하게 할 수 있습니다.
- 수명 주기에 따라 액세스 관리: 액세스 제어를 운영자 및 애플리케이션 수명 주기/중앙 집중식 연동 공급자와 통합합니다. 예를 들어 조직에서 나가거나 역할이 변경될 때 사용자의 액세스 권한을 제거합니다.
- 퍼블릭 및 교차 계정 액세스 분석: 퍼블릭 및 교차 계정 액세스를 강조하는 결과를 지속적으로 모니터링합니다. 이 유형의 액세스가 필요한 리소스 전용 퍼블릭 액세스 및 교차 계정 액세스를 줄입니다.
- 안전하게 리소스 공유: 계정 전체에 걸쳐 또는 AWS 조직 내에서 공유된 리소스의 소비를 관리합니다. 공유 리소스를 모니터링하고 공유 리소스 액세스를 검토합니다.

탐지

SEC 4 보안 관련 이벤트를 어떻게 감지하나요?

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

모범 사례:

- 서비스 및 애플리케이션 로깅 구성: 워크로드 전반에 걸쳐 애플리케이션 로그, 리소스 로그 및 AWS 서비스 로그를 포함한 로깅을 구성합니다. 예를 들어 조직 내 모든 계정에 대해 AWS CloudTrail, Amazon CloudWatch Logs, Amazon GuardDuty 및 AWS Security Hub가 활성화되어 있는지 확인합니다.
- 로그, 결과 및 지표를 중앙에서 분석: 모든 로그, 지표 및 원격 측정 결과를 중앙에서 수집한 다음 자동으로 분석하여 무단 활동을 나타내는 이상 상태나 지표를 탐지합니다. 대시보드를 사용하면 실시간 상태에 대한 인사이트를 손쉽게 얻을 수 있습니다. 예를 들어 Amazon GuardDuty 및 Security Hub 로그가 알림 및 분석을 위해 중앙 위치로 전송되도록 합니다.
- 이벤트 대응 자동화: 자동화 기능을 사용하여 이벤트를 조사하고 해결하면 수작업 부담과 인적 오류가 줄어들고 조사 역량을 확대할 수 있습니다. 정기적인 검토는 자동화 도구를 튜닝하고 지속적으로 반복하는 데 도움이 됩니다. 예를 들어 첫 번째 조사 단계를 자동화하여 Amazon GuardDuty 이벤트에 대한 대응을 자동화하고, 이를 반복하여 수작업 점진적으로 없앱니다.
- 조치 가능한 보안 이벤트 구현: 팀에게 전송되고 팀에서 조치를 취할 수 있는 알림을 생성합니다. 팀에서 조치를 취하는 데 필요한 관련 정보가 알림에 포함되도록 합니다. 예를 들어 Amazon GuardDuty 및 AWS Security Hub 알림을 팀으로 전송하여 조치를 취하도록 하거나, 대응 자동화 도구로 해당 알림을 전송하고 자동화 프레임워크의 메시징을 통해 팀에 정보를 제공합니다.

인프라 보호

SEC 5 네트워크 리소스는 어떻게 보호합니까?

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

모범 사례:

- 네트워크 계층 생성: 동일한 연결 요구 사항을 공유하는 구성 요소를 계층으로 그룹화합니다. 예를 들어 인터넷에 액세스할 필요가 없는 VPC의 RDS 데이터베이스 클러스터는 인터넷에 연결되는 경로가 없는 서브넷에 배치해야 합니다. VPC 없이 운영되는 서버리스 워크로드의 경우, 마이크로서비스를 사용한 유사한 계층화 및 세분화를 통해 동일한 목표를 실현할 수 있습니다.
- 모든 계층에서 트래픽 제어: 인바운드 및 아웃바운드 트래픽 모두에 대해 심층 방어 방식을 사용하여 제어 기능을 적용합니다. 예를 들어 Amazon Virtual Private Cloud(VPC)의 경우 이러한 기능에는 보안 그룹, 네트워크 ACL, 서브넷 등이 포함됩니다. AWS Lambda는 VPC 기반 제어 기능을 제공하는 프라이빗 VPC에서 실행하는 것이 좋습니다.
- 네트워크 보호 자동화: 위협 정보 및 이상 상태 감지 결과에 따라 자체 방어 네트워크를 제공하는 보호 메커니즘을 자동화합니다. 예를 들어 최신 위협에 사전 예방적으로 대응하고 위협의 영향을 줄일 수 있는 침입 탐지 및 방지 도구가 있습니다.
- 검사 및 보호 구현: 각 계층에서 트래픽을 검사하고 필터링합니다. 예를 들어 웹 애플리케이션 방화벽을 사용하여 애플리케이션 네트워크 계층에서 실수로 액세스하는 것을 방지할 수 있습니다. Lambda 함수의 경우 타사 도구를 활용하여 런타임 환경에 애플리케이션 계층 방화벽을 추가할 수 있습니다.

SEC 6 컴퓨팅 리소스를 어떻게 보호 하시나요?

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:

- 취약성 관리 수행: 코드, 종속성 및 인프라의 취약성을 자주 스캔하고 패치하여 새로운 위협으로부터 보호합니다.
- 공격 대상 영역 축소: 운영 체제를 강화하고 사용 중인 구성 요소, 라이브러리 및 외부 사용 서비스를 최소화하여 공격 대상 영역을 줄입니다.
- 관리형 서비스 구현: 공유 책임 모델의 일환으로 보안 유지 관리 태스크를 줄일 수 있도록 Amazon RDS, AWS Lambda, Amazon ECS 등 리소스를 관리하는 서비스를 구현합니다.
- 컴퓨팅 보호 자동화: 취약성 관리, 공격 대상 영역 축소, 리소스 관리 등 컴퓨팅 보호 메커니즘을 자동화합니다.
- 사용자가 원격으로 작업을 수행할 수 있도록 지원: 대화형 액세스 기능을 제거하면 인적 오류의 위험과 수동 구성 또는 관리의 필요성을 줄일 수 있습니다. 예를 들어 변경 관리 워크플로를 사용하여 코드형 인프라로 EC2 인스턴스를 배포한 다음, 직접 액세스나 배스천 호스트를 허용하는 것이 아니라 도구를 사용하여 EC2 인스턴스를 관리합니다.
- 소프트웨어 무결성 검증: 워크로드에 사용되는 소프트웨어, 코드, 라이브러리가 신뢰할 수 있는 소스에서 온 것이며 변조되지 않았는지 검증하는 메커니즘(예: 코드 서명)을 구현합니다.

데이터 보호

SEC 7 데이터는 어떻게 분류합니까?

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

모범 사례:

- 워크로드 내 데이터 식별: 여기에는 데이터 유형 및 분류, 관련 비즈니스 프로세스, 데이터 소유자, 적용 가능한 법률 및 규정 준수 요구 사항, 저장 위치, 적용해야 할 결과 제어 항목이 포함됩니다. 여기에는 데이터가 공개적으로 사용 가능한지, 데이터가 PII(고객 개인 식별 정보)처럼 내부에서만 사용되는지, 지적 재산처럼 데이터에 대한 액세스가 더 엄격히 제한되는지, 법적으로 권한이 있는지, 민감한 데이터로 표시되는지 등을 나타내는 분류가 포함될 수 있습니다.
- 데이터 보호 제어 정의: 분류 수준에 따라 데이터를 보호합니다. 예를 들어 관련 권장 사항을 사용하여 공개용으로 분류된 데이터를 보호하면서, 추가 제어 기능을 통해 민감한 데이터를 보호합니다.
- 식별 및 분류 자동화: 데이터 식별 및 분류를 자동화하여 수동 상호 작용으로 인한 인적 오류의 위험을 줄입니다.
- 데이터 수명 주기 관리 정의: 정의된 수명 주기 전략은 민감도 수준과 법률 및 조직 요구 사항을 기준으로 해야 합니다. 데이터 보존 기간, 데이터 폐기, 데이터 액세스 관리, 데이터 변환, 데이터 공유 등의 측면을 고려해야 합니다.

SEC 8 저장된 데이터는 어떻게 보호합니까?

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

모범 사례:

- 보안 키 관리 구현: 암호화 키는 엄격한 액세스 제어를 통해 안전하게 저장해야 합니다. 예를 들어 AWS KMS와 같은 키 관리 서비스를 사용할 수 있습니다. 데이터 분류 수준 및 분리 요구 사항에 맞추어, AWS IAM 및 리소스 정책과 함께 키에 대한 액세스 제어 기능과 여러 키를 사용하는 것이 좋습니다.
- 저장 데이터 암호화 적용: 저장된 데이터를 보호할 수 있도록 최신 표준 및 권장 사항에 따라 암호화 요구 사항을 적용합니다.
- 저장 데이터 보호 자동화: 자동화된 도구를 사용하여 저장 데이터 보호를 지속적으로 검증하고 적용합니다. 예를 들어 암호화된 스토리지 리소스만 있는지 확인합니다.
- 액세스 제어 적용: 저장된 데이터를 보호할 수 있도록 백업, 격리, 버전 관리 등의 메커니즘과 최소 권한을 사용하여 액세스 제어를 적용합니다. 운영자가 데이터에 대한 퍼블릭 액세스 권한을 부여하지 못하게 합니다.
- 사람들이 데이터에 쉽게 액세스할 수 없도록 유지하는 메커니즘 사용: 정상적인 운영 상황에서 모든 사용자가 민감한 데이터와 시스템에 직접 액세스하지 못하도록 합니다. 예를 들어 데이터 스토어에 직접 액세스를 허용하는 대신 대시보드를 제공하여 쿼리를 실행하도록 합니다. CI/CD 파이프라인이 사용되지 않는 경우, 정상적으로 비활성화된 브레이크-글래스 액세스 메커니즘을 적절하게 제공하기 위해 필요한 컨트롤 및 프로세스를 결정합니다.

SEC 9 전송 중인 데이터는 어떻게 보호합니까?

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

모범 사례:

- 보안 키 및 인증서 관리 구현: 암호화 키와 인증서를 안전하게 저장하고 엄격하게 액세스를 제어하면서 적절한 시간 간격으로 교체해야 합니다. 예를 들어 AWS Certificate Manager(ACM)와 같은 인증서 관리 서비스를 사용할 수 있습니다.
- 전송 중 데이터 암호화 적용: 조직/법률/규정 준수 요구 사항을 충족할 수 있도록 적절한 표준 및 권장 사항에 따라 정의된 암호화 요구 사항을 적용합니다.
- 의도하지 않은 데이터 액세스 감지 자동화: GuardDuty와 같은 도구를 사용하여 데이터 분류 수준에 따라 정의된 경계 외부로 데이터를 이동하려는 시도를 자동으로 감지합니다. 예를 들어 DNS 프로토콜을 사용하여 알 수 없거나 신뢰할 수 없는 네트워크로 데이터를 복사하는 트로이 목마를 감지할 수 있습니다.
- 네트워크 통신 인증: TLS(전송 계층 보안) 또는 IPsec과 같은 인증을 지원하는 프로토콜을 사용하여 통신의 자격 증명을 확인합니다.

인시던트 대응

SEC 10 인시던트를 어떻게 예상하고 대응하며 어떻게 사후 복구합니까?

조직의 업무 중단을 최소화할 수 있도록 보안 인시던트를 제때 효고적으로 조사 및 대응하고 사후 복구하려면 철저한 준비가 필요합니다.

모범 사례:

- 주요 직원과 외부 리소스 파악: 조직이 인시던트에 대응하는 데 도움이 될 수 있는 내/외부 직원, 리소스, 법적 의무를 파악합니다.
- 인시던트 관리 계획 개발: 인시던트에 대응하고, 인시던트 중에 커뮤니케이션하고, 인시던트로부터 복구하는 데 도움이 되는 계획을 수립합니다. 예를 들어 워크로드와 조직에서 발생할 가능성이 가장 큰 시나리오부터 인시던트 대응 계획을 시작할 수 있습니다. 내/외부의 커뮤니케이션 및 에스컬레이션 방법도 포함해야 합니다.
- 포렌식 역량 확보: 외부 전문가, 도구 및 자동화를 비롯하여 적합한 포렌식 역량을 파악하고 확보합니다.
- 억제 기능 자동화: 대응 시간과 조직에 대한 영향을 줄일 수 있도록 인시던트 억제 및 복구를 자동화합니다.
- 액세스 권한 사전 프로비저닝: 인시던트 대응자에게 AWS에 사전 프로비저닝된 올바른 액세스 권한을 부여하여 조사 및 복구 시간을 단축할 수 있도록 합니다.
- 도구 사전 배포: AWS에 사전 배포된 적절한 도구를 보안 담당자에게 제공함으로써 조사부터 복구까지 걸리는 시간을 단축할 수 있도록 합니다.
- 게임 데이 실행: 인시던트 대응 게임 데이(시뮬레이션)를 정기적으로 연습하고, 파악한 내용을 인시던트 관리 계획에 통합하고, 지속적으로 개선합니다.

안정성

기반

REL 1 서비스 할당량과 제약 조건은 어떻게 관리합니까?

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

모범 사례:

- 서비스 할당량 및 제약 조건 인식: 워크로드 아키텍처에 대한 기본 할당량이 있으며 할당량 증가를 요청할 수 있습니다. 또한 디스크 또는 네트워크와 같은 리소스 제약 조건은 잠재적인 영향을 미칠 수 있습니다.
- 계정 및 리전 전체에서 서비스 할당량 관리: 여러 AWS 계정 또는 AWS 리전을 사용하는 경우 프로덕션 워크로드가 실행되는 모든 환경에서 적절한 할당량을 요청해야 합니다.
- 아키텍처를 통해 고정된 서비스 할당량 및 제약 조건 수용: 변경할 수 없는 서비스 할당량과 물리적 리소스를 인지하고 안정성에 영향을 미치지 않도록 설계합니다.
- 할당량 모니터링 및 관리: 잠재적 사용량을 평가하고 할당량을 적절히 늘려 사용량 증가를 계획합니다.
- 할당량 관리 자동화: 임계값에 근접했을 때 알림을 받을 수 있는 도구를 구현합니다. AWS Service Quotas API를 사용하여 할당량 증가 요청을 자동화할 수 있습니다.
- 현재의 할당량과 최대 사용량 간에 장애 조치를 수용할 만큼 여유가 충분히 있는지 확인합니다.: 리소스는 장애가 발생하더라도 정상적으로 종료할 때까지는 할당량 계산에 계속 포함될 수 있습니다. 그러므로 할당량에는 장애 발생 리소스가 종료되기 전까지 장애가 발생한 모든 리소스와 교체 리소스의 중복이 포함되어야 합니다. 이 차이를 계산할 때는 가용 영역 장애를 고려해야 합니다.

REL 2 네트워크 토폴로지는 어떻게 계획합니까?

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

모범 사례:

- 워크로드 퍼블릭 엔드포인트에 고가용성 네트워크 연결을 사용합니다.: 이러한 엔드포인트와 엔드포인트로의 라우팅은 고가용성으로 구현해야 합니다. 이러한 고가용성을 달성하려면 고가용성 DNS, CDN(콘텐츠 전송 네트워크), API Gateway, 로드 밸런싱 또는 역방향 프록시를 사용합니다.
- 클라우드와 온프레미스 환경의 프라이빗 네트워크 간에 이중화된 연결을 프로비저닝합니다.: 별도로 배포된 프라이빗 네트워크 간에 여러 AWS Direct Connect(DX) 연결 또는 VPN 터널을 사용합니다. 가용성을 높이려는 경우에는 여러 DX 위치를 사용합니다. 여러 AWS 리전을 사용하는 경우 2개 이상의 AWS 리전에 이중화되도록 해야 합니다. VPN을 종료하는 AWS Marketplace 어플라이언스를 평가해야 할 수 있습니다. AWS Marketplace 어플라이언스를 사용하는 경우 다른 가용 영역에서 고가용성을 위해 중복 인스턴스를 배포합니다.
- 확장 및 가용성을 위한 IP 서브넷 할당 계정을 확인합니다.: 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.
- 다대다 메시보다 허브 앤 스포크 토폴로지를 선호합니다.: 셋 이상의 네트워크 주소 공간(예: VPC와 온프레미스 네트워크)이 VPC 피어링, AWS Direct Connect 또는 VPN을 통해 연결되는 경우 AWS Transit Gateway가 제공하는 것과 같은 허브 앤 스포크 모델을 사용합니다.
- 연결된 모든 프라이빗 주소 공간에서 겹치지 않는 프라이빗 IP 주소 범위를 적용합니다.: VPN을 통해 피어링되거나 연결된 경우 각 VPC의 IP 주소 범위가 겹치지 않아야 합니다. 마찬가지로, VPC와 온프레미스 환경 간의 IP 주소 충돌 또는 사용하는 다른 클라우드 공급자와의 IP 주소 충돌을 방지해야 합니다. 필요한 경우 프라이빗 IP 주소 범위를 할당할 수 있어야 합니다.

워크로드 아키텍처

REL 3 워크로드 서비스 아키텍처는 어떻게 설계합니까?

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

모범 사례:

- 워크로드를 세그먼트화하는 방법 선택: 모놀리식 아키텍처는 피해야 합니다. 대신 SOA와 마이크로서비스 중에서 선택하십시오. 각 아키텍처를 선택할 때는 이점과 복잡성의 균형을 고려합니다. 신제품의 첫 출시에 필요한 것과 워크로드를 확장할 때 필요한 것은 다릅니다. 작은 세그먼트를 사용하면 민첩성, 조직의 유연성 및 확장성이 향상됩니다. 복잡성에는 지연 시간 증가, 디버깅 복합성, 운영 부담 증가가 포함됩니다.
- 특정 비즈니스 도메인 및 기능을 중심으로 서비스 구축: SOA는 비즈니스 요구 사항에 따라 명확하게 정의된 기능으로 서비스를 구축합니다. 마이크로서비스는 도메인 모델과 경계 컨텍스트를 사용하여 이를 추가로 제한함으로써 각 서비스가 한 가지 작업을 수행하도록 합니다. 특정 기능에 집중하면 다양한 서비스의 안정성 요구 사항을 구분하고 보다 구체적으로 투자의 대상을 지정할 수 있습니다. 또한 비즈니스 문제가 간단해지고 각 서비스에 소규모 팀이 연결되므로 조직의 확장이 수월해집니다.
- API별로 서비스 계약 제공: 서비스 계약은 서비스 통합에 대한 팀 간의 문서화된 계약으로, 머신 판독 가능한 API 정의, 속도 제한 및 성능 기대치를 포함합니다. 버전 관리 전략을 사용하면 클라이언트에서 기존 API를 계속 사용하면서 준비가 될 때 애플리케이션을 최신 API로 마이그레이션할 수 있습니다. 계약을 위반하지 않는 한 언제든지 배포가 가능합니다. 서비스 공급자 팀은 원하는 기술 스택을 사용하여 API 계약을 충족할 수 있습니다. 마찬가지로 서비스 소비자는 자체 기술을 사용할 수 있습니다.

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

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

모범 사례:

- 필요한 분산 시스템의 종류 식별: 하드 실시간 분산 시스템은 응답을 동기식으로 신속하게 제공해야 하며, 소프트 실시간 시스템은 몇 분 이상의 보다 관대한 기간에 응답을 제공할 수 있습니다. 오프라인 시스템은 배치 또는 비동기식 처리를 통해 응답을 처리합니다. 하드 실시간 분산 시스템은 안정성 요구 사항이 가장 엄격합니다.
- 약결합 종속성 구현:: 대기열 처리 시스템, 스트리밍 시스템, 워크플로 및 로드 밸런서와 같은 종속성은 약결합됩니다. 약한 결합은 한 구성 요소의 동작을 다른 종속 구성 요소에서 분리하여 복원력 및 민첩성을 높이는 데 도움이 됩니다.
- 모든 응답의 멱등성 유지: 멱등성이 있는 서비스는 각 요청이 정확히 한 번만 완료되도록 합니다. 이렇게 하면 다수의 동일한 요청에서 단일 요청과 동일한 결과가 나옵니다. 멱등성이 있는 서비스를 사용하면 클라이언트가 요청이 오류로 여러 번 처리될 것이라는 염려 없이 재시도를 시행할 수 있습니다. 이를 위해 클라이언트는 멱등성 토큰을 사용하여 API 요청을 실행할 수 있으며 요청이 반복될 때마다 동일한 토큰이 사용됩니다. 멱등성이 있는 서비스 API는 토큰을 사용하여 요청이 처음 완료되었을 때 반환된 응답과 동일한 응답을 반환합니다.
- 일정한 작업 처리: 대규모 로드가 급속도로 변경되면 시스템에서 장애가 발생할 수 있습니다. 예를 들어 서버 수천 대의 상태를 모니터링하는 상태 확인 시스템은 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.

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

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

모범 사례:

- 관련 하드 종속성을 소프트 종속성으로 변환하는 정상적인 성능 저하 구현: 구성 요소의 종속성이 비정상 상태인 경우에도 구성 요소 자체가 성능이 저하된 방식으로 작동할 수 있습니다. 예를 들어 종속성 호출이 실패하면 미리 결정된 정적 응답으로 장애 조치됩니다.
- 요청 조절: 예기치 않은 수요 증가에 대응하기 위한 완화 패턴입니다. 일부 요청은 처리되지만 정의된 제한을 초과하는 요청은 거부되고 병목 현상이 발생했음을 나타내는 메시지를 반환합니다. 클라이언트는 요청을 다시 중단하고 중단하거나 더 느린 속도로 다시 시도할 것이라는 기대입니다.
- 재시도 호출 제어 및 제한: 지수 백오프를 사용하여 점진적으로 더 긴 간격 후에 다시 시도합니다. 지터를 도입하여 이러한 재시도 간격을 무작위로 지정하고 최대 재시도 횟수를 제한합니다.
- 빠른 실패 및 대기열 제한: 워크로드가 요청에 성공적으로 응답할 수 없는 경우 빠르게 실패합니다. 이렇게 하면 요청에 연결된 리소스를 해제할 수 있고 리소스가 부족한 경우 서비스 복구를 허용할 수 있습니다. 워크로드가 성공적으로 응답할 수 있지만 요청 속도가 너무 높은 경우에는 대기열을 사용하여 요청을 버퍼링합니다. 하지만 긴 대기열을 허용하지 마십시오. 대기열이 길면 클라이언트가 이미 포기한 무효 요청을 처리할 수 있습니다.
- 클라이언트 시간 제한 설정: 시간 제한을 적절히 설정하고, 체계적으로 확인합니다. 기본값을 사용해서는 안 됩니다. 기본값은 일반적으로 너무 높게 설정됩니다.
- 가능한 경우 서비스를 상태 비저장으로 설계: 서비스는 상태를 필요로 하지 않거나 디스크 또는 메모리의 로컬로 저장된 데이터에 종속성이 없도록 서로 다른 클라이언트 요청 간에 상태를 오프로드해야 합니다. 이렇게 하면 가용성에 미치는 영향 없이 서버를 대체할 수 있습니다. Amazon ElastiCache 또는 Amazon DynamoDB는 오프로드된 상태에 적합한 대상입니다.
- 비상 레버 구현: 이 프로세스는 워크로드의 가용성에 미치는 영향을 신속하게 완화할 수 있는 프로세스입니다. 이 프로세스는 근본 원인이 없는 상태에서 작동할 수 있습니다. 이상적인 비상 레버는 완전히 결정적인 활성화 및 비활성화 기준을 제공하여 확인자에 대한 인지 부담을 0으로 줄여줍니다. 레버의 예로는 모든 로봇 트래픽을 차단하거나 정적 응답을 제공하는 것이 있습니다. 레버는 수동인 경우가 많지만 자동화할 수도 있습니다.

변경 관리

REL 6 워크로드 리소스는 어떻게 모니터링합니까?

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

모범 사례:

- 워크로드의 모든 구성 요소 모니터링(생성): Amazon CloudWatch 또는 타사 도구를 사용하여 워크로드의 구성 요소를 모니터링합니다. Personal Health Dashboard로 AWS 서비스를 모니터링합니다.
- 지표 정의 및 계산(집계): 로그 데이터를 저장하고 필요한 경우 필터를 적용하여 특정 로그 이벤트 수 또는 로그 이벤트 타임스탬프에서 계산된 지연 시간과 같은 지표를 계산합니다.
- 알림 전송(실시간 처리 및 경보): 중요한 이벤트가 발생할 때 알아야 하는 조직에 알림이 전송됩니다.
- 응답 자동화(실시간 처리 및 경보): 이벤트가 감지되면 자동화를 사용하여 실패한 구성 요소를 대체하는 등의 조치를 취합니다.
- 저장 및 분석: 로그 파일 및 지표 기록을 수집하고 이를 분석하여 더 광범위한 추세 및 워크로드 인사이트를 확보합니다.
- 정기적인 검토 시행: 워크로드 모니터링이 구현되는 방식을 자주 검토하고 중요한 이벤트 및 변경 사항에 따라 업데이트합니다.
- 시스템을 통한 요청의 종단 간 추적 모니터링: 개발자는 AWS X-Ray 또는 타사 도구를 사용하여 분산 시스템을 보다 쉽게 분석하고 디버깅하여 애플리케이션과 기반 서비스의 성능을 파악할 수 있습니다.

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

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

모범 사례:

- 리소스를 확보하거나 조정할 때 자동화 사용: 손상된 리소스를 교체하거나 워크로드를 조정할 때 Amazon S3 및 AWS Auto Scaling과 같은 관리형 AWS 서비스를 사용하여 프로세스를 자동화합니다. 타사 도구 및 AWS SDK를 사용하여 크기를 자동 조정할 수도 있습니다.
- 워크로드 장애 감지 시 리소스 확보: 가용성이 영향을 받는 경우 필요에 따라 리소스를 사후에 확장하여 워크로드 가용성을 복원합니다.
- 워크로드에 더 많은 리소스가 필요한 것으로 감지되면 리소스를 확보합니다.: 수요를 충족하고 가용성에 영향을 미치지 않도록 리소스를 사전에 확장합니다.
- 워크로드 부하 테스트: 확장 작업이 워크로드의 필요 사항을 충족하는지 측정하기 위한 로드 테스트 방식을 채택하십시오.

REL 8 변경 사항은 어떻게 적용합니까?

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

모범 사례:

- 배포와 같은 표준 활동에 런북 사용: 런북은 특정 결과를 달성하기 위해 미리 정의된 단계입니다. 수동 또는 자동으로 표준 활동을 수행할 때 런북을 사용합니다. 워크로드 배포, 패치 적용 또는 DNS 수정과 같은 활동이 여기에 포함됩니다.
- 배포의 일부로 기능 테스트 통합: 기능 테스트는 자동화된 배포의 일부로 실행됩니다. 성공 기준이 충족되지 않으면 파이프라인이 중지되거나 롤백됩니다.
- 배포의 일부로 복원력 테스트 통합: 복원력 테스트는 카오스 엔지니어링의 일부로 사전 프로덕션 환경에서 자동화된 배포 파이프라인에 포함되어 실행됩니다.
- 변경 불가능한 인프라를 사용하여 배포: 프로덕션 워크로드의 현재 위치에서 업데이트, 보안 패치 또는 구성 변경이 발생하지 않도록 규정하는 모델입니다. 변경이 필요한 경우 아키텍처가 새 인프라에 구축되고 프로덕션 환경에 배포됩니다.
- 자동화를 통한 변경 사항 배포: 배포 및 패치 적용이 자동화되므로 부정적인 영향이 제거됩니다.

장애 관리

REL 9 데이터는 어떻게 백업합니까?

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

모범 사례:

- 백업해야 하는 모든 데이터를 확인하고 백업하거나, 소스에서 데이터를 복제합니다.: Amazon S3는 여러 데이터 원본의 백업 대상으로 사용할 수 있습니다. Amazon EBS, Amazon RDS 및 Amazon DynamoDB와 같은 AWS 서비스에는 백업 생성 기능이 내장되어 있습니다. 타사 백업 소프트웨어를 사용할 수도 있습니다. RPO 충족을 위해 다른 소스에서 데이터를 재현할 수 있는 경우에는 백업이 필요하지 않을 수도 있습니다.
- 백업 보안 및 암호화: AWS IAM과 같은 인증 및 권한 부여를 사용하여 액세스를 감지하고 암호화를 사용하여 데이터 무결성 손상을 감지합니다.
- 자동으로 데이터 백업 수행: 정기적인 일정 또는 데이터 세트의 변경에 따라 백업이 자동으로 생성되도록 구성합니다. RDS 인스턴스, EBS 볼륨, DynamoDB 테이블 및 S3 객체에 대해 모두 자동 백업을 구성할 수 있습니다. AWS Marketplace 솔루션 또는 타사 솔루션을 사용해도 됩니다.
- 백업 무결성 및 프로세스를 확인하기 위해 데이터의 주기적인 복구 수행: 복구 테스트를 수행하여 백업 프로세스 구현이 RTO(복구 시간 목표) 및 RPO(복구 시점 목표)를 충족하는지 검증합니다.

REL 10 장애 격리를 사용하여 워크로드를 보호하려면 어떻게 해야 합니까?

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

모범 사례:

- 워크로드를 여러 위치에 배포: 워크로드 데이터와 리소스를 여러 가용 영역에 분산하거나 필요한 경우 AWS 리전 전체에 분산합니다. 필요에 따라 다양한 위치를 사용할 수 있습니다.
- 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 구현해야 합니다.
- 격벽 아키텍처 사용: 이 패턴은 선박의 격벽처럼 장애를 요청/사용자의 소수의 하위 집합으로 제한하여 손상된 요청 수를 제한하고 대부분의 요청은 오류 없이 계속될 수 있도록 합니다. 데이터에 대한 격벽은 일반적으로 파티션 또는 샤드로 불리며 서비스에 대한 격벽은 셀이라고 합니다.

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

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

모범 사례:

- 워크로드의 모든 구성 요소를 모니터링하여 장애 감지: 워크로드 상태를 지속적으로 모니터링하여 성능 저하 또는 완전한 장애가 발생하는 즉시 수동 및 자동화된 시스템을 통해 이를 인식할 수 있도록 합니다. 비즈니스 가치를 기반으로 KPI(핵심 성능 지표)를 모니터링합니다.
- 영향을 받지 않은 위치의 정상 리소스로 장애 조치: 위치 장애가 발생하는 경우 정상 위치의 데이터 및 리소스로 계속해서 요청을 처리할 수 있어야 합니다. 다중 영역 워크로드의 경우 Elastic Load Balancing 및 AWS Auto Scaling과 같은 AWS 서비스로 가용 영역에 로드를 분산할 수 있으므로 이 작업이 쉽습니다. 다중 리전 워크로드의 경우 이 작업이 더 복잡합니다. 예를 들어 리전 간 읽기 전용 복제본을 사용하여 데이터를 여러 AWS 리전에 배포할 수 있지만, 기본 위치에 장애가 발생할 경우 읽기 전용 복제본을 마스터로 승격하고 트래픽을 이 위치로 지정해야 합니다. Amazon Route 53와 AWS Global Accelerator는 AWS 리전 간에 트래픽을 라우팅하는 데 도움이 됩니다.
- 모든 계층에서 복구 자동화: 장애가 감지되면 자동화된 기능을 사용하여 수정 작업을 수행합니다.
- 정적 안정성을 사용하여 바이모달 동작 방지: 바이모달 동작은 워크로드가 정상 모드와 장애 모드에서 다른 동작을 보이는 것을 말합니다. 예를 들어 가용 영역에 장애가 발생할 경우 새 인스턴스를 시작하는 방법을 사용할 수 있습니다. 그러나 이 방법 대신 정적으로 안정적이며 한 모드에서만 작동하는 워크로드를 구축해야 합니다. 한 AZ가 제거된 경우 제거된 영역의 워크로드 로드를 처리하기에 충분한 인스턴스를 각 가용 영역에 프로비저닝한 다음 Elastic Load Balancing 또는 Amazon Route 53 상태 확인을 사용하여 손상된 인스턴스에서 로드를 이동합니다.
- 이벤트가 가용성에 영향을 미치는 경우 알림 전송: 중대한 이벤트가 감지되면 이벤트로 인해 야기된 문제가 자동으로 해결된 경우에도 알림이 전송됩니다.

REL 12 안정성은 어떻게 테스트합니까?

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

모범 사례:

- 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.
- 인시던트 사후 분석 수행: 고객에게 영향을 주는 이벤트를 검토하고 발생 요인과 예방 조치 항목을 식별합니다. 이 정보를 사용하여 재발을 제한하거나 방지하는 완화 기능을 개발합니다. 신속하고 효과적인 대응을 위한 절차를 개발합니다. 목표 대상에 맞게 적절히 원인과 교정 조치를 전달합니다. 필요한 경우 다른 관계자들에게 이러한 원인을 전달하는 방법을 마련합니다.
- 기능 요구 사항 테스트: 여기에는 단위 테스트와 필수 기능을 검증하는 통합 테스트가 포함됩니다.
- 확장 및 성능 요구 사항 테스트: 여기에는 워크로드가 조정 및 성능 요구 사항을 충족하는지 확인하기 위한 로드 테스트가 포함됩니다.
- 카오스 엔지니어링을 이용한 복원력 테스트: 사전 프로덕션 및 프로덕션 환경에 정기적으로 장애를 주입하는 테스트를 실행합니다. 워크로드에서 장애에 대응할 방법에 대한 가설을 테스트 결과와 비교하고 일치하지 않는 경우 반복합니다. 프로덕션 테스트가 사용자에게 영향을 미치지 않는지 확인합니다.
- 정기적으로 게임 데이 진행: 실제 장애 시나리오에 참여할 직원들과 함께 실전 연습을 정기적으로 수행하여 프로덕션에 최대한 근접한 장애 절차(프로덕션 환경의 장애 절차 포함)를 연습합니다. 실전 연습에서는 프로덕션 테스트가 사용자에게 영향을 미치지 않도록 하는 조치가 시행됩니다.

REL 13 DR(재해 복구)를 어떻게 계획합니까?

DR 전략의 시작은 백업 및 중복 워크로드 구성 요소를 갖추는 것입니다. RTO 및 RPO는 가용성 복원에 대한 목표입니다. 비즈니스 요구 사항에 따라 이러한 목표를 설정하십시오. 워크로드 리소스 및 데이터의 위치와 기능을 고려하여 이러한 목표를 충족하는 전략을 구현합니다.

모범 사례:

- 가동 중단 시간 및 데이터 손실 시의 복구 목표 정의: 워크로드에는 RTO(복구 시간 목표) 및 RPO(복구 시점 목표)가 있습니다.
- 복구 목표 달성을 위해 정의된 복구 전략 사용: 목표를 달성하기 위한 DR(재해 복구) 전략이 정의되었습니다.
- 재해 복구 구현을 테스트하여 구현 확인: DR에 대한 장애 조치를 정기적으로 테스트하여 RTO와 RPO를 충족하는지 확인합니다.
- DR 사이트 또는 리전에서 구성 드리프트 관리: DR 사이트 또는 리전에 필요한 인프라, 데이터 및 구성이 갖추어져 있는지 확인합니다. 예를 들어 AMI와 서비스 할당량이 최신 상태인지 확인합니다.
- 복구 자동화: AWS 또는 타사 도구를 사용하여 시스템 복구를 자동화하고 트래픽을 DR 사이트 또는 리전으로 라우팅합니다.

성능 효율성

선택

PERF 1 최고의 성능을 제공하는 아키텍처를 선택하려면 어떻게 해야 합니까?

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

모범 사례:

- 사용 가능한 서비스 및 리소스 파악: 클라우드에서 사용 가능한 방대한 서비스와 리소스에 대해 알아보고 이해합니다. 워크로드와 관련이 있는 서비스 및 구성 옵션을 확인하고, 성능을 최적화하는 방법을 파악합니다.
- 아키텍처를 선택하는 프로세스 정의: 클라우드에 대한 내부 경험과 지식을 사용하거나 게시된 사용 사례, 관련 설명서 또는 백서 등의 외부 리소스를 사용하여 리소스 및 서비스를 선택하는 프로세스를 정의합니다. 워크로드에 사용할 수 있는 서비스를 사용한 실험과 벤치마킹을 장려하는 프로세스를 정의해야 합니다.
- 비용 요구 사항을 의사 결정에 반영합니다. : 워크로드에는 작업에 대한 비용 요구 사항이 있는 경우가 많습니다. 내부 비용 제어 기능을 사용하여 예상되는 리소스 요구 사항에 적합한 유형 및 크기의 리소스를 선택하십시오.
- 정책 또는 참조 아키텍처 사용: 내부 정책 및 기존 참조 아키텍처를 평가하고 분석을 사용하여 워크로드에 사용할 서비스 및 구성을 선택하면 성능 및 효율성을 극대화할 수 있습니다.
- 클라우드 공급자 또는 해당하는 파트너의 지침 사용: 클라우드 회사 리소스(예: 솔루션스 아키텍트, 전문 서비스 또는 적절한 파트너)를 의사 결정의 지침으로 사용하십시오. 이러한 리소스는 최적의 성능을 위해 아키텍처를 검토하고 개선하는 데 도움이 될 수 있습니다.
- 기존 워크로드 벤치마크: 기존 워크로드의 성능을 벤치마크하여 클라우드에서의 성능을 파악합니다. 이러한 벤치마크에서 수집된 데이터를 사용하면 아키텍처를 원활하게 결정할 수 있습니다.
- 워크로드에 대한 로드 테스트: 다양한 리소스 유형 및 크기의 최신 워크로드 아키텍처를 클라우드에 배포합니다. 배포를 모니터링하여 병목 현상 또는 초과 용량을 식별하는 성능 지표를 캡처합니다. 이 성능 정보를 사용하여 아키텍처 및 리소스 선택을 설계하거나 개선할 수 있습니다.

PERF 2 컴퓨팅 솔루션을 어떻게 선택합니까?

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

모범 사례:

- 사용 가능한 컴퓨팅 옵션 평가: 사용 가능한 컴퓨팅 관련 옵션의 성능 특성을 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: 컴퓨팅 시스템 성능을 가장 효율적으로 파악하는 방법 중 하나는 다양한 리소스의 실제 사용률을 기록하고 추적하는 것입니다. 이 데이터를 사용하면 리소스 요구 사항을 보다 정확하게 파악할 수 있습니다.
- 적절하게 크기를 조정하여 필요한 구성 확인: 워크로드의 다양한 성능 특성, 그리고 이러한 특성과 메모리/네트워크/CPU 사용량 간의 관계를 분석합니다. 이 데이터를 사용하면 워크로드 프로필에 가장 적합한 리소스를 선택할 수 있습니다. 예를 들어 데이터베이스와 같은 메모리 집약적 워크로드는 r 패밀리 인스턴스로 처리하는 것이 가장 좋습니다. 하지만 버스트 워크로드의 경우 탄력적인 컨테이너 시스템을 사용하는 것이 더 유리할 수 있습니다.
- 사용 가능한 리소스 탄력성 사용: 클라우드는 수요 변화에 맞춰 다양한 메커니즘을 통해 리소스를 동적으로 확장 또는 축소할 수 있는 유연성을 제공합니다. 컴퓨팅 관련 지표를 함께 활용하는 경우 워크로드가 변화에 자동으로 대응하여 목표를 달성하는 데 가장 적합한 리소스 세트를 활용할 수 있습니다.
- 지표를 기준으로 컴퓨팅 요구 사항 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 있습니다. 이 경우 다른 인스턴스 패밀리나 크기로 전환하면 성능과 효율성이 모두 개선될 수 있습니다.

PERF 3 스토리지 솔루션을 어떻게 선택합니까?

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

모범 사례:

- 스토리지 특성 및 요구 사항 파악: 객체 스토리지, 블록 스토리지, 파일 스토리지 또는 인스턴스 스토리지 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 특성 및 구성 옵션과 스토리지와의 관련성을 평가합니다. 프로비저닝된 IOPS, SSD, 마그네틱 스토리지, 객체 스토리지, 아카이브 스토리지 또는 휘발성 스토리지를 사용하는 위치와 방법을 파악하여 워크로드의 성능과 스토리지 공간을 최적화합니다.
- 액세스 패턴과 지표를 기준으로 결정: 워크로드의 액세스 패턴을 기준으로 스토리지 시스템을 선택하고, 워크로드에서 데이터에 액세스하는 방법을 결정하여 이러한 스토리지 시스템을 구성합니다. 블록 스토리지 대신 객체 스토리지를 선택하여 스토리지 효율성을 높이십시오. 선택한 스토리지 옵션을 데이터 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 키-값, 문서, 와이드 열, 그래프, 시계열 또는 인메모리 스토리지)을 선택합니다.
- 사용 가능한 옵션 평가: 워크로드 스토리지 메커니즘 선택 프로세스의 일환으로 사용 가능한 서비스 및 스토리지 옵션을 평가합니다. 데이터 저장용으로 지정된 서비스나 시스템을 사용하는 방법과 시기를 파악합니다. 또한 프로비저닝된 IOPS, 메모리/컴퓨팅 리소스 및 캐싱 등 데이터베이스 성능이나 효율성을 최적화하는 데 사용할 수 있는 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 트랜잭션에 필요한 워크로드에는 관계형 데이터베이스를 활용하거나, 처리량은 더 높지만 최종 처리량은 일정한 키-값 저장소(해당하는 경우)를 활용합니다.
- 액세스 패턴 및 지표를 기준으로 데이터 스토리지 최적화: 성능을 최대한 높이려면 데이터 저장 또는 쿼리 방식을 최적화하는 성능 특성과 액세스 패턴을 사용합니다. 인덱싱, 키 분산, 데이터 웨어하우스 설계, 캐싱 전략 등의 최적화가 시스템 성능이나 전반적인 효율성에 미치는 영향을 측정합니다.

PERF 5 네트워킹 솔루션을 구성하려면 어떻게 해야 합니까?

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

모범 사례:

- 네트워킹이 성능에 미치는 영향 파악: 네트워크 관련 결정 사항이 워크로드 성능에 영향을 주는 방식을 분석하고 파악합니다. 예를 들어 네트워크 지연 시간은 사용자 환경에 영향을 주는 경우가 많으며 잘못된 프로토콜을 사용하는 경우 과도한 오버헤드로 인해 네트워크 용량이 고갈될 수 있습니다.
- 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.
- 하이브리드 워크로드에 적절한 규모의 전용 연결 또는 VPN 선택: 온프레미스 통신에 대한 요구 사항이 있는 경우 대역폭이 워크로드 성능을 제공하기에 충분한지 확인합니다. 대역폭 요구 사항에 따라 단일의 전용 연결 또는 단일 VPN으로는 충분하지 않을 수 있으며, 여러 연결 간에 트래픽 로드 밸런싱을 활성화해야 합니다.
- 로드 밸런싱 및 암호화 오프로딩 활용: 클라우드의 탄력성을 워크로드에 활용할 수 있도록 여러 리소스 또는 서비스에 트래픽을 분산합니다. 로드 밸런싱을 사용하여 암호화 종료를 오프로드하면 성능을 개선하고 트래픽을 효율적으로 관리/라우팅할 수 있습니다.
- 성능을 개선할 수 있는 네트워크 프로토콜 선택: 워크로드 성능에 미치는 영향을 기준으로 시스템과 네트워크 간의 통신에 사용할 프로토콜을 결정합니다.
- 네트워크 요구 사항에 따라 워크로드의 위치 선택: 제공되는 클라우드 위치 옵션을 사용하여 네트워크 지연 시간을 줄이고 처리량을 개선합니다. AWS 리전, 가용 영역, 배치 그룹, 엣지 로케이션(예: Outposts, Local Region 및 Wavelength)을 활용하여 네트워크 지연 시간을 줄이거나 처리량을 늘립니다.
- 지표를 기준으로 네트워크 구성 최적화: 수집 및 분석된 데이터가 제공하는 정보를 사용하여 네트워크 구성 최적화를 결정합니다. 이러한 변경의 영향을 측정한 다음 영향 측정값을 활용해 향후 결정을 내립니다.

검토

PERF 6 새 릴리스를 활용하여 워크로드를 어떻게 발전시킵니까?

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

모범 사례:

- 새 리소스 및 서비스에 대한 최신 정보 숙지: 새로운 서비스, 설계 패턴 및 제품 오퍼링이 제공되면 이를 사용하여 성능을 개선할 방법을 평가하십시오. 임시 평가, 내부 논의 또는 외부 분석을 통해 워크로드의 효율성이나 성능을 개선할 수 있는 방법을 결정하십시오.
- 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.
- 장기적인 워크로드 성능 개선: 조직에서 평가 프로세스를 통해 수집된 정보를 사용하여 제공되는 새 서비스나 리소스를 적극적으로 도입합니다.

모니터링

PERF 7 리소스 성능을 모니터링하려면 어떻게 해야 합니까?

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

모범 사례:

- 성능 관련 지표 기록: 모니터링 및 관찰 서비스를 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 예상 성능을 제공하지 못하는 워크로드의 부분을 파악할 수 있습니다.
- 워크로드 성능을 측정하는 KPI(핵심 성능 지표) 설정: 워크로드가 예상 성능을 제공하는지 여부를 나타내는 KPI를 식별합니다. 예를 들어 API 기반 워크로드에는 전체 응답 지연 시간을 전반적인 성능의 지표로 사용할 수 있으며, 전자상거래 사이트에는 구매 건 수를 KPI로 사용하도록 선택할 수 있습니다.
- 모니터링을 사용하여 경보 기반 알림 생성: 정의한 성능 관련 KPI를 사용하여 측정값이 예상 경계를 벗어나는 경우 경보를 자동으로 생성하는 모니터링 시스템을 사용합니다.
- 정기적인 간격으로 지표 검토: 주기적인 유지 관리의 일환으로 또는 이벤트나 인시던트 대응 과정에서 수집된 지표를 검토합니다. 이러한 검토를 수행하면 문제를 해결하는 데 반드시 필요했던 지표와 문제를 확인/해결/방지하는 데 도움이 되었던 지표(추적한 경우)를 추가로 파악할 수 있습니다.
- 사전 모니터링 및 경보 생성: KPI(핵심 성능 지표)를 모니터링 및 경보 시스템과 함께 사용하여 성능 관련 문제를 선제적으로 해결합니다. 경보를 사용하여 가능한 경우 문제를 해결하는 자동화 작업을 트리거합니다. 자동 대응이 불가능한 경우 대응을 수행할 수 있는 담당자에게 경보를 에스컬레이션합니다. 예를 들어 필요한 KPI(핵심 성능 지표) 값을 예측하고 해당 값이 특정 임계값을 초과하는 경우 경보를 생성할 수 있는 시스템이나, KPI가 필요한 값의 범위를 벗어나는 경우 배포를 자동으로 중지하거나 롤백할 수 있는 도구로 경보를 에스컬레이션할 수 있습니다.

절충

PERF 8 절충을 통해 성능을 개선하려면 어떻게 해야 합니까?

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

모범 사례:

- 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.
- 설계 패턴 및 서비스 파악: 워크로드 성능 개선에 도움이 되는 다양한 설계 패턴과 서비스를 조사하고 파악합니다. 분석을 수행하는 동안 성능 개선을 위해 절충할 수 있는 요소를 파악합니다. 예를 들어 캐시 서비스를 사용하면 데이터베이스 시스템의 로드를 줄일 수 있습니다. 하지만 안전한 캐싱을 구현하기 위한 엔지니어링을 수행해야 하거나, 특정 영역에서 최종 일관성 개념을 도입해야 할 수 있습니다.
- 절충이 고객 및 효율성에 주는 영향 파악:: 성능 관련 개선 사항을 평가할 때는 고객 및 워크로드 효율성에 영향을 미치는 옵션을 결정합니다. 예를 들어 키–값 데이터 스토어를 사용하여 시스템 성능이 개선되는 경우, 이 옵션의 지속되는 특성이 결과적으로 고객에 미치는 영향을 평가하는 것이 중요합니다.
- 성능 개선의 영향 측정:: 성능 개선을 위해 변경이 수행된 경우 수집된 지표와 데이터를 평가합니다. 이 정보를 사용하여 성능 개선이 워크로드, 워크로드의 구성 요소 및 고객에게 미치는 영향을 확인합니다. 이 측정을 수행하면 절충을 통한 성능 개선을 파악할 수 있으며 부정적인 부작용 발생 여부를 확인할 수 있습니다.
- 다양한 성능 관련 전략 사용: 해당하는 경우 다수의 전략을 활용하여 성능을 개선합니다. 예를 들어 데이터 캐싱 등의 전략을 사용해 과도한 네트워크 또는 데이터베이스 호출을 방지하고, 데이터베이스 엔진용 읽기 전용 복제본을 사용해 읽기 속도를 높이고, 가능한 경우 데이터 샤딩/압축을 수행하여 데이터 볼륨을 줄이고, 제공되는 결과를 버퍼링/스트리밍하여 차단을 방지하는 등의 전략을 사용할 수 있습니다.

비용 최적화

클라우드 재무 관리 시행

COST 1 클라우드 재무 관리를 어떻게 구현합니까?

클라우드 재무 관리 시행을 통해 조직은 비용과 사용량을 최적화하고 AWS에서 규모를 확대하면서 비즈니스 가치를 실현하고 재정적 성공을 거둘 수 있습니다.

모범 사례:

- 비용 최적화 역할 설정: 조직 전체에서 비용 인식 설정 및 유지를 담당하는 팀을 만드십시오. 조직 전체에서 재무, 기술 및 비즈니스 역할을 수행하는 인력이 팀에 필요합니다.
- 재무와 기술 간의 파트너십 수립: 클라우드 여정의 모든 단계에서 비용 및 사용량 논의에 재무 및 기술 팀이 참여하도록 하십시오. 팀은 정기적으로 만나 조직의 목적과 목표, 비용 및 사용량의 현재 상태, 재무 및 회계 실무와 같은 주제에 대해 논의합니다.
- 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 조합을 사용하여 동적이어야 합니다.
- 조직의 프로세스에서 비용 인식 구현: 사용량에 영향을 미치는 신규 또는 기존 프로세스에 비용 인식을 구현하고 비용 인식에 기존 프로세스를 활용합니다. 직원 교육에 비용 인식을 구현합니다.
- 비용 최적화 보고 및 알림: 목표를 기준으로 하여 비용 및 사용량 알림을 제공하도록 AWS Budgets를 구성합니다. 정기 회의를 통해 이 워크로드의 비용 효율성을 분석하고 비용 인식 문화를 장려합니다.
- 비용 사전 모니터링: 도구 및 대시보드를 구현하여 워크로드에 대한 비용을 사전에 모니터링합니다. 알림을 받을 때 비용과 범주만 살펴보지 마십시오. 이는 긍정적인 추세를 파악하여 조직 전체에서 이를 장려하는 데 도움이 됩니다.
- 신규 서비스 릴리스 파악: 정기적으로 전문가 또는 APN 파트너의 상담을 받아 더 저렴한 가격을 제공하는 서비스와 기능을 고려합니다. AWS 블로그와 기타 정보 출처를 검토합니다.

지출 및 사용량 인식

COST 2 사용량을 어떻게 관리합니까?

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. 정책은 리소스 수명 주기 동안의 생성, 수정, 폐기를 포함한 리소스와 워크로드의 비용 측면을 다루어야 합니다.
- 목표 및 타겟 이행: 워크로드에 대한 비용 및 사용량 목표를 모두 이행합니다. 목표는 조직에 비용 및 사용량에 대한 방향을 제공하고, 타겟은 워크로드에 대한 측정 가능한 결과를 제공합니다.
- 계정 구조 구현: 조직에 적합한 계정 구조를 구현합니다. 이를 통해 조직 전체에서 비용을 쉽게 할당하고 관리할 수 있습니다.
- 그룹 및 역할 만들기: 정책에 따라 그룹과 역할을 만들고 각 그룹에서 인스턴스와 리소스를 생성, 수정 또는 폐기할 수 있는 사용자를 제어합니다. 예를 들어 개발, 테스트 및 프로덕션 그룹을 만듭니다. 이는 AWS 서비스와 타사 솔루션에 적용됩니다.
- 비용 제어 기능 만들기: 조직 정책과 정의된 그룹 및 역할을 기준으로 제어 기능을 만듭니다. 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 따른 비용 지출을 막으십시오.

COST 3 사용량과 비용을 어떻게 모니터링 합니까?

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

모범 사례:

- 세부 정보 소스 구성: AWS 비용 및 사용 보고서와 Cost Explorer에서 시간 단위의 세분 수준을 구성하여 자세한 비용 및 사용량 정보를 제공합니다. 전송된 모든 비즈니스 성과에 대한 로그 항목을 생성하도록 워크로드를 구성합니다.
- 비용 속성 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 페이지가 있습니다.
- 결제 및 비용 관리 도구 구성: 조직 정책에 맞게 AWS Cost Explorer와 AWS 예산을 구성합니다.
- 비용 및 사용량에 조직 정보 추가: 조직, 워크로드 속성 및 비용 할당 범주를 기준으로 하는 태그 지정 스키마를 정의합니다. 모든 리소스에 태그를 지정합니다. Cost Categories를 사용하여 조직 속성에 따라 비용과 사용량을 그룹화합니다.
- 워크로드 지표를 기준으로 비용 할당: 지표 또는 비즈니스 성과에 따라 워크로드의 비용을 할당하여 워크로드 비용 효율성을 측정합니다. 인사이트와 차지백 기능을 제공할 수 있는 Amazon Athena를 사용하여 AWS 비용 및 사용 보고서를 분석하는 프로세스를 만듭니다.

COST 4 리소스를 어떻게 폐기합니까?

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

모범 사례:

- 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.
- 리소스 폐기: 정기적 감사 또는 사용량 변화와 같은 이벤트에 의해 트리거된 리소스를 폐기합니다. 폐기는 일반적으로 주기적으로 수행되며 수동 또는 자동입니다.
- 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 5 서비스를 선택할 때 비용을 어떻게 평가합니까?

Amazon EC2, Amazon EBS 및 Amazon S3는 기본 구성 AWS 서비스입니다. Amazon RDS 및 Amazon DynamoDB와 같은 관리형 서비스는 더 높은 수준이거나 애플리케이션 수준의 AWS 서비스입니다. 기본 구성 서비스와 관리형 서비스를 적절히 선택하여 이 워크로드의 비용을 최적화할 수 있습니다. 예를 들어, 관리형 서비스를 사용하여 관리 및 운영 고정 비용을 많이 줄이거나 없앨 수 있으며, 응용 프로그램 및 비즈니스 관련 활동을 수행할 수 있습니다.

모범 사례:

- 조직의 비용 요구 사항 파악:: 팀원과 협의하여 이 워크로드의 비용 최적화와 기타 원칙(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 등의 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.
- 비용 효율적인 라이선스가 포함된 소프트웨어 선택: 오픈 소스 소프트웨어는 워크로드 비용에서 상당한 부분을 차지할 수 있는 소프트웨어 라이선스 비용을 없앱니다. 라이선스가 부여된 소프트웨어가 필요한 경우 CPU와 같은 임의의 속성에 바인딩된 라이선스를 피하고 결과 또는 성과에 바인딩된 라이선스를 찾으십시오. 이러한 라이선스의 비용은 해당 라이선스가 제공하는 혜택에 더 근접하게 조정됩니다.
- 이 워크로드의 구성 요소를 선택하여 조직의 우선순위에 따라 비용을 최적화합니다.: 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). 오픈 소스 소프트웨어 또는 라이선스 요금이 없는 소프트웨어를 사용하여 라이선스 비용을 최소화합니다(예: 컴퓨팅 워크로드용 Amazon Linux 또는 Amazon Aurora로 데이터베이스 마이그레이션).
- 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 수행하여 수명 주기 동안 워크로드의 비용 효율성을 유지할 수 있습니다.

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

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

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.
- 데이터를 기준으로 리소스 유형 및 크기 선택: 워크로드 및 리소스 특성(예: 컴퓨팅, 메모리, 처리량, 쓰기 집약형)에 대한 데이터를 기준으로 리소스 크기나 유형을 선택합니다. 일반적으로는 온프레미스 버전과 같은 워크로드의 이전 버전, 설명서 또는 워크로드와 관련된 기타 정보 출처를 사용해 리소스 사용량을 선택합니다.
- 지표를 기준으로 리소스 유형 및 크기 자동 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 워크로드의 사용자 지정 코드로 이 프로비저닝을 수행할 수 있습니다.

COST 7 비용 절감을 위해 가격 결정 모델을 어떻게 사용합니까?

리소스가 비용을 최소화하는 데 가장 적합한 요금 모델을 사용합니다.

모범 사례:

- 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. AWS Cost Explorer의 권장 사항 기능을 사용하여 워크로드 분석을 수행합니다.
- 비용을 기준으로 리전 구현: 리소스 요금은 리전별로 다를 수 있습니다. 따라서 리전 비용을 고려하면 이 워크로드의 전체 가격을 최저 수준으로 낮출 수 있습니다.
- 비용 효율적인 조건을 갖춘 타사 계약 선택: 비용 효율적인 계약과 조건은 이러한 서비스의 비용이 제공하는 혜택에 따라 늘어나도록 보장합니다. 조직에 추가적인 혜택을 제공할 때 조정되는 계약 및 요금을 선택하십시오.
- 워크로드의 모든 구성 요소용 요금 모델 구현: 영구 실행되는 리소스는 Savings Plans 또는 예약 인스턴스와 같은 예약 용량을 활용해야 합니다. 단기간 용량은 스팟 인스턴스나 스팟 플릿을 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.
- 마스터 계정 수준에서 요금 모델 분석 수행: Cost Explorer Savings Plans 및 예약 인스턴스 권장 사항을 사용하여 마스터 계정 수준에서 약정 할인에 대한 정기 분석을 수행합니다.

COST 8 데이터 전송 요금을 위한 계획은 어떻게 됩니까?

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.

수요 관리 및 리소스 공급

COST 9 수요와 리소스 공급은 어떻게 관리합니까?

비용과 성능을 적절하게 절충한 워크로드에서는 비용을 결제한 모든 리소스가 사용되는지 확인하고, 사용률이 매우 낮은 인스턴스가 없도록 해야 합니다. 사용률 지표가 매우 높거나 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.
- 수요 관리를 위한 버퍼 또는 조절 구현: 버퍼링 및 조절은 워크로드의 수요를 수정하여 평준화합니다. 클라이언트가 재시도를 수행할 때 조절을 구현합니다. 요청을 저장하고 나중에 처리하도록 버퍼링을 구현합니다. 클라이언트가 필요한 시간에 응답을 수신하도록 조절 및 버퍼가 설계되었는지 확인합니다.
- 동적으로 리소스 공급: 리소스가 계획된 방식으로 프로비저닝됩니다. 자동 조정과 같은 수요 기반이거나, 수요를 예측할 수 있고 리소스가 시간을 기준으로 제공되는 시간 기반입니다. 이러한 방법을 사용하면 너무 많거나 적게 프로비저닝되는 리소스의 양을 최소화할 수 있습니다.

시간 경과에 따른 최적화

COST 10 새로운 서비스를 어떻게 평가합니까?

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

모범 사례:

- 워크로드 검토 프로세스 개발: 워크로드 검토 기준과 프로세스를 정의하는 프로세스를 개발합니다. 검토 작업은 제공될 수 있는 이점을 반영해야 합니다. 예를 들어 핵심 워크로드와 비용이 청구 금액의 10%보다 많은 워크로드는 분기별로 검토하고, 비용이 청구 금액의 10%보다 적은 워크로드는 연 1회 검토할 수 있습니다.
- 정기적으로 워크로드 검토 및 분석: 정의된 프로세스에 따라 기존 워크로드를 정기적으로 검토합니다.