AWS Well-Architected
프레임워크

2016년 11월

고지 사항

이 문서는 정보 제공 목적으로만 제공됩니다. 본 문서의 발행일 당시 AWS의 현재 제품 및 실행방법을 설명하며, 예고 없이 변경될 수 있습니다. 고객은 본 문서에 포함된 정보나 AWS 제품 또는 서비스의 사용을 독립적으로 평가할 책임이 있으며, 각 정보 및 제품은 명시적이든 묵시적이든 어떠한 종류의 보증 없이 "있는 그대로" 제공됩니다. 본 문서는 AWS, 그 계열사, 공급업체 또는 라이선스 제공자로부터 어떠한 보증, 표현, 계약 약속, 조건 또는 보증을 구성하지 않습니다. 고객에 대한 AWS의 책임 및 채무는 AWS 계약에 준거합니다. 본 문서는 AWS와 고객 간의 어떠한 계약도 구성하지 않으며 이를 변경하지도 않습니다.

목차

요약

이 문서에서는 고객이 스스로 클라우드 기반 아키텍처를 검토 및 개선하고 설계에 대한 의사결정이 사업에 미치는 영향을 더 확실히 파악할 수 있도록 **AWS Well-Architected 프레임워크**에 대해 설명합니다. Well-Architected 프레임워크의 *기반*으로 통하는 다섯 가지 개념 분야에 걸쳐 일반적인 설계 원칙은 물론 구체적인 모범 사례와 지침을 살펴보겠습니다.

서론

AWS(Amazon Web Services)는 고객이 클라우드에서 안정적이고, 안전하고, 효율적이며, 경제적인 시스템을 설계하고 운영할 수 있도록 하려면 아키텍처 모범 사례를 고객과 공유해야 한다는 점을 잘 알고 있습니다. 이러한 노력의 일환으로 개발된 AWS Well-Architected 프레임워크를 이용하면 AWS에서 시스템을 구축하면서 내리게 되는 결정의 장점과 단점을 이해할 수 있습니다. Well-Architected 시스템을 마련하면 사업의 성공 가능성이 대폭 높아진다고 생각합니다.

AWS 솔루션스 아키텍트들은 광범위한 업종 및 사용 사례에 걸쳐 오랫동안 솔루션을 설계한 경험이 있으며, 수천 여 고객들의 AWS 아키텍처를 설계하고 평가해 왔습니다. 그리고 이러한 경험을 바탕으로 클라우드에서 시스템 설계의 모범 사례와 핵심 전략을 밝혀냈습니다.

AWS Well-Architected 프레임워크 설명서에는 특정 아키텍처가 클라우드 모범 사례와 잘 맞는지 여부를 파악하기 위한 근본적인 질문 여러 가지가 수록되어 있습니다. 이 프레임워크를 이용하면 클라우드 기반의 현대적 시스템에 대해 고객이 기대하는 품질 기준에 따라 일관된 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 문서는 CTO(최고 기술 책임자), 아키텍트, 개발자, 운영 팀 팀원 등 기술 업무 담당자를 위해 작성되었습니다. 이 문서를 읽으면 클라우드 아키텍처를 설계 및 운영할 때 따라야 할 AWS 모범 사례 및 전략을 이해하게 됩니다. 본 백서에서 구현 세부 정보 또는 아키텍처 패턴을 제공하는 것은 아닙니다. 하지만 이 정보를 얻을 수 있는 적절한 리소스가 언급되어 있습니다.

AWS Well-Architected 프레임워크의 정의

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

AWS는 이렇게 얻은 교훈을 토대로 아키텍처가 AWS 모범 사례에 얼마나 잘 맞는지 평가할 수 있는 여러 가지 질문을 모은 AWS Well-Architected 프레임워크를 만들어 냈습니다.

AWS Well-Architected 프레임워크는 보안, 안정성, 성능 효율성, 비용 최적화 및 운영 우수성이라는 다섯 가지 기반 위에 서 있으며 각각의 정의는 다음과 같습니다.

기반 이름	설명
보안	위험 평가 및 완화 전략을 통해 정보, 시스템 및 자산을 보호하는 동시에 비즈니스 가치를 제공하는 능력입니다.
안정성	인프라 또는 서비스 장애를 복구하고, 수요에 따라 컴퓨팅 리소스를 탄력적으로 확보하고, 구성 오류나 일시적 네트워크 문제 같은 중단 사태를 완화할 수 있는 시스템의 능력입니다.
성능 효율성	컴퓨팅 리소스를 시스템 요구 사항에 맞게 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 능력입니다.
비용 최적화	불필요한 비용 또는 최선이 아닌 리소스 사용을 피하거나 제거하는 능력입니다.
운영 우수성	비즈니스 가치를 제공하고 지원 프로세스 및 절차를 지속적으로 개선하기 위해 시스템을 운영 및 모니터링하는 능력입니다.

솔루션을 설계할 때 비즈니스 상황에 기초하여 이들 기반을 절충해야 하는데 이러한 비즈니스 의사결정이 엔지니어링 우선순위에 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.

일반 설계 원칙

Well-Architected 프레임워크는 다음과 같은 여러 가지 일반 설계 원칙을 확립하여 클라우드에서의 우수한 설계를 촉진합니다.

- **필요 용량 추측 중단:** 필요한 인프라 용량을 추측할 필요가 없습니다. 시스템을 배포하기 전에 용량을 결정했다가는 결국 고가의 유휴 리소스를 방치하게 되거나 제한된 용량으로 인한 성능 문제를 처리해야 합니다. 하지만 클라우드 컴퓨팅에서는 이러한 문제가 사라집니다. 필요한 만큼 용량을 많이 또는 적게 사용하다가 자동으로 확장하거나 축소할 수 있기 때문입니다.

- **프로덕션 규모의 테스트 시스템:** 클라우드에서는 온디맨드 방식으로 프로덕션 규모의 테스트 환경을 만들고, 테스트를 완료한 다음 해당 리소스를 폐기할 수 있습니다. 테스트 환경을 실행하는 동안에만 비용을 지불하면 되기 때문에 온프레미스 테스트 비용의 몇 분의 일에 불과한 가격으로 실제 환경을 시뮬레이션할 수 있습니다.
- **자동화를 통해 더 간편해진 아키텍처 실험:** 자동화를 통해 수작업 없이 저렴한 비용으로 시스템을 만들고 복제할 수 있습니다. 자동화 과정의 변경 사항을 추적하고, 그 효과를 감사하고, 필요하면 이전의 파라미터로 되돌릴 수 있습니다.
- **아키텍처의 지속적인 혁신:** 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 따르는 결과적 위험이 줄어듭니다. 따라서 시간이 지날수록 시스템은 진화하고, 기업은 혁신을 표준 사례로 활용할 수 있게 됩니다.
- **데이터 기반 아키텍처:** 클라우드에서는 아키텍처 선택이 워크로드 거동에 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.
- **실전을 통한 개선:** 프로덕션 환경에서 이벤트를 시뮬레이션하기 위한 정기적으로 실전 연습을 실시하여 아키텍처 및 프로세스가 어떻게 작동하는지 테스트할 수 있습니다. 그러면 어느 분야에서 개선이 필요한지 파악하고, 조직이 이벤트에 대처하는 경험을 쌓도록 도울 수 있습니다.

Well-Architected 프레임워크의 다섯 가지 기반

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

이 단원에서는 다섯 가지 기반에 대해 설명하고 각각의 정의, 모범 사례, 질문, 고려 사항 및 관련된 주요 AWS 서비스를 소개합니다.

보안 기반

보안 기반에는 위험 평가 및 완화 전략을 통해 정보, 시스템 및 자산을 보호하는 동시에 비즈니스 가치를 제공하는 능력이 포함됩니다.

설계 원칙

클라우드에는 시스템 보안을 강화하는 데 도움이 되는 여러 가지 원칙이 있습니다.

- **모든 계층에 보안 적용:** 그저 인프라의 엣지 부분에서만 보안 어플라이언스(예: 방화벽)를 운영하는 대신, 모든 리소스(예: 모든 가상 서버, 로드 밸런서, 네트워크 서브넷)에서 방화벽을 비롯한 각종 보안 제어를 사용하십시오.
- **추적 가능성 활성화:** 사용 환경에 대한 모든 변경 사항 및 작업을 기록하고 감사합니다.
- **최소 권한 원칙 구현:** 권한 부여가 AWS 리소스와 상호 작용 각각에 적합해야 하며 리소스에 대해 직접적으로 강력한 논리적 액세스 제어를 구현합니다.
- **시스템 보안에 집중:** AWS 공동 책임 모델에 따라 사용자는 애플리케이션, 데이터 및 운영 체제 보안에 집중하고, 보안 인프라 및 서비스는 AWS 에서 제공합니다.
- **보안 모범 사례의 자동 적용:** 소프트웨어 기반의 보안 메커니즘으로 더욱더 빠르고 경제적인 확장성이 안전하게 제공됩니다. 가상 서버의 패치 적용 및 강화된 이미지를 만들어 저장한 다음, 새 서버를 실행할 때마다 그 이미지를 자동으로 사용하면 됩니다. 버전 관리를 통해 템플릿에서 정의 및 관리되는 전체 신뢰 영역 아키텍처를 생성합니다. 루틴 및 이상 보안 이벤트 모두에 대한 응답을 자동화합니다.

정의

클라우드에서의 보안에는 다섯 가지 모범 사례 영역이 있습니다.

1. ID 및 액세스 관리
2. 탐지 제어
3. 인프라 보호
4. 데이터 보호
5. 인시던트 대응

시스템을 설계하려면 먼저 보안과 관련된 관행부터 마련해야 합니다. 누가 어떤 작업을 수행할 수 있는지 제어해야 합니다. 또한 보안 인시던트를 식별하고, 시스템 및 서비스를 보호하고, 데이터 보호를 통해 데이터의 신뢰성 및 무결성을 유지할 수 있어야 합니다 보안 인시던트에 대응하기 위한 잘 정의된 프로세스를 마련하고 숙련해야 합니다. 이것은 금전적 손해 방지 또는 규제 의무 준수 등 목표 달성을 뒷받침하는 중요한 도구와 기법입니다.

AWS 공동 책임 모델에 따라 클라우드를 도입하면 보안 및 규정 준수 목표를 달성할 수 있습니다. 클라우드 서비스를 뒷받침하는 인프라를 AWS가 물리적으로 보호해 주기 때문에 AWS 고객들은 서비스를 이용하여 목표를 달성하는 데 집중할 수 있습니다. 또한 AWS 클라우드에서는 보안 데이터에 더 폭넓게 액세스할 수 있으며 보안 이벤트에 대한 응답도 자동화되어 있습니다.

모범 사례

ID 및 액세스 관리

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

AWS에서는 기본적으로 AWS 서비스 및 리소스에 대한 사용자 액세스를 고객이 직접 제어할 수 있도록 하는 AWS IAM(Identity and Access Management) 서비스로 권한 관리를 지원합니다. 사용자, 그룹, 역할 또는 리소스에 대한 권한을 세부 정책으로 지정할 수 있습니다. 또한 복잡성 수준, 재사용 금지, Multi-Factor Authentication(MFA) 사용 등 강력한 암호 관행을 요구할 수도 있습니다. 기존 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 보안과 관련된 권한 관리 고려 사항에 대한 것입니다. 보안에 대한 질문과 답변 및 모범 사례는 부록을 참조하십시오.

보안 1. AWS 루트 계정 자격 증명에 대한 액세스 및 사용을 어떻게 보호하고 있습니까?

보안 2. AWS Management Console 및 API에 대한 액세스를 제어하기 위해 시스템 사용자의 역할 및 책임을 어떻게 정의하고 있습니까?

보안 3. AWS 리소스에 대한 자동 액세스를 어떻게 제한하고 있습니까? (예: 애플리케이션, 스크립트 및/또는 타사 도구 또는 서비스)

루트 계정의 자격 증명은 항상 보호해야 하며, 이를 위해서는 루트 계정에 MFA를 연결하고 물리적 보안 장치가 있는 장소에 MFA와 자격 증명을 안전하게 보관하는 것이 좋습니다. IAM 서비스를 이용하여 루트 이외의 다른 사용자 권한을 만들고 관리하는 것은 물론 리소스에 대한 액세스 수준을 설정할 수 있습니다.

탐지 제어

탐지 제어를 사용하여 잠재적 보안 인시던트를 식별할 수 있습니다. 이러한 제어는 일반적인 거버넌스 프레임워크의 핵심 부분으로, 품질 프로세스, 법률 준수 의무 및/또는 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, 이것을 운영의 기준으로 삼을 수 있습니다. 또는 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 경보가 가능)을 처리하여 탐지 제어를 구현할 수 있습니다. AWS CloudTrail 로그, AWS API 호출 및 Amazon CloudWatch는 측정치 모니터링을 경보와 함께 제공하며 AWS Config는 구성 내역을 제공합니다. 또한 서비스 수준 로그도 사용 가능한데, 예를 들어 Amazon Simple Storage Service(S3)를 사용하여 액세스 요청을 기록할 수 있습니다. 마지막으로 Amazon Glacier는 감사 가능한 장기 보관을 지원하기 위해 설계된 준수 제어 기능과 함께 미션 크리티컬 데이터를 보존하는 볼트 잠금 기능을 제공합니다.

다음 질문은 보안과 관련된 탐지 제어 고려 사항에 초점을 맞추고 있습니다.

보안 4. 로그를 어떻게 캡처하고 분석하고 있습니까?

Well-Architected 설계에서 로그 관리가 중요한 이유는 보안/과학수사에서 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/삭제 위치를 정의하는 기능을 고객에게 부여함으로써 로그 관리를 보다 쉽게 구현할 수 있도록 합니다. 이렇게 하면 더 단순하고 경제적인 방식으로, 예측 가능하고 안정적으로 데이터를 처리할 수 있습니다.

인프라 보호

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

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

다음 질문은 보안을 위한 인프라 보호 고려 사항에 관한 것입니다.

보안 5. 네트워크 및 호스트 수준 경계 보호를 어떻게 적용하고 있습니까?

보안 6. AWS 서비스 수준 보안 기능을 어떻게 활용하고 있습니까?

보안 7. Amazon EC2 인스턴스에서 운영 체제의 무결성을 어떻게 보호하고 있습니까?

어떤 환경이든 여러 단계의 방어 계층을 두는 것이 좋으며, 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.

설계 원칙 단원에서 설명했듯이 AWS 고객들은 EC2 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 받게 됩니다.

데이터 보호

시스템을 설계하려면 먼저 보안과 관련된 기본적인 관행부터 마련해야 합니다. 예를 들어 *데이터 분류*는 민감도에 따라 조직의 데이터를 구분하는 한 가지 방법이고, *암호화*는 무단 액세스 사용자가 데이터를 해석하지 못하게 만들어 데이터를 보호하는 방법입니다. 이것은 금전적 손해 방지 또는 규제 의무 준수 등 목표 달성을 뒷받침하는 중요한 도구와 기법입니다.

AWS에서는 다음과 같은 관행으로 데이터 보호를 가능하게 합니다.

- AWS 고객들은 데이터에 대한 완전한 통제력을 유지합니다.
- AWS는 정기적인 키 교체 등 키 관리 및 데이터 암호화를 더 간편하게 처리하도록 만들어 드립니다. AWS 기본 서비스를 이용하거나 고객이 직접 관리하여 손쉽게 자동화할 수 있습니다.

- 파일 액세스, 변경 사항 등 중요한 내용이 수록된 상세 로그를 확인할 수 있습니다.
- AWS는 탁월한 회복력을 목표로 스토리지 시스템을 설계했습니다. 예를 들어, Amazon S3(Simple Storage Service)는 99.999999999%의 내구성을 제공하기 위해 설계되었습니다. (다시 말해, Amazon S3에 10,000개의 객체를 저장하는 경우 1,000만 년에 한 번씩 객체 한 개를 잃을 수 있다고 예상할 수 있습니다.)
- 광범위한 데이터 수명 주기 관리 프로세스에 버전 관리를 포함시켜 우발적인 덮어쓰기나 삭제 및 그와 유사한 손해를 방지할 수 있습니다.
- AWS는 절대로 리전 간 데이터 이동을 하지 않습니다. 특정 리전에 저장된 콘텐츠는 고객이 명시적으로 기능을 활성화하거나 그 기능을 제공하는 서비스를 이용하지 않는 한 해당 리전을 벗어나지 않습니다.

다음 질문은 데이터 보안과 관련된 고려 사항에 관한 것입니다.

보안 8. 데이터를 어떻게 분류하고 있습니까?

보안 9. 저장된 데이터를 어떤 방식으로 암호화하고 보호합니까?

보안 10. 키를 어떻게 관리하고 있습니까?

보안 11. 전송 중인 데이터를 어떤 방식으로 암호화하고 보호하고 있습니까?

AWS는 저장된 데이터 및 전송 중인 데이터를 암호화할 수 있는 여러 가지 수단을 제공합니다. 데이터를 암호화하기 쉽도록 AWS 제품 및 서비스에 각종 기능을 내장했습니다. 예를 들어, Amazon S3에는 SSE(서버 측 암호화)를 구현하여 데이터를 암호화된 형태로 저장하기 쉽게 만들었습니다. 또한 흔히 SSL 종료(termination)라고 부르는 전체 HTTPS 암호화 및 복호화 프로세스를 Elastic Load Balancing을 통해 처리하도록 설정할 수도 있습니다.

인시던트 대응

매우 성숙한 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. 보인 인시던트보다 앞서 도구 및 액세스를 마련하고 인시던트 대응을 정기적으로 연습한다면 적기에 조사 및 복구가 가능하도록 아키텍처를 업데이트할 수 있습니다.

AWS에서는 다음과 같은 관행이 효과적인 인시던트 대응을 지원합니다.

- 파일 액세스, 변경 사항 등 중요한 내용이 수록된 상세 로그를 확인할 수 있습니다.
- 이벤트가 자동으로 처리될 수 있고 AWS API 사용을 통해 실행서를 자동화하는 스크립트를 트리거할 수 있습니다.
- AWS CloudFormation을 사용하여 도구 및 "안전한 공간"을 사전 프로비저닝할 수 있습니다. 이를 통해 안전하고 격리된 환경에서 과학수사를 수행할 수 있습니다.

다음 질문은 인시던트 대응과 관련된 고려 사항에 관한 것입니다.

보안 12. 적절한 인시던트 대응을 보장하기 위해 어떻게 합니까?

InfoSec 팀에게 신속하게 액세스를 부여할 수 있는 절차를 마련하고 인스턴스 격리와 과학수사를 위한 데이터 및 상태 캡처를 자동화합니다.

AWS의 핵심 서비스

보안에 꼭 필요한 AWS 제품은 AWS 서비스 및 리소스에 대한 사용자 액세스를 고객이 안전하게 제어할 수 있도록 하는 AWS IAM(Identity and Access Management) 서비스입니다. 이와 함께 다음 서비스 및 기능으로 네 가지 보안 영역을 뒷받침합니다.

자격 증명 및 액세스 관리: IAM을 통해 AWS 서비스와 리소스에 대한 액세스를 안전하게 제어할 수 있습니다. Multi-Factor Authentication(MFA)은 사용자 이름과 암호 외에 보호 계층을 한 단계 더 추가해 줍니다.

탐지 제어: AWS CloudTrail은 AWS API 호출을 기록하고, AWS Config는 AWS 리소스 및 구성에 대한 자세한 재고 정보를 제공하며, Amazon CloudWatch는 AWS 리소스에 대한 모니터링 서비스입니다.

인프라 보호: Amazon Virtual Private Cloud(VPC)를 통해 AWS 클라우드의 격리된 프라이빗 영역을 프로비저닝하고, 가상 네트워크에서 AWS 리소스를 실행할 수 있습니다.

데이터 보호: Elastic Load Balancing, Amazon Elastic Block Store(EBS), Amazon Simple Storage Service(S3) 및 Amazon Relational Database Service(RDS) 등의 서비스에는 암호화 기능이 포함되어 있으므로 전송 및 저장 상태의 데이터를 보호해 줍니다. AWS Key Management Service(KMS)를 이용하면 암호화에 사용되는 키를 더 쉽게 만들고 제어할 수 있습니다.

인시던트 대응: IAM을 사용하여 인시던트 대응 팀에 적절한 권한을 부여해야 합니다. Amazon CloudFormation을 사용하여 조사를 수행할 신뢰할 수 있는 환경을 만들 수 있습니다.

리소스

AWS의 보안 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

설명서 및 블로그

- AWS 보안 센터
- AWS 규정 준수
- AWS 보안 블로그

백서

- AWS 보안 개요
- AWS 보안 모범 사례
- AWS 위험 및 규정 준수

비디오

- AWS 클라우드의 보안
- 공동 책임 개요

안정성 기반

안정성 기반에는 인프라 또는 서비스 장애를 복구하고, 수요에 따라 컴퓨팅 리소스를 탄력적으로 확보하고, 구성 오류나 일시적 네트워크 문제 같은 중단 사태를 완화할 수 있는 능력이 포함됩니다.

설계 원칙

클라우드에는 안정성을 강화하는 데 도움이 되는 여러 가지 원칙이 있습니다.

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

정의

클라우드에서의 안정성에는 세 가지 모범 사례 영역이 있습니다.

1. 기반
2. 변경 관리
3. 장애 관리

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

모범 사례

기반

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

그러나 AWS에는 이러한 기반 요구 사항이 대부분 이미 통합되어 있거나 필요에 따라 적용할 수 있습니다. 클라우드는 기본적으로 한계가 없도록 설계되어 있으므로 충분한 네트워킹 및 컴퓨팅 파워에 대한 요구는 AWS가 책임지고 완수하겠습니다. 고객 여러분은 스토리지 디바이스의 크기 등 리소스 크기와 할당 비율을 필요에 따라 자유롭게 변경하실 수 있습니다.

다음 질문은 안정성과 관련된 기반 고려 사항에 관한 것입니다.

안정성 1. 계정에 대한 AWS 서비스 한도를 어떻게 관리하고 있습니까?

안정성 2. AWS에서 네트워크 토폴로지를 어떻게 계획하고 있습니까?

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

변경 관리

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

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

다음 질문은 안정성과 관련된 변경 관련 고려 사항에 초점을 맞추고 있습니다.

안정성 3. 시스템이 수요 변화에 어떻게 대처하고 있습니까?

안정성 4. AWS 리소스를 어떻게 모니터링하고 있습니까?

안정성 5. 변경 관리를 어떻게 수행하고 있습니까?

수요 변화에 따라 리소스를 자동으로 추가하거나 제거하도록 시스템을 설계하면 안정성이 향상될 뿐 아니라 사업 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 마련이며, 이러한 장애를 파악하는 방법과 대응 방법, 재발 방지 방법 등을 알아 둘 필요가 있습니다.

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

다음 질문은 안정성과 관련된 장애 관리 고려 사항에 관한 것입니다.

안정성 6. 데이터를 어떻게 백업하고 있습니까?

안정성 7. 구성 요소 장애 시 시스템에서 어떻게 대처합니까?

안정성 8. 복원성은 어떻게 테스트하고 있습니까?

안정성 9. 재해 복구를 어떻게 계획하고 있습니까?

정기적으로 데이터를 백업하고 백업 파일을 테스트하여 논리적 오류와 물리적 오류를 모두 복구할 수 있는지 확인하십시오. 빈번한 시스템 자동 테스트를 통해 장애를 파악하고 복구하는 것이 장애 관리의 열쇠입니다(정기적으로 실시하는 것이 바람직하며 중요한 시스템 변경 후에도 테스트 실시). 목표 복구 시간(Recovery Time Objective, RTO), 목표 복구 시점(Recovery Point Objective, RPO) 같은 KPI를 적극적으로 추적하여 특히 장애 테스트 시나리오에서 시스템의 복원성을 평가하십시오. KPI를 추적하면 단일 장애 지점을 파악 및 완화하는 데 도움이 됩니다. 목표는 시스템 복구 프로세스를 철저히 테스트함으로써 모든 데이터를 복구할 수 있으며 문제가 지속되더라도 고객에게 계속 서비스를 제공할 수 있다는 확신을 얻는 것입니다. 통상적인 프로덕션 프로세스와 마찬가지로 복구 프로세스도 제대로 실행해야 합니다.

AWS의 핵심 서비스

런타임 지표를 모니터링하는 Amazon CloudWatch는 안정성 보장을 위한 AWS의 핵심 서비스입니다. 세 가지 안정성 영역을 뒷받침하는 그 밖의 서비스와 기능은 다음과 같습니다.

기반: AWS Identity and Access Management (IAM)를 통해 AWS 서비스와 리소스에 대한 액세스를 안전하게 제어할 수 있습니다. Amazon VPC를 통해 AWS 클라우드의 격리된 프라이빗 영역을 프로비저닝하고, 가상 네트워크에서 AWS 리소스를 실행할 수 있습니다.

변경 관리: AWS CloudTrail은 계정에 대한 AWS API 호출을 기록하고 로그 파일을 감사할 수 있도록 사용자에게 전달합니다. AWS Config는 AWS 리소스 및 구성에 대한 자세한 목록 정보를 제공하고, 구성 변경을 지속적으로 기록합니다.

장애 관리: AWS CloudFormation은 AWS 리소스의 템플릿을 만들어 순서에 따라 예측 가능한 방식으로 프로비저닝할 수 있는 템플릿을 제공합니다.

리소스

안정성과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

비디오 및 분석 보고서

- 장애 극복: 오류 삽입 및 서비스 안정성
- 클라우드의 가용성 및 안정성 벤치마크

설명서 및 블로그

- 서비스 한도
- 서비스 한도 보고서 블로그

백서

- AWS를 이용한 백업 아카이브 및 복원 방식
- AWS 대규모 인프라 관리
- AWS 재해 복구
- AWS Amazon VPC 연결 옵션

AWS Support

- AWS Premium Support
- Trusted Advisor

성능 효율성 기반

성능 효율성 기반에서는 요구 사항에 따라 컴퓨팅 리소스를 효율적으로 사용하고, 수요 변화 및 기술 진화에 발맞춰 그러한 효율성을 유지하는 데 초점을 맞춥니다.

설계 원칙

클라우드에는 성능 효율성을 확보하는 데 도움이 되는 여러 가지 원칙이 있습니다.

- **고급 기술의 대중화:** 기술에 대한 지식과 복잡성을 클라우드 업체가 제공하는 서비스로 극복하면서, 구현하기 어려운 기술도 쉽게 사용할 수 있습니다. 새로운 기술을 호스팅하고 실행하는 방법을 IT 팀에서 직접 배우는 대신 서비스 방식으로 간편하게 이용하면 됩니다. 예를 들어 NoSQL 데이터베이스, 미디어 트랜스코딩, 기계 학습 등은 모두 기술 커뮤니티 전반에 고르게 확산되지 않은 전문 지식이 요구되는 기술입니다. 클라우드에서는 이러한 기술을 서비스 방식으로 제공하기 때문에 그대로 사용하면 됩니다. 리소스 프로비저닝 및 관리에 신경 쓰지 말고 제품 개발에 집중하십시오.
- **즉각적인 세계화:** 클릭 몇 번이면 전 세계 여러 리전에 손쉽게 시스템을 배포할 수 있습니다. 이를 통해 최소 비용으로 지연 시간을 줄이면서 고객에게 더 나은 사용 환경을 제공할 수 있습니다.
- ***서버리스* 아키텍처 사용:** 클라우드에서 서버 없는 아키텍처를 이용하면 서버를 실행하고 유지하는 등의 전통적인 컴퓨팅 관리 업무를 할 필요가 없습니다. 예를 들면 스토리지 서비스에 정적 웹 사이트 역할을 맡기면 웹 서버가 필요 없고, 이벤트 서비스를 통해 코드를 호스팅할 수 있습니다. 이렇게 하면 서버 관리에 따르는 운영 부담이 줄어들 뿐 아니라, 이러한 관리 서비스는 클라우드 단위에서 작동하므로 트랜잭션 비용을 낮출 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:** 보관 대상에 가장 적합한 기술 접근 방식을 사용합니다. 예를 들어 데이터베이스 또는 스토리지 접근 방식을 선택할 때 데이터 액세스 패턴을 고려합니다.

정의

클라우드에서의 성능 효율성에는 네 가지 모범 사례 영역이 있습니다.

1. 선택(컴퓨팅, 스토리지, 데이터베이스, 네트워크)
2. 복습
3. 모니터링
4. 트레이드오프

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

모범 사례

Selection

특정 시스템에 대한 최적의 솔루션은 워크로드에 따라 달라지며, 흔히 여러 접근 방식이 복합적으로 사용됩니다. 제대로 구축된 시스템은 여러 솔루션을 사용하며 성능을 향상하기 위해 다양한 기능을 활용합니다.

AWS에서는 리소스가 가상화되고 다양한 유형 및 구성으로 제공됩니다. 따라서 보다 손쉽게 요구에 보다 근사하게 일치하는 접근 방식을 찾을 수 있을 뿐만 아니라, 온프레미스 인프라에서와 달리 다양한 스토리지 옵션을 손쉽게 찾을 수 있습니다. 예를 들어 Amazon DynamoDB와 같은 관리형 서비스는 완벽하게 관리되는 NoSQL 데이터베이스로, 어떤 규모에서도 지연 시간이 한 자릿수 밀리초 단위로 매우 짧습니다.

다음 예제 질문은 선택 고려 사항에 관한 것입니다.

성능 1. 최상의 성능을 제공하는 아키텍처를 어떻게 선택합니까?

아키텍처에 대한 패턴 및 구현을 선택할 때는 최적의 솔루션에 대한 데이터 기반 접근 방식을 사용합니다. AWS 솔루션 아키텍트, AWS 레퍼런스 아키텍처 및 AWS 파트너가 이제까지 AWS에 축적된 지식을 바탕으로 고객이 아키텍처를 선택하는 과정을 도울 수 있습니다. 하지만 아키텍처를 최적화하기 위해서는 벤치마킹 또는 부하 테스트를 통해 획득한 데이터가 필요합니다.

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

컴퓨팅

특정 시스템에 가장 적합한 컴퓨팅 솔루션은 애플리케이션의 설계, 사용 패턴 및 구성 설정에 따라 달라질 수 있습니다. 아키텍처는 다양한 구성 요소마다 서로 다른 컴퓨팅 솔루션을 사용하고 성능을 개선하기 위해 다양한 기능을 이용할 수 있습니다. 아키텍처에 적합하지 않은 컴퓨팅 솔루션을 선택하면 성능 효율이 저하될 수 있습니다.

AWS에서는 컴퓨팅이 인스턴스, 컨테이너, 기능 등 세 가지 형태로 제공됩니다.

- **인스턴스**는 가상화된 서버이며, 따라서 버튼을 클릭하거나 API 호출을 통해 기능을 변경할 수 있습니다. 클라우드에서는 리소스를 한번 결정하면 그대로 고정되는 것이 아니므로 다양한 서버 유형을 시험해 볼 수 있습니다. AWS에서는 이러한 가상 서버, 즉 *인스턴스*를 다양한 제품군과 크기로 제공하며 솔리드 스테이트 드라이브(SSD)와 그래픽 처리 장치(GPU)를 비롯한 매우 다양한 기능을 사용할 수 있습니다.
- **컨테이너**는 애플리케이션 및 종속 요소를 리소스가 격리된 프로세스에서 실행할 수 있는 일종의 운영 체제 가상화입니다.
- **기능**은 실행하려는 코드로부터 실행 환경을 추상화합니다. 예를 들어, AWS Lambda를 이용하면 인스턴스를 실행하지 않고도 코드를 실행할 수 있습니다.

다음 예제 질문은 컴퓨팅 고려 사항에 관한 것입니다.

성능 2. 컴퓨팅 솔루션은 어떻게 선택합니까?

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

스토리지
특정 시스템에 대한 최적의 스토리지 솔루션은 액세스 방식(블록, 파일 또는 객체), 액세스 패턴(무작위 또는 순차적), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적), 가용성 및 내구성 제약에 따라 다릅니다. 제대로 구축된 시스템은 여러 스토리지 솔루션을 사용하며 성능을 향상하기 위해 다양한 기능을 활용합니다.

AWS에서 스토리지는 가상화되며 다양한 유형으로 제공됩니다. 따라서 보다 손쉽게 스토리지 방식을 요구에 보다 밀접하게 맞출 수 있을 뿐만 아니라, 온프레미스 인프라에서와 달리 다양한 스토리지 옵션을 손쉽게 선택할 수 있습니다. 예를 들어, Amazon S3는 99.999999999%의 내구성을 제공하도록 설계되었습니다. 또한 HDD(마그네틱 하드 드라이브)에서 SSD(Solid State Drive)로 변경할 수 있으며, 몇 초 안에 한 인스턴스에서 다른 인스턴스로 가상 드라이브를 손쉽게 이동할 수 있습니다.

다음 예제 질문은 성능 효율을 위한 스토리지 고려 사항에 관한 것입니다.

성능 3. 스토리지 솔루션은 어떻게 선택합니까?

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

데이터베이스
특정 시스템에 대한 최적의 데이터베이스 솔루션은 가용성, 일관성, 파티션 허용치, 지연 시간, 내구성, 확장성 및 쿼리 용량에 대한 요구 사항에 따라 다릅니다. 많은 시스템들은 다양한 하위 시스템에 대해 서로 다른 데이터베이스 솔루션을 사용하며, 성능을 향상시키기 위해 다양한 기능을 사용합니다. 시스템에 적합하지 않은 데이터베이스 솔루션 및 기능을 선택하면 성능 효율이 저하될 수 있습니다.

AWS의 Amazon Relational Database Service(RDS)는 완벽하게 관리되는 관계형 데이터베이스를 제공합니다. Amazon RDS를 이용하면 거의 가동 중지 없이 데이터베이스의 컴퓨팅과 스토리지 리소스를 확장할 수 있습니다. Amazon DynamoDB는 완벽하게 관리되는 NoSQL 데이터베이스로, 확장 시 지연 시간이 한 자릿수 밀리초 단위로 매우 짧습니다. Amazon Redshift는 페타바이트 규모의 관리형 데이터 웨어하우스로, 성능 또는 용량을 변경해야 할 경우 노드 수 또는 유형을 변경할 수 있습니다.

다음 예제 질문은 성능 효율을 위한 데이터베이스 고려 사항에 관한 것입니다.

성능 4. 데이터베이스 솔루션은 어떻게 선택합니까?

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

네트워크

특정 시스템에 대한 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항 등에 따라 다릅니다. 사용자 또는 온프레미스 리소스와 같은 물리적 제약 조건이 위치 옵션을 좌우하며, 이는 에지 기술 또는 리소스 배치를 사용하여 상쇄될 수 있습니다.

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

다음 예제 질문은 성능 효율을 위한 스토리지 고려 사항에 관한 것입니다.

성능 5. 네트워크 솔루션은 어떻게 선택합니까?

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

복습

솔루션을 설계할 때 선택 가능한 옵션 세트는 유한합니다. 하지만 시간이 지나면서 아키텍처의 성능을 개선시킬 수 있는 새로운 기술 및 접근 방식을 사용할 수 있게 됩니다.

AWS를 사용하면 고객의 요구를 충족하기 위해 지속적으로 추진되는 혁신의 혜택을 받을 수 있습니다. AWS는 정기적으로 새로운 리전, 에지 위치, 서비스 및 기능을 출시합니다. 이들 모두는 아키텍처의 성능 효율을 확실히 개선할 수 있습니다.

다음 예제 질문은 성능 효율 검토에 관한 것입니다.

성능 6. 새로운 리소스 유형 및 기능을 도입할 때 가장 적절한 리소스 유형을 선택하기 위해 어떻게 합니까?

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

모니터링

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

AWS를 사용하면 Amazon CloudWatch가 모니터링 기능 및 알림 경보 전송 기능을 제공하며, 사용자는 Amazon Kinesis, Amazon Simple Queue Service(SQS) 및 AWS Lambda를 통해 조치를 트리거하여 성능 문제를 해결하는 자동화를 이용할 수 있습니다.

다음 예제 질문은 성능 효율 모니터링에 관한 것입니다.

성능 7. 리소스를 시작한 후 정상적으로 수행되고 있는지 확인하기 위해 리소스를 어떻게 모니터링합니까?

오탐(false positive) 또는 데이터가 과도하게 발생하지 않도록 하는 것이 효과적인 모니터링 솔루션의 관건입니다. 자동 트리거는 운영자의 실수를 방지하고 문제 해결 시간을 단축시킬 수 있습니다. 프로덕션 환경에서 시뮬레이션을 통해 경보 솔루션이 올바로 문제를 인지하는지 테스트하는 "실전 연습"을 계획합니다.

트레이드오프

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

AWS를 사용하면 몇 분 내에 전 세계의 여러 곳에 리소스를 배포하여 최종 사용자에게 더욱 가깝게 다가갈 수 있습니다. 또한 데이터베이스 시스템과 같은 정보 저장소에 읽기 전용 복제본을 동적으로 추가하여 기본 데이터베이스의 부하를 줄일 수 있습니다. 또한 AWS는 인 메모리 데이터 스토어 또는 캐시를 제공하는 Amazon ElastiCache, 정적 콘텐츠의 사본을 최종 사용자에게 더 가깝게 캐시하는 Amazon CloudFront와 같은 캐싱 솔루션을 제공합니다.

다음 예제 질문은 성능 효율을 위한 시공간 트레이드오프에 관한 것입니다.

성능 8. 성능을 개선하기 위해 트레이드오프를 어떻게 사용합니까?

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

AWS의 핵심 서비스

성능 효율을 달성하기 위한 AWS의 핵심 서비스는 Amazon CloudWatch입니다. 이 서비스는 리소스와 시스템을 모니터링하여 전반적인 성능 및 운영 상태를 확인할 수 있도록 합니다. 다음 서비스는 성능 효율성 영역에서 중요합니다.

선택:

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

스토리지: Amazon EBS는 사용 사례에 맞춰 최적화할 수 있는 광범위한 스토리지 옵션(예: SSD 및 프로비저닝된 IOPS(초당 입/출력 작업 수)을 제공합니다. Amazon S3는 서버리스 콘텐츠 전송을 제공하며 Amazon S3 Transfer Acceleration은 장거리에서 파일을 빠르고, 쉽고, 안전하게 전송할 수 있게 해줍니다.

데이터베이스: Amazon RDS는 사용 환경에 맞춰 최적화할 수 있게 해주는 다양한 데이터베이스 기능(프로비저닝된 IOPS, 읽기 전용 복제본 등)을 제공합니다. Amazon DynamoDB는 확장 시 지연 시간이 한 자릿수 밀리초 단위로 매우 짧습니다.

네트워크: Amazon Route 53은 지연 시간 기반 라우팅을 제공합니다. Amazon VPC 엔드포인트 및 Direct Connect는 네트워크 거리 또는 지터를 줄일 수 있습니다.

검토: AWS 블로그 및 AWS 웹 사이트의 새 소식 섹션에는 새로 출시된 기능과 서비스에 대한 자세한 정보가 제공됩니다.

모니터링: Amazon CloudWatch는 기존의 모니터링 솔루션과 통합할 수 있고 AWS Lambda와 함께 사용하여 조치를 트리거할 수 있는 측정치, 경보 및 알림을 제공합니다.

트레이드오프: Amazon ElastiCache, Amazon CloudFront 및 AWS Snowball은 성능 개선을 위해 사용할 수 있는 서비스입니다. Amazon RDS의 읽기 전용 복제본을 사용하면 읽기 중심의 워크로드를 확장할 수 있습니다.

리소스

성능 효율과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

비디오

- 성능 채널
- AWS의 성능 벤치마킹

설명서

- Amazon S3 성능 최적화
- Amazon EBS 볼륨 성능

비용 최적화 기반

비용 최적화 기반에는 전체 수명 주기에 걸쳐 지속적으로 시스템을 개선 및 개량하는 프로세스가 포함됩니다. 개념 증명의 초기 설계에서 이후의 프로덕션 워크로드 운영에 이르기까지 본 백서의 관행을 채택하면 비용을 최소화하면서 비즈니스 성과를 달성할 수 있는, 따라서 비즈니스가 투자수익률을 극대화할 수 있는 비용 효율적 시스템을 구축 및 운영할 수 있습니다.

설계 원칙

클라우드에서는 다음과 같은 다양한 원칙에 따라 비용을 최적화할 수 있습니다.

- **소비 모델 채택:** 실제로 소비하는 컴퓨팅 리소스에 대해서만 지불하고, 정교한 예측이 아닌 비즈니스 요구 사항에 따라 사용량을 증감합니다. 예를 들어 개발 및 테스트 환경은 대체로 주중에 하루 여덟 시간 동안만 사용됩니다. 이러한 리소스를 사용하지 않는 동안 중지하면 비용 절감 효과는 최대 75%에 달합니다(40시간 vs. 168시간).
- **규모의 경제에 따른 이점:** AWS는 규모의 경제를 실현하기 때문에 클라우드 컴퓨팅을 사용하면 직접 소유하고 관리할 때보다 가변 비용을 낮출 수 있습니다. AWS 클라우드를 사용하는 고객이 수십만 명에 달하므로 보다 저렴한 종량 요금제를 이용할 수 있습니다.
- **데이터 센터 운영에 필요한 비용 제거:** 서버를 랙에 설치하고, 쌓아 올리고, 서버에 전원을 공급하는 등의 과중한 업무를 AWS에서 처리하므로 IT 인프라에 신경 쓸 필요 없이 고객과 사업 프로젝트에만 집중할 수 있습니다.
- **비용 분석 및 부과:** 클라우드는 보다 손쉽게 시스템의 사용 및 비용을 정확하게 파악할 수 있게 해주며, 따라서 개별 비즈니스 소유자에게 IT 비용을 투명하게 부과할 수 있습니다. 이를 통해 투자수익률(ROI)을 측정할 수 있으며, 시스템 소유자가 리소스를 최적화하고 비용을 절감할 수 있는 기회를 제공합니다.
- **관리형 서비스를 사용하여 소유 비용 절감:** 클라우드에서 관리형 서비스는 이메일 전송이나 데이터베이스 관리 같은 작업을 위해 서버를 유지관리하는 운영상의 부담을 없애줍니다. 그리고 관리형 서비스는 클라우드 규모에서 운영되기 때문에 트랜잭션 또는 서비스당 비용이 저렴합니다.

정의

클라우드에서의 비용 최적화에는 네 가지 모범 사례 영역이 있습니다.

1. 비용 효율적인 리소스
2. 공급과 수요의 균형
3. 비용 인지
4. 시간에 따른 최적화

다른 기반과 마찬가지로 여기에도 트레이드오프가 필요합니다. 예를 들어, 출시 기간 단축을 위해 최적화할지 아니면 비용 최소화를 위해 최적화할지 하는 등의 문제입니다. 경우에 따라서는 선불 비용을 최적화하는 데 투자하는 것보다 출시 기간을 단축하여 시장에 빨리 내보내거나, 새로운 기능을 전송하거나, 마감일자에 맞추는 것이 더 중요할 수 있습니다. 경험적 데이터와 달리 설계 결정은 때로 급하게 내려집니다. 가장 비용 효율적인 배포를 벤치마킹하는 데 시간을 투자하기보다는 "만약의 사태"에 과도하게 대비하려는 유혹이 항상 존재하기 때문입니다. 이는 종종 지나친 과다 프로비저닝이나 최적화되지 않은 배포로 이어집니다. 다음 단원에서는 초기는 물론 지속적으로 배포 비용을 최적화하기 위한 기법과 전략적 지침을 제공합니다.

모범 사례

비용 효율적인 리소스

시스템에 적합한 인스턴스와 리소스를 사용하는 것은 비용을 절감하기 위한 핵심 요소입니다. 예를 들어, 작은 서버에서 리포팅 프로세스를 실행하는 데는 5시간이 걸릴 수 있으나 비용이 두 배인 큰 서버에서는 1시간에 실행할 수 있습니다. 두 작업 모두 결과는 동일하지만, 시간이 흐름에 따라 작은 서버가 더 많은 비용을 발생시킵니다.

제대로 구축된 시스템은 가장 비용 효율적인 리소스를 사용하며, 이는 상당히 긍정적인 경제적 효과를 가져올 수 있습니다. 관리형 서비스를 사용하여 비용을 절감할 수도 있습니다. 예를 들어, 이메일 전송 서버를 유지관리하는 대신 메시지당 부과되는 서비스를 사용할 수 있습니다.

AWS는 필요에 가장 적합한 Amazon EC2 인스턴스를 구매할 수 있도록 유연하고 비용 효율적인 다양한 요금 옵션을 제공합니다. *온디맨드 인스턴스*를 사용하면 최소 약정을 체결할 필요 없이 시간 단위로 컴퓨팅 파워를 구입할 수 있습니다. *예약 인스턴스*(RI)를 사용하면 용량을 예약하고 온디맨드 요금을 최대 75%까지 절약할 수 있습니다. *스팟 인스턴스*를 사용하면 대폭 할인된 요금으로 미사용 Amazon EC2 용량에 입찰할 수 있습니다. 스팟 인스턴스는 HPC나 빅 데이터를 사용하는 경우와 같이 개별 서버가 동적으로 이동할 수 있는 서버 집합을 사용해도 시스템에서 이를 허용할 수 있는 경우에 적합합니다.

다음 예제 질문은 비용 최적화를 위해 비용 효율적인 리소스를 선택하는 것에 관한 것입니다.

비용 1. 솔루션을 위해 AWS 서비스를 선택할 때 비용을 고려하고 있습니까?

비용 2. 비용 목표에 부합하는 리소스 규모를 선택했습니까?

비용 3. 비용 목표에 부합하는 적절한 요금 모델을 선택했습니까?

AWS Trusted Advisor와 같은 도구를 사용하여 AWS 사용량을 정기적으로 확인하면 사용률을 능동적으로 모니터링하고 그에 따라 배포를 조정할 수 있습니다.

공급과 수요의 균형

공급과 수요가 최적의 균형을 이루면 시스템 비용을 최대로 절감할 수 있지만, 프로비저닝 시간과 개별 리소스 오류에 대비해 충분한 공급이 있어야 합니다. 수요는 고정적이거나 가변적일 수 있으므로, 막대한 관리 비용을 절감하려면 측정치와 자동화가 필요합니다.

AWS에서는 리소스를 자동으로 프로비저닝하여 수요를 충족할 수 있습니다. Auto Scaling과 수요, 버퍼 및 시간 기반 접근 방식을 사용하면 필요에 따라 리소스를 추가하고 제거할 수 있습니다. 수요 변화를 예측할 수 있으면 더 많은 비용을 절감하고 시스템 요구에 맞는 리소스를 제공할 수 있습니다.

다음 예제 질문은 비용 최적화를 위한 공급과 수요의 균형에 관한 것입니다.

비용 4. 필요한 용량에 크게 초과되지 않도록 용량을 맞추기 위해 어떻게 하고 있습니까?

공급이 수요와 균형을 이루도록 설계할 때는 사용 패턴과 새로운 리소스를 프로비저닝하는 데 소요되는 시간을 적극적으로 고려해야 합니다.

비용 인지

클라우드의 유연성과 민첩성이 증가함에 따라 혁신과 신속한 개발 및 구축이 가속화되고 있습니다. 그 덕분에 하드웨어 사양 확인, 가격 견적 협상, 구매 주문 관리, 배송 예약, 리소스 배포 등을 비롯하여 온프레미스 인프라의 프로비저닝과 관련된 수동 프로세스가 필요 없어졌습니다. 하지만 손쉽게 사용할 수 있고 거의 무제한의 온디맨드 용량이 제공됨에 따라 비용에 대한 새로운 사고 방식이 필요할 수 있습니다.

대부분의 비즈니스는 다양한 팀에서 운영하는 여러 시스템으로 구성되어 있습니다. 개별 비즈니스 또는 제품 소유자별로 리소스 비용을 부과하면 효율적인 사용 습관이 조성되고 낭비를 줄일 수 있습니다. 또한 비용을 사용자별로 정확하게 부과하면 어떤 제품이 비용 효율적인지 파악할 수 있으므로 예산을 할당할 때 현명한 의사결정을 내릴 수 있습니다.

다음 예제 질문은 비용 최적화를 위한 비용 인지에 관한 것입니다.

비용 5. 아키텍처를 설계할 때 데이터 전송 요금을 고려합니까?

비용 6. 사용량과 지출을 어떻게 모니터링하고 있습니까?

비용 7. 더 이상 필요하지 않은 리소스를 폐기하거나 일시적으로 필요하지 않은 리소스를 중지합니까?

비용 8. AWS 사용량을 관리하기 위한 액세스 제어 및 절차를 마련해 두었습니까?

비용 할당 태그를 사용하면 AWS 비용을 분류하고 추적할 수 있습니다. AWS 리소스(예: Amazon EC2 인스턴스 또는 Amazon S3 버킷)에 태그를 적용하면 AWS는 사용 내역 및 비용을 태그별로 집계한 비용 할당 보고서를 만듭니다. 비즈니스 범주를 나타내는 태그(예: 비용 센터, 시스템 이름 또는 소유자)를 적용하여 여러 서비스에 대한 비용을 정리할 수 있습니다.

태그가 지정된 리소스를 엔터티 수명 주기 추적(직원, 프로젝트)과 결합하면 비즈니스에 더 이상 가치를 창출하지 않으므로 폐기해야 하는 불필요한 리소스나 프로젝트를 확인할 수 있습니다. 과다 지출을 미리 확인할 수 있도록 결제 알림을 설정할 수 있으며, AWS 월 사용량 계산기를 사용하여 데이터 전송 비용을 계산할 수 있습니다.

시간에 따른 최적화

AWS에서 여러 새로운 서비스와 기능을 출시함에 따라 기존 설계를 검토하고 그것이 여전히 비용 효율적인지 확인해 봐야 합니다. 또한 요구 사항의 변화에 따라 더 이상 필요하지 않은 리소스 및 전체 서비스 또는 시스템을 폐기하는 방안을 적극적으로 고려해야 합니다.

AWS의 관리형 서비스는 획기적으로 솔루션을 최적화할 수 있으므로, 새롭게 출시되는 관리형 서비스를 알아 두어야 합니다. 예를 들어, Amazon RDS 데이터베이스를 사용하는 것이 Amazon EC2에서 데이터베이스를 직접 운영하는 것보다 경제적일 수 있습니다.

다음 예제 질문은 비용 최적화를 위한 비용 재평가에 관한 것입니다.

비용 9. 새로운 서비스의 활용을 어떻게 관리하고 고려합니까?

배포를 정기적으로 검토하면 새로운 AWS 서비스를 활용하여 비용을 낮출 수 있습니다. 또한 새로운 서비스가 어떻게 비용을 절감해줄 수 있는지도 평가합니다. 예를 들어, Aurora용 AWS RDS를 사용하여 관계형 데이터베이스의 비용을 절감할 수 있습니다.

AWS의 핵심 서비스

비용 최적화를 지원하는 AWS의 핵심 기능은 시스템 비용을 파악할 수 있게 해주는 비용 할당 태그입니다. 다음 서비스 및 기능은 네 가지 비용 최적화 영역에서 중요합니다.

비용 효율적인 리소스: 예약 인스턴스와 선불 용량을 통하여 비용을 절감할 수 있습니다. AWS Trusted Advisor를 사용하면 AWS 환경을 검사하고 비용을 절약할 수 있는 가능성을 모색할 수 있습니다.

공급과 수요의 균형: Auto Scaling을 사용하면 과다 지출 없이 수요에 맞춰 리소스를 추가하거나 제거할 수 있습니다.

비용 인지: Amazon CloudWatch 경보와 Amazon Simple Notification Service(SNS) 알림은 예산 금액을 초과하거나 초과할 것으로 예상될 경우 경고해 줍니다.

시간에 따른 최적화: AWS 블로그 및 AWS 웹 사이트의 *새 소식* 섹션에는 새로 출시된 기능과 서비스에 대한 자세한 정보가 제공됩니다. AWS Trusted Advisor는 AWS 환경을 검사하고 미사용 또는 유휴 리소스를 제거하거나 예약 인스턴스 용량을 사용함으로써 비용을 절감할 수 있는 가능성을 모색합니다.

리소스

AWS의 비용 최적화 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

비디오

- AWS를 통한 비용 최적화

설명서

- AWS 클라우드 경제 센터

도구

- AWS 총 소유 비용(TCO) 계산기
- AWS 세부 결제 보고서
- AWS 월 사용량 계산기
- AWS 비용 탐색기

운영 우수성 기반

운영 우수성 기반에는 프로덕션 워크로드를 관리하는 데 사용되는 운영 관행 및 절차가 포함됩니다. 계획된 변경을 실행하는 방법과 예기치 못한 운영 이벤트에 대한 대응이 여기에 포함됩니다. 변경 실행 및 대응은 자동화되어야 합니다. 운영 우수성의 모든 프로세스 및 절차는 문서화하고, 테스트하고, 정기적으로 검토해야 합니다.

설계 원칙

클라우드에는 운영 우수성을 추진하는 여러 원칙이 있습니다.

- **코드를 사용한 작업 수행:** 자주 반복되는 프로세스 또는 절차가 있는 경우 자동화를 사용합니다. 예를 들어 구성 관리, 변경 및 이벤트 대응을 자동화할 것을 고려합니다.
- **비즈니스 목표에 맞춘 운영 프로세스:** 비즈니스 목표 달성에서 운영 우수성을 나타내는 측정치를 수집합니다. 목표는 운영 모니터링 및 대응이 비즈니스에 중요한 요구 사항을 지원하도록 측정치에서 신호 대비 잡음 비율을 낮추는 것입니다 불필요한 측정치를 수집하면 모니터링 및 대응이 복잡해져 예기치 못한 운영 이벤트에 효과적으로 대응할 수 없게 됩니다.
- **정기적으로 소규모 증분식 변경을 실시:** 워크로드는 구성 요소를 정기적으로 업데이트할 수 있도록 설계해야 합니다. 변경은 대규모 배치가 아니라 소규모 증분식으로 이루어져야 하며 운영에 영향을 미치지 않고 롤백시킬 수 있어야 합니다. 이러한 변경을 유지 관리를 위한 가동 중지 또는 종속된 서비스 구성 요소의 교체 없이 구현할 수 있는 운영 절차를 마련합니다.
- **예기치 못한 이벤트에 대한 대응을 테스트:** 구성 요소 장애 및 기타 예기치 못한 운영 이벤트에 대해 워크로드를 테스트해야 합니다. 운영 이벤트가 발생할 경우 준수할 수 있도록 운영 이벤트에 대응하기 위한 절차를 테스트하고 이해하는 것이 중요합니다. 시뮬레이션된 운영 이벤트 및 오류 삽입에 대한 대응을 테스트할 수 있도록 실전 연습을 계획합니다.
- **운영 이벤트 및 장애로부터 교훈 학습:** 모든 유형의 운영 이벤트 및 장애를 포착하고, 검토하고, 개선에 활용할 수 있는 프로세스를 수립해야 합니다. 복수의 부서가 정기적으로 운영을 검토하여 운영 우수성을 추진하는 프로세스 개선을 도출해야 합니다.
- **운영 절차를 최신 상태로 유지:** 환경과 운영이 진화함에 따라 프로세스 및 절차 가이드도 이에 맞게 조정되어야 합니다. 여기에는 정규 운영 실행서(표준 운영 절차)는 물론 지침서(예기치 못한 운영 이벤트 또는 프로덕션 장애에 대한 대응 계획)의 업데이트가 포함됩니다. 운영을 위한 지침 및 교훈은 동일한 실수를 반복하지 않도록 다른 팀과 공유해야 합니다. 이 정보에 관해 wiki 또는 내부 기술 자료를 사용할 것을 고려하십시오. 평가해야 할 정보에는 운영 측정치, 예기치 못한 이상, 실패한 배포, 시스템 장애 및 비효과적 또는 부적절한 장애 대응이 포함됩니다. 또한 환경과 운영이 진화함에 따라 자동화를 통해 시스템 및 아키텍처 설명서를 캡처하고 업데이트해야 합니다.

정의

클라우드에서의 운영 우수성에는 세 가지 모범 사례 영역이 있습니다.

1. 준비
2. 운영
3. 응답

운영 우수성을 추진하려면 준비가 필수적입니다. 많은 운영 문제는 워크로드를 설계할 때 모범 사례를 따름으로써 방지할 수 있으며, 수정은 프로덕션 환경에서보다 설계 단계에서 구현하는 것이 비용이 덜 발생합니다. 운영 프로세스 및 절차는 철저하게 계획, 테스트 및 검토해야 합니다. 워크로드는 진화해야 하며 자동화되고 관리 가능한 방식으로 변경되어야 합니다. 변경은 소규모의 증분식으로 자주 이루어져야 하며 지속적인 운영에 악영향을 주지 않아야 합니다. 운영 팀은 운영 이벤트 및 장애에 대응할 준비 태세를 갖춰야 하며 이러한 이벤트를 통해 교훈을 학습할 프로세스를 마련해야 합니다.

모범 사례

준비

운영 우수성을 추진하려면 효과적인 준비가 필요합니다. 운영 체크리스트를 활용하면 워크로드가 프로덕션 운영을 위한 준비가 되었는지 확인할 수 있고 효과적 준비 없이 의도치 않게 프로덕션으로 전환하는 것을 방지할 수 있습니다. 워크로드는 운영 팀이 정상적인 일상 작업을 수행하기 위해 참조할 수 있는 운영 지침(실행서)와 예기치 못한 운영 이벤트에 대응하기 위한 지침(지침서)을 구비해야 합니다. 지침서에는 대응 계획은 물론 에스컬레이션 경로와 이해관계자 알림이 포함되어야 합니다. 운영상 변경을 촉발할 수 있는 비즈니스 주기 이벤트도 정기적으로 검토해야 합니다(예: 마케팅 이벤트, 반짝 세일 등). 모든 실행서와 지침서는 차이 또는 문제를 식별하고 잠재적 위험을 완화시킬 수 있도록 테스트해야 합니다. 장애를 추적하고 이로부터 교훈을 얻을 수 있는 메커니즘을 마련해야 합니다. 환경, 아키텍처, 그리고 해당 리소스에 대한 구성 매개 변수는 추적 및 문제 해결을 위해 손쉽게 구성 요소를 식별할 수 있도록 문서화해야 합니다. 구성 변경 역시 추적이 가능하고 자동화되어야 합니다.

AWS에서는 운영 준비성을 지원하기 위해 사용할 수 있는 여러 방법, 서비스 및 기능, 그리고 정상적인 일상 작업과 예기치 못한 운영 이벤트에 대비하기 위한 기능이 제공됩니다. 운영 준비성에는 감독을 보장하기 위해 복수의 부서가 참여하는 수동 검토 또는 피어 검토가 포함될 수 있습니다. 프로덕션으로 배포할 환경이 필요한 리소스를 모두 포함하는지, 환경 구성이 검증된 모범 사례를 기반으로 하여 사용자 실수의 가능성이 줄어드는지 확인하기 위해 AWS CloudFormation과 같은 AWS 서비스를 사용할 수 있습니다. Auto Scaling이나 기타 자동화된 조정 메커니즘을 구현하면 비즈니스 관련 이벤트가 운영 요구 사항에 영향을 미칠 경우 워크로드가 자동으로 대응하도록 할 수 있습니다. AWS Config 규칙을 포함한 AWS Config와 같은 서비스는 AWS 워크로드 및 환경에 대한 변경을 자동으로 추적하고 대응하는 메커니즘을 만듭니다. 또한 태그 지정과 같은 기능을 사용하여 운영 및 대응 시 필요할 경우 워크로드의 모든 리소스를 손쉽게 식별할 수 있도록 하는 것도 중요합니다.

다음 질문은 운영 우수성과 관련된 준비 고려 사항에 관한 것입니다.

운영 1. 클라우드 운영을 위해 어떤 모범 사례를 사용하고 있습니까?

운영 2. 워크로드에 대한 구성 관리는 어떻게 하고 있습니까?

절차 변경에 따라 설명서도 최신 상태를 유지하도록 하십시오. 또한 모든 내용이 포함되어야 합니다. 애플리케이션 설계, 환경 구성, 리소스 구성, 대응 계획 및 완화 계획이 없으면 설명서는 완전하지 않습니다. 설명서를 정기적으로 업데이트 및 테스트하지 않을 경우 예기치 못한 운영 이벤트가 발생했을 때 쓸모가 없어집니다. 워크로드를 프로덕션 전에 검토하지 않을 경우 탐지되지 않은 문제가 발생했을 때 운영에 악영향을 미칩니다. 리소스를 문서화하지 않을 경우 운영 이벤트가 발생했을 때 올바른 리소스가 식별되더라도 대응 방법을 결정하기가 더 어려워집니다.

운영

운영은 표준화해야 하며 일상적으로 관리 가능해야 합니다. 자동화, 잦은 소규모 변경, 정기적 품질 보증 테스트, 변경 사항을 추적, 감사, 롤백 및 검토하기 위해 정의된 메커니즘에 초점을 맞춰야 합니다. 변경은 대규로 드문드문 실시되거나 예정된 가동 중지가 필요하거나 수동 실행이 필요하면 안 됩니다. 워크로드에 대한 주요 운영 인디케이터에 기초한 광범위한 로그 및 측정치를 수집하고 검토하여 지속적인 운영이 보장되어야 합니다.

AWS에서는 지속적인 통합/지속적인 배포(CI/CD) 파이프라인(예: 소스 코드 리포지토리, 빌드 시스템, 배포 및 테스트 자동화)을 설정할 수 있습니다. 릴리스 관리 프로세스(수동 또는 자동)는 테스트해야 하며 소규모 증분식 변경과 추적된 버전을 기반으로 해야 합니다. 운영 문제를 유발한 변경 사항은 운영에 영향을 미치지 않고 되돌릴 수 있어야 합니다. 변경 품질 보증에는 블루/그린, Canary 및 A/B 테스팅과 같은 위험 완화 전략이 포함되어야 합니다. 운영 체크리스트를 사용하여 워크로드의 프로덕션 준비성을 평가해야 합니다. 중앙 집중식 모니터링 및 경보를 위해 로그를 집계합니다. 경보가 알림 및 에스컬레이션을 포함하여 자동 대응을 트리거해야 합니다. 또한 장애뿐 아니라 이상에 대한 모니터도 설계하십시오.

다음 질문은 운영 우수성과 관련된 워크로드 운영 방식에 관한 것입니다.

운영 3. 워크로드를 어떻게 변경의 영향을 최소화하면서 진화시키고 있습니까?

운영 4. 워크로드가 예상대로 운영되는지 확인하기 위해 어떻게 모니터링합니까?

일상적 운영은 물론 예기치 못한 이벤트에 대한 응답도 자동화해야 합니다. 배포, 릴리스 관리, 변경 및 롤백에 대한 수동 프로세스는 피해야 합니다. 릴리스는 드문드문 이루어지는 대규모 배치가 아니어야 합니다. 대규모 변경에서는 롤백이 더 어려우며 롤백 계획 또는 실패 영향을 완화하는 기능이 없을 경우 지속적인 운영이 방해를 받게 됩니다. 대응이 비즈니스 연속성을 유지하는 데 효과가 있도록 모니터링을 비즈니스 요구 사항에 맞춥니다. 대응이 수동으로 이루어지는 중앙 집중화되지 않은 애드혹 모니터링은 예기치 못한 이벤트가 발생했을 때 운영에 미치는 영향이 더 큽니다.

응답

예기치 못한 운영 이벤트에 대한 대응은 자동화해야 합니다. 자동화는 경보만이 아니라 완화, 수정, 롤백 및 복구에도 적용해야 합니다. 경보는 시기 적절해야 하며 대응이 운영 이벤트의 영향을 완화하는 데 적절하지 않을 경우 에스컬레이션션을 트리거해야 합니다. 실패한 배포를 자동으로 롤백하기 위한 품질 보증 메커니즘이 마련해야 합니다. 대응은 이해관계자, 에스컬레이션 프로세스 및 절차를 포함하는 사전 정의된 지침서를 따라야 합니다.

에스컬레이션 경로를 정의해야 하며 기능 및 계층 에스컬레이션 기능을 모두 포함해야 합니다. 계층 에스컬레이션은 자동화해야 하며 에스컬레이션된 우선 순위가 이해관계자 알림을 생성해야 합니다.

AWS에서는 예기치 못한 운영 이벤트에 대한 대응과 자동 대응에서 적절한 경보 및 알림을 보장하는 다수의 메커니즘을 제공합니다. 중앙 집중식으로 워크로드를 모니터링하고 주요 운영 인디케이터와 관련하여 사용 가능한 모든 로그 및 측정치를 기반으로 유효한 경보 및 알림을 생성하는 도구도 있습니다. 여기에는 서비스 또는 구성 요소 장애뿐 아니라 한계를 벗어난 이상에 대한 경보 및 알림도 포함됩니다. 대응은 워크로드의 애플리케이션 상태 이외에 종속적 AWS 환경 및 서비스에 대해서도 활성화해야 합니다. 운영 이벤트가 발생한 후에는 근본 원인 분석(RCA)을 실시하고 아키텍처 및 대응 계획을 개선하는 데 사용해야 합니다.

다음 질문은 운영 우수성과 관련된 이벤트 대응에 관한 것입니다.

운영 5. 예기치 못한 운영 이벤트에는 어떻게 대응합니까?

운영 6. 예기치 못한 운영 이벤트에 대응할 때 에스컬레이션을 어떻게 관리합니까?

대응 계획이 정의되지 않은 애드혹 방식으로 이루어질 경우 결과를 예측할 수 없고 종종 이벤트의 영향이 가중됩니다. 대응이 오래된 지침서를 기반으로 하거나 문제 발생 시 지침서에 액세스할 수 없는 경우에도 예측할 수 없는 결과가 초래될 수 있습니다. 이벤트를 검토할 프로세스가 마련되지 않은 경우 향후의 운영 이벤트를 방지하기가 더 어려워지고 또한 동일한 영향을 받게 될 것입니다.

AWS의 핵심 서비스

운영 우수성을 추진하는 데 사용할 수 있는 두 가지의 주요 서비스가 있습니다. AWS CloudFormation을 사용하여 모범 사례를 기반으로 한 템플릿을 생성하고 리소스를 순서에 따라 예측 가능한 방식으로 프로비저닝할 수 있습니다. Amazon CloudWatch는 측정치를 모니터링하고, 로그를 수집하고, 경보를 생성하고, 대응을 트리거하는 데 사용할 수 있습니다. 세 가지 운영 우수성 영역을 뒷받침하는 그 밖의 서비스와 기능은 다음과 같습니다.

준비: AWS Config는 AWS 리소스 및 구성에 대한 자세한 목록 정보를 제공하고, 구성 변경을 지속적으로 기록합니다. AWS Service Catalog는 모범 사례에 맞춘 표준화된 서비스 제품군을 생성하는 데 유용합니다. Auto Scaling, Amazon SQS와 같은 서비스를 통해 자동화를 사용하는 워크로드를 설계하는 것은 예기치 못한 운영 이벤트가 발생했을 때 지속적인 운영을 보장하는 훌륭한 방법입니다.

운영: AWS CodeCommit, AWS CodeDeploy 및 AWS CodePipeline을 사용하여 AWS 워크로드에 대한 코드 변경을 관리 및 자동화할 수 있습니다. AWS SDK 또는 타사 라이브러리를 사용하여 운영 변경을 자동화할 수 있습니다. AWS CloudTrail을 사용하여 AWS 환경에 적용된 변경 사항을 감사하고 추적할 수 있습니다.

대응: 효과적이고 자동화된 대응을 위해 Amazon CloudWatch 서비스의 모든 기능을 활용합니다. Amazon CloudWatch 경보를 사용하여 경보 및 알림 임계값을 설정할 수 있으며 Amazon CloudWatch 이벤트는 알림 및 자동 대응을 트리거할 수 있습니다.

리소스

운영 우수성과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하십시오.

비디오

- AWS re:Invent 2015 – DevOps at Amazon
- AWS re:Invent 2015 - Scaling Infrastructure Operations
- AWS Summit 2016 - DevOps, Continuous Integration and Deployment on AWS

설명서 및 블로그

- DevOps 및 AWS
- 지속적 통합이란 무엇입니까?
- 지속적 전달이란 무엇입니까?
- AWS DevOps 블로그

백서

- AWS Cloud Adoption Framework - Operations Perspective
- Introduction to DevOps on AWS
- AWS 운영 체크리스트

AWS Support

- AWS Cloud Operations Review
- AWS Premium Support
- AWS Trusted Advisor

결론

AWS의 Well-Architected 프레임워크는 클라우드에 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 각각의 주요 기반에 대한 AWS 모범 사례를 제시합니다. 아키텍처에 이 프레임워크를 사용하면 기능적 요구 사항에 초점을 둔 안정적이고 효율적인 시스템을 구축할 수 있습니다.

기고자

다음은 이 문서의 작성에 도움을 준 개인 및 조직입니다.

- Philip Fitzsimons, Sr. Well-Architected 담당 매니저, Amazon Web Services
- Erin Rifkin, 선임 제품 매니저, Amazon Web Services
- Amazon Web Services의 보안 솔루션 수석 아키텍트 Max Ramsay
- Amazon Web Services의 보안 솔루션 아키텍트 Scott Paddock
- John Steele, Sr. 기술 계정 매니저, Amazon Web Services
- Amazon Web Services의 솔루션 아키텍트 Callum Hughes

문서 이력

2015년 11월 20일. 최신 Amazon CloudWatch 로그 정보로 부록 업데이트함.

2016년 11월 20일. 운영 우수성 기반을 포함하도록 프레임워크를 업데이트하고, 중복을 줄이기 위해 다른 기반을 개정 및 업데이트했으며, 수천의 고객과 검토를 통해 획득한 내용을 통합.

부록: Well-Architected 질문, 답변 및 모범 사례

이 부록에서는 Well-Architected 질문과 답변 및 다음과 같은 주요 기반에 따른 모범 사례를 제공합니다.

보안 기반

ID 및 액세스 관리

SEC 1. How are you protecting access to and use of the AWS root account credentials?

AWS 루트 계정 자격 증명은 여러 운영 체제의 루트 또는 로컬 관리자와 유사하며, 가능한 한 적게 사용해야 합니다. 현재 모범 사례는 AWS Identity and Access Management(IAM) 사용자를 만들고, 이를 관리자 그룹에 연결한 다음, IAM 사용자가 계정을 관리하는 것입니다. AWS 루트 계정에는 API 키가 있으면 안 되며, 강력한 암호가 있어야 합니다. 또한 하드웨어 Multi-Factor Authentication(MFA) 디바이스에 연결되어 있어야 합니다. 이는 루트 자격 증명을 AWS Management Console을 통해서만 사용할 수 있도록 하며, 애플리케이션 프로그래밍 인터페이스(API) 호출에 사용하지 못하도록 합니다. 일부 재판매업체 또는 리전에서는 AWS 루트 계정 자격 증명을 배포하거나 지원하지 않습니다.

모범 사례:

- **MFA를 사용하고 루트를 및 최소한으로 사용** 최소한의 필수 작업에 대해서만 AWS 루트 계정 자격 증명을 사용합니다.
- **루트를 사용하지 않음**

SEC 2. How are you defining roles and responsibilities of system users to control human access to the AWS Management Console and API?

현재 모범 사례는 사용자 그룹을 만들어 시스템 사용자에 대해 정의된 역할 및 책임을 구분하는 것입니다. 사용자 그룹은 IAM(Identity and Access Management) 그룹이나 교차 계정 액세스를 위한 IAM 역할, Web Identities를 사용하거나, SAML(Security Assertion Markup Language) 통합을 통해(예: Active Directory의 역할 정의) 또는 일반적으로 SAML이나 AWS STS(Security Token Service)를 통해 통합하는 타사 솔루션(예: Okta, Ping Identity 또는 기타 맞춤형 기술) 등의 다양한 기술을 사용하여 정의할 수 있습니다. 공유 계정은 사용하지 않는 것이 좋습니다.

모범 사례:

- **직원 수명 주기 관리** 직원 수명 주기 정책을 정의하고 시행합니다.
- **최소 권한** 사용자, 그룹 및 역할에 대해 비즈니스 요구 사항을 충족하는 데 필요한 최소한의 권한만 명확하게 정의하고 부여합니다.

SEC 3. How are you limiting automated access to AWS resources? (e.g., applications, scripts, and/or third-party tools or services)

사용자 그룹을 만드는 것과 유사한 방식으로 액세스를 체계적으로 정의해야 합니다. Amazon EC2 인스턴스에서 이러한 그룹을 EC2의 IAM 역할이라고 합니다. 현재 모범 사례는 EC2 및 AWS SDK 또는 CLI의 IAM 역할을 사용하는 것입니다. 이들은 EC2 자격 증명의 IAM 역할 검색을 기본적으로 지원합니다. 일반적으로 사용자 자격 증명은 EC2 인스턴스로 삽입되지만, 스크립트 및 소스 코드에 자격 증명을 하드 코딩하는 것은 권장되지 않습니다.

모범 사례:

- **자동 액세스에 사용되는 정적 자격 증명** 이러한 자격 증명을 안전하게 저장합니다.
- **자동 액세스를 위한 동적 인증** 인스턴스 프로필 또는 Amazon STS를 사용하여 관리합니다.

탐지 제어

SEC 4. How are you capturing and analyzing logs?

로그를 캡처하는 것은 성능에서 보안 인시던트에 이르기까지 모든 영역을 조사하는 데 중요합니다. 현재 모범 사례는 로그를 소스에서 로그 처리 시스템(예: CloudWatch Logs, Splunk, Papertrail 등)으로 주기적으로 직접 옮기거나, 비즈니스 요구에 따라 이후 처리할 수 있도록 Amazon S3 버킷에 저장하는 것입니다. 일반적인 로그 소스는 AWS API 및 사용자 관련 로그(예: AWS CloudTrail), AWS 서비스별 로그(예: Amazon S3, Amazon CloudFront 등), 운영 체제 생성 로그, 타사 애플리케이션별 로그입니다. Amazon CloudWatch 로그를 사용하여 Amazon EC2 인스턴스, AWS CloudTrail 또는 기타 소스의 로그 파일을 모니터링하고, 저장하며, 액세스할 수 있습니다.

모범 사례:

- **활동을 적절히 모니터링 Amazon CloudWatch 로그, 이벤트, VPC 흐름 로그, ELB 로그, S3 버킷 로그 등**
- **AWS Cloud Trail을 활성화**
- **운영 체제 또는 애플리케이션 로그를 모니터링**

인프라 보호

SEC 5. How are you enforcing network and host-level boundary protection?

온프레미스 데이터 센터에서 DMZ는 방화벽을 사용하여 신뢰할 수 있는 영역 및 신뢰할 수 없는 영역에 있는 개별 시스템에 접근합니다. AWS에서는 상태 저장 및 상태 비저장 방화벽을 사용합니다. 상태 저장 방화벽을 보안 그룹이라고 하며, 상태 비저장 방화벽을 Amazon Virtual Private Cloud(VPC)의 서브넷을 보호하는 네트워크 액세스 제어 목록(ACL)이라고 합니다. 현재 모범 사례는 VPC에서 시스템을 실행하고, 보안 그룹에서 역할 기반 보안(예: 웹 계층, 앱 계층 등) 및 네트워크 ACL에서 위치 기반 보안(예: 가용 영역당 한 서브넷에서 Elastic Load Balancing 계층, 가용 영역당 또 다른 서브넷에서 웹 계층 등)을 정의하는 것입니다.

모범 사례:

- **VPC 내에서 네트워크 트래픽 제어** 예를 들어 방화벽, 보안 그룹, NACLS, 배스천 호스트 등을 사용합니다.
- **경계에서 네트워크 트래픽 제어** 예를 들어 AWS WAF, 호스트 기반 방화벽, 보안 그룹, NACLS 등을 사용합니다.

SEC 6. How are you leveraging AWS service level security features?

AWS 서비스는 추가 보안 기능을 제공할 수 있습니다(예: Amazon S3 버킷 정책, Amazon SQS, Amazon DynamoDB, KMS 키 정책 등).

모범 사례:

- **적절할 경우 추가 기능을 사용**

SEC 7. How are you protecting the integrity of the operating system on your Amazon EC2 instances?

또 다른 기존 제어는 운영 체제의 무결성을 보호하는 것입니다. 기존 호스트 기반 기술(예: OSSEC, Tripwire, Trend Micro Deep Security 등)을 사용하여 Amazon EC2에서 이를 손쉽게 수행할 수 있습니다.

모범 사례:

- **파일 무결성** EC2 인스턴스에 파일 무결성 제어를 사용합니다.
- **EC2 침입 탐지** EC2 인스턴스에 호스트 기반 침입 탐지 제어를 사용합니다.
- **AWS Marketplace 또는 파트너 솔루션** AWS Marketplace 또는 APN 파트너의 솔루션을 사용합니다.
- **구성 관리 도구** 기본적으로 보호되는 사용자 지정 Amazon Machine Image(AMI) 또는 구성 관리 도구(예: Puppet 또는 Chef)를 사용합니다.

데이터 보호

SEC8. How are you classifying your data?

데이터 분류는 민감도에 따라 조직의 데이터를 구분하는 방법을 제공합니다. 여기에는 사용 가능한 데이터 유형, 데이터 위치, 액세스 레벨, 데이터 보호(예: 암호화 또는 액세스 제어를 사용)가 포함됩니다.

모범 사례:

- **데이터 분류 스키마를 사용**
- **모든 데이터를 민감 데이터로 취급**

SEC 9. How are you encrypting and protecting your data at rest?

기존의 보안 제어는 저장된 데이터를 암호화하는 것입니다. AWS는 클라이언트 측(예: SDK 지원, 운영 체제 지원, Windows Bitlocker, dm-crypt, Trend Micro SafeNet 등) 및 서버 측(예: Amazon S3) 암호화를 모두 사용하여 이를 지원합니다. 서버 측 암호화(SSE) 및 Amazon Elastic Block Store 암호화 볼륨 등을 사용할 수도 있습니다.

모범 사례:

- **필요하지 않음** 저장 데이터 암호화가 필요하지 않습니다.
- **저장 시 암호화**

SEC 10. How are you managing keys?

키는 암호이므로 보호해야 하며 적절한 교체 정책을 정의하고 사용해야 합니다. 모범 사례는 관리 스크립트 및 애플리케이션에 이러한 암호를 하드코딩하지 않도록 주의하는 것입니다.

모범 사례:

- **AWS CloudHSM** AWS CloudHSM을 사용합니다.
- **AWS 서비스 제어 사용** AWS 서비스별 제어를 사용하여 저장된 데이터를 암호화할 수 있습니다(예: Amazon S3 SSE, Amazon EBS 암호화 볼륨, Amazon Relational Database Service(RDS) Transparent Data Encryption(TDE) 등).
- **클라이언트 측 사용** 클라이언트 측 기술을 사용하여 저장된 데이터를 암호화합니다.
- **AWS Marketplace 또는 파트너 솔루션** AWS Marketplace 또는 APN 파트너의 솔루션을 사용합니다. (예: SafeNet, TrendMicro 등)

SEC 11. How are you encrypting and protecting your data in transit?

모범 사례는 암호화를 사용하여 전송 중인 데이터를 보호하는 것입니다. AWS는 서비스 API에 대한 암호화된 엔드포인트 사용을 지원합니다. 또한 고객은 자신의 Amazon EC2 인스턴스 내에서 다양한 기술을 사용할 수 있습니다.

모범 사례:

- **필요하지 않음** 전송 데이터 암호화가 필요하지 않습니다.
- **통신 암호화** 통신에 TLS 등을 적절히 사용합니다.

인시던트 대응

SEC 12. How do you ensure that you have the appropriate incident response?

보인 인시던트보다 앞서 도구 및 액세스를 마련하고 인시던트 대응을 정기적으로 연습한다면 적기에 조사 및 복구가 가능하도록 아키텍처를 업데이트할 수 있습니다.

모범 사례:

- **액세스 사전 프로비저닝** Infosec에 적절한 액세스 또는 빠르게 액세스할 수 있는 수단이 부여되어 있습니다. 인시던트에 적절히 대응할 수 있도록 이러한 액세스가 사전에 프로비저닝되어야 합니다.

- **도구 사전 배포**인시던트에 적절히 대응할 수 있도록 Infosec가 적절한 도구를 AWS에 사전 배포합니다.
- **비 프로덕션 실전 연습** 인시던트 대응 시뮬레이션을 비 프로덕션 환경에서 정기적으로 실시하고 이를 통해 얻은 교훈을 아키텍처 및 운영에 통합합니다.
- **프로덕션 실전 연습** 인시던트 대응 시뮬레이션을 프로덕션 환경에서 정기적으로 실시하고 이를 통해 얻은 교훈을 아키텍처 및 운영에 통합합니다.

안정성 기반

기반

REL 1. How are you managing AWS service limits for your accounts?

AWS 계정은 기본 서비스 제한에 따라 프로비저닝되어 신규 사용자가 필요한 것보다 더 많은 리소스를 실수로 프로비저닝하는 일을 방지합니다. AWS 고객은 각자의 AWS 서비스 요구를 평가하고 사용되는 각 리전에 대한 제한을 적절히 변경하도록 요청해야 합니다.

모범 사례:

- **제한도 모니터링 및 관리** 잠재적 AWS 사용량을 평가하고, 리전 제한을 적절히 높이고, 사용량 증가를 계획합니다.
- **자동 모니터링 설정** 임계값에 접근하면 경고하도록 도구(예: SDK)를 구현합니다.
- **고정된 서비스 제한 숙지** 변경할 수 없는 서비스 제한을 숙지하고 이에 따라 설계합니다.
- **서비스 제한과 최대 사용량 사이에 장애 조치를 수용할 정도로 충분한 갭이 있는지 확인**
- **모든 관련 계정 및 리전에서 서비스 제한을 고려**

REL 2. How are you planning your network topology on AWS?

애플리케이션은 기본으로 EC2 Classic, VPC, VPC 등 하나 이상의 환경에 존재할 수 있습니다. 시스템 연결, 엘라스틱 IP/퍼블릭 IP 주소 관리, VPC/프라이빗 주소 관리, 이름 확인과 같은 네트워크 고려 사항은 클라우드의 리소스를 활용하기 위한 기본 사항입니다. 중첩 및 경합 위험을 줄이기 위해서는 구축을 제대로 계획하고 문서화하는 것이 핵심입니다.

모범 사례:

- **데이터 센터로 다시 연결이 필요하지 않음**
- **AWS와 온프레미스 환경 간 고가용성 연결(해당하는 경우)** 해당하는 경우 복수의 DX 회로, 복수의 VPN 터널, AWS Marketplace 어플라이언스.
- **워크로드 사용자를 위한 고가용성 네트워크 연결** 고가용성 로드 밸런싱 및/또는 프록시, DNS 기반 솔루션, AWS Marketplace 어플라이언스 등.
- **비중첩 프라이빗 IP 주소 범위** Virtual Private Cloud에서 가상 IP 주소 범위 및 서브넷을 사용하는 경우 이 두 개가 서로 중복되거나 여러 클라우드 환경 또는 온프레미스 환경을 중복 사용하면 안 됩니다.
- **IP 서브넷 할당** 개별 Amazon VPC IP 주소 범위는 가용 영역의 서브넷에 IP 주소를 할당하고 추후 확장을 고려하는 등 애플리케이션의 요구 사항을 수용할 수 있도록 충분히 커야 합니다.

변경 관리

REL 3. How does your system adapt to changes in demand?

확장 가능한 시스템은 리소스를 자동으로 추가하거나 제거할 수 있는 탄력성을 갖추고 있으므로 어떤 상황에서도 수요에 빈틈없이 대응할 수 있습니다.

모범 사례:

- **자동 조정** 확장 가능한 서비스를 자동으로 사용합니다(예: Amazon S3, Amazon CloudFront, Auto Scaling, Amazon DynamoDB, AWS Elastic Beanstalk 등).
- **부하 테스트** 부하 테스트 방법을 채택하여 조정 활동이 애플리케이션 요구 사항을 충족하는지 여부를 평가합니다.

REL 4. How are you monitoring AWS resources?

로그와 측정 지표는 애플리케이션의 상태를 파악하기 위한 강력한 도구입니다. 로그와 측정 지표들을 모니터링하고 임계값을 초과했거나 중요한 이벤트가 발생한 경우 알림을 보내도록 시스템을 구성할 수 있습니다. 임계값을 초과했거나 오류가 발생한 경우 자동으로 자체 복구를 하거나 변화에 대응하여 조정할 수 있도록 시스템을 설계하는 것이 가장 좋습니다.

모범 사례:

- **모니터링** Amazon CloudWatch 또는 타사 도구를 통해 애플리케이션을 모니터링합니다.
- **알림** 중요한 이벤트가 발생하는 경우 알림을 받도록 설정합니다.
- **자동 응답** 오류가 감지된 경우 자동으로 조치를 취하도록 합니다(예: 실패한 구성 요소 대체).

REL 5. How are you executing change?

환경에 대한 제어되지 않은 변경은 변경의 영향을 예측하기 어렵게 만듭니다. 애플리케이션과 운영 환경이 알려진 소프트웨어를 실행하고 있으며 예측 가능한 방식으로 패치 또는 교체될 수 있도록 프로비저닝된 AWS 리소스 및 애플리케이션에 대한 제어된 변경이 필요합니다.

모범 사례:

- **자동화** 배포 및 패치 적용을 자동화합니다.

장애 관리

REL 6. How are you backing up your data?

데이터, 애플리케이션 및 운영 환경(애플리케이션으로 구성된 운영 체제로 정의됨)을 백업하여 평균 복구 시간(MTTR) 및 목표 복구 시점(RPO) 요구 사항을 충족시키십시오.

모범 사례:

- **자동 백업** AWS 기능, AWS Marketplace 솔루션 또는 타사 소프트웨어를 사용하여 백업을 자동화합니다.
- **정기적인 복구 테스트** 복구 테스트를 통해 백업 프로세스 구현이 RTO 및 RPO를 충족하는지 확인합니다.

REL 7. How does your system withstand component failures?

애플리케이션에 명시적으로든 암시적으로든 고가용성과 낮은 평균 복구 시간(MTTR)에 대한 요구 사항이 있습니까? 만약 그렇다면 복원성을 갖추도록 애플리케이션을 구축하고 중단에 대처할 수 있도록 배포합니다. 높은 가용성을 달성하려면 여러 물리적 위치에 배포해야 합니다. 복원성을 갖추도록 개별 계층(예: 웹 서버, 데이터베이스)을 구축합니다. 여기에는 모니터링, 자체 복구, 중요한 이벤트 중단 및 오류에 대한 알림 등이 포함됩니다.

모범 사례:

- **다중 AZ/리전** 애플리케이션 로드를 여러 가용 영역/리전(예: DNS, ELB, Application Load Balancer, API Gateway)으로 분산시킵니다.
- **소결합 종속성** 예를 들어 대기열 시스템, 스트리밍 시스템, 워크플로우, 로드 밸런서 등을 사용합니다.
- **단계적 기능 축소** 특정 구성 요소의 종속성이 정상 상태가 아닐 경우 구성 요소 자체가 이상으로 보고되지 않습니다. 해당 구성 요소는 계속해서 제한된 방식으로 요청을 처리할 수 있습니다.
- **자동 복구** 자동 기능을 사용하여 오류를 감지하고 문제 해결 작업을 수행합니다. 시스템 상태를 지속적으로 모니터링하고 중요한 이벤트에 대한 알림을 수신하도록 계획합니다.

REL 8. How are you testing your resiliency?

복원성을 테스트할 때 프로덕션에서만 부상하는 지연된 버그가 발견될 수 있습니다. 실전 연습을 통해 정기적으로 절차를 실습하면 조직이 원활하게 절차를 실행하는 데 도움이 됩니다.

모범 사례:

- **지침서** 장애 시나리오에 대한 지침서를 마련합니다.
- **오류 삽입** 정기적으로 장애를 테스트하여(예: Chaos Monkey 사용) 장애 경로가 커버되는지 확인합니다.
- **실전 연습 예약**
- **근본 원인 분석(RCA)** 중요한 이벤트를 기반으로 시스템 장애를 검토하여 아키텍처를 평가합니다.

REL 9. How are you planning for disaster recovery?

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

모범 사례:

- **목표 정의** RTO와 RPO를 정의합니다.
- **재해 복구** DR 전략을 세웁니다.
- **구성 드리프트** DR 사이트/리전에서 Amazon 머신 이미지(AMI)와 시스템 구성 상태가 최신인지 확인합니다.
- **DR 테스트 및 검증** DR에 대한 장애 조치를 정기적으로 테스트하여 RTO와 RPO를 충족하는지 확인합니다.
- **자동 복구 구현** AWS 및/또는 타사 도구를 사용하여 시스템 복구를 자동화합니다.

성능 기반

Selection

PERF 1. How do you select the best performing architecture?

특정 시스템에 대한 최적의 솔루션은 워크로드에 따라 달라지며, 흔히 여러 접근 방식이 복합적으로 사용됩니다. 제대로 구축된 시스템은 여러 솔루션을 사용하며 성능을 향상하기 위해 다양한 기능을 활용합니다.

모범 사례:

- **벤치마킹** AWS에서 알려진 워크로드에 대한 부하 테스트를 수행하고 이를 바탕으로 최적의 선택을 평가합니다.
- **부하 테스트** 여러 리소스 유형과 크기를 사용하여 AWS에 최신 시스템 버전을 배포하고, 모니터링을 통해 성능 측정치를 캡처한 다음, 성능/비용 계산 결과에 따라 선택합니다.

PERF 2. How did you select your compute solution?

특정 시스템에 가장 적합한 컴퓨팅 솔루션은 애플리케이션의 설계, 사용 패턴 및 구성 설정에 따라 달라질 수 있습니다. 아키텍처는 다양한 구성 요소마다 서로 다른 컴퓨팅 솔루션을 사용하고 성능을 개선하기 위해 다양한 기능을 이용할 수 있습니다. 아키텍처에 적합하지 않은 컴퓨팅 솔루션을 선택하면 성능 효율이 저하될 수 있습니다.

모범 사례:

- **옵션 고려** 최상의 성능을 얻기 위해 인스턴스, 컨테이너 및 기능을 사용하는 다양한 옵션을 고려합니다.
- **인스턴스 구성 옵션** 인스턴스를 사용하는 경우 제품군, 인스턴스 크기, 기능(GPU, I/O, 버스트 가능)과 같은 구성 옵션을 고려합니다.
- **컨테이너 구성 옵션** 컨테이너를 사용하는 경우 컨테이너의 메모리, CPU, 테넌시 구성과 같은 구성 옵션을 고려합니다.
- **기능 구성 옵션** 기능을 사용하는 경우 메모리, 런타임, 상태와 같은 구성 옵션을 고려합니다.
- **탄력성** 탄력성(예: Auto Scaling, Amazon EC2 Container Service(ECS), AWS Lambda)을 이용하여 수요 변화에 대처합니다.

PERF 3. How do you select your storage solution?

특정 시스템에 대한 최적의 스토리지 솔루션은 액세스 방식(블록, 파일 또는 객체), 액세스 패턴(무작위 또는 순차적), 필요한 처리량, 액세스 빈도(온라인, 오프라인, 보관), 업데이트 빈도(WORM, 동적), 가용성 및 내구성 제약에 따라 다릅니다. 제대로 구축된 시스템은 여러 스토리지 솔루션을 사용하며 성능을 향상하기 위해 다양한 기능을 활용합니다.

모범 사례:

- **특성 고려** 사용할 서비스(Amazon S3, Amazon EBS, Amazon Elastic File System(EFS), EC2 인스턴스 스토어)를 선택하는 데 필요한 다양한 특성(예: 공유 가능, 파일 크기, 캐시 크기, 액세스 패턴, 지연 시간, 처리량, 데이터 지속성)을 고려합니다.
- **구성 옵션 고려** PIOPS, SSD, 마그네틱, Amazon S3 Transfer Acceleration과 같은 구성 옵션을 고려합니다.
- **액세스 패턴** 액세스 패턴(예: 스트라이핑, 키 배포, 파티셔닝)을 기반으로 스토리지 시스템 사용 방식을 최적화합니다.

PERF 4. How do you select your database solution?

특정 시스템에 대한 최적의 데이터베이스 솔루션은 가용성, 일관성, 파티션 허용치, 지연 시간, 내구성, 확장성 및 쿼리 용량에 대한 요구 사항에 따라 다릅니다. 많은 시스템들은 다양한 하위 시스템에 대해 서로 다른 데이터베이스 솔루션을 사용하며, 성능을 향상시키기 위해 다양한 기능을 사용합니다. 시스템에 적합하지 않은 데이터베이스 솔루션 및 기능을 선택하면 성능 효율이 저하될 수 있습니다.

모범 사례:

- **특성 고려** 가장 성능이 우수한 데이터베이스(관계형, No-SQL, 웨어하우스, 인 메모리)를 선택할 수 있도록 다양한 특성(예: 가용성, 일관성, 파티션 허용오차, 지연 시간, 내구성, 확장성, 쿼리 용량)을 고려합니다.
- **구성 옵션 고려** 스토리지 최적화, 데이터베이스 레벨 설정, 메모리, 캐시와 같은 구성 옵션을 고려합니다.

- **액세스 패턴 고려** 액세스 패턴(예: 인덱스, 키 배포, 파티셔닝, 수평적 조정)을 기반으로 데이터베이스 시스템 사용 방식을 최적화합니다.
- **다른 접근 방식 고려** 검색 인덱스, 데이터 웨어하우스, 빅 데이터 등 쿼리 가능한 데이터를 제공하기 위한 다른 접근 방식을 고려합니다.

PERF 5. How do you configure your networking solution?

특정 시스템에 대한 최적의 네트워크 솔루션은 지연 시간, 처리량 요구 사항 등에 따라 다릅니다. 사용자 또는 온프레미스 리소스와 같은 물리적 제약 조건이 위치 옵션을 좌우하며, 이는 에지 기술 또는 리소스 배치를 사용하여 상쇄될 수 있습니다.

모범 사례:

- **위치 고려** 네트워크 지연 시간을 줄이기 위해 위치 옵션(예: 리전, 가용 영역, 배치 그룹, 에지)을 고려합니다.
- **제품 기능 고려** 네트워크 트래픽을 최적화하기 위해 제품 기능을 고려합니다(예: EC2 인스턴스 네트워크 용량, 초대고용량 네트워크 인스턴스 유형, Amazon EBS 최적화 인스턴스, Amazon S3 Transfer Acceleration, Dynamic Amazon CloudFront).
- **네트워킹 기능 고려** 네트워크 거리 또는 지터를 줄이기 위해 네트워킹 기능을 고려합니다(예: Amazon Route 53 지연 시간 라우팅, Amazon VPC 엔드포인트, AWS Direct Connect).
- **적절한 NACLS** 네트워크 처리량을 유지하기 위한 최소한의 NACLS 세트를 사용합니다.
- **암호화 오프로드 고려** 로드 밸런싱을 사용한 암호화 종단(TLS) 오프로드를 고려합니다.
- **프로토콜 고려** 네트워크 성능을 최적화하기 위해 어떤 프로토콜이 필요한지 고려합니다.

복습

PERF 6. How do you ensure that you continue to have the most appropriate resource type as new resource types and features are introduced?

솔루션을 설계할 때 선택 가능한 옵션 세트는 유한합니다. 하지만 시간이 지나면서 아키텍처의 성능을 개선시킬 수 있는 새로운 기술 및 접근 방식을 사용할 수 있게 됩니다.

모범 사례:

- **검토** 새로운 리소스의 유형 및 크기를 검토하는 프로세스를 마련합니다. 성능 테스트를 다시 실행하여 성능 효율이 개선되는지 평가합니다.

모니터링

PERF 7. How do you monitor your resources post-launch to ensure they are performing as expected?

시스템 성능은 내부 및/또는 외부 요인으로 인해 시간이 지남에 따라 저하될 수 있습니다. 시스템 성능을 모니터링하면 성능 저하 여부를 확인하여 내부 또는 외부 요인(운영 체제 또는 애플리케이션 부하 등)에 대한 조치를 취할 수 있습니다.

모범 사례:

- **모니터링** Amazon CloudWatch, 타사 또는 사용자 지정 모니터링 도구를 사용하여 성능을 모니터링합니다.
- **경보 기반 알림** 측정치가 안전 범위를 벗어났을 경우 모니터링 시스템에서 자동 알림을 받습니다.
- **트리거 기반 작업** 자동으로 문제에 대한 조치를 취하거나 에스컬레이션하도록 경보를 설정합니다.

트레이드오프

PERF 8. How do you use tradeoffs to improve performance?

솔루션을 설계할 때는 트레이드오프를 적극적으로 고려하면 최적의 접근 방식을 선택할 수 있습니다. 일관성, 내구성 및 공간을 위해 시간과 지연 시간을 희생함으로써 보다 고성능을 제공할 수 있는 경우가 자주 있습니다.

모범 사례:

- **서비스 고려** Amazon ElastiCache, Amazon CloudFront, AWS Snowball과 같은 성능 개선 서비스를 사용합니다.
- **패턴 고려** 성능을 개선하기 위해 캐싱, 읽기 전용 복제본, 셰이딩, 압축, 버퍼링과 같은 패턴을 사용합니다.

비용 최적화 기반

비용 효율적인 리소스

COST 1. Are you considering cost when you select AWS services for your solution?

Amazon EC2, Amazon EBS, Amazon S3 등은 "기본적인" AWS 서비스입니다. Amazon RDS, Amazon DynamoDB 등의 관리형 서비스는 "더 높은 수준의" AWS 서비스입니다. 적절한 기본 및 관리형 서비스를 선택하여 아키텍처를 비용에 대해 최적화할 수 있습니다. 예를 들어 관리형 서비스를 사용하면 관리 및 운영상의 오버헤드를 상당수 줄이거나 제거하여 애플리케이션과 비즈니스 관련 작업에 집중할 수 있습니다.

모범 사례:

- **비용 절감을 위한 서비스 선택** 서비스를 분석하여 비용 절감을 위해 사용할 수 있는 서비스를 확인합니다.
- **라이선스 비용에 대해 최적화**
- **서버리스 및 컨테이너 기반 접근 방식을 사용하여 최적화** AWS Lambda, Amazon S3 웹 사이트, Amazon DynamoDB 및 Amazon ECS를 사용하여 비용을 절감합니다.
- **적절한 스토리지 솔루션을 사용하여 최적화** 사용 패턴을 기반으로 가장 비용 효율적인 스토리지 솔루션을 사용합니다(예: Amazon EBS 콜드 스토리지, Amazon S3 Standard-Infrequent Access, Amazon Glacier 등).
- **적절한 데이터베이스를 사용하여 최적화** Amazon Relational Database Service(RDS)(Postgres, MySQL, SQL Server, Oracle Server) 또는 Amazon DynamoDB(또는 기타 키-값 저장소, NoSQL 대안)를 사용합니다.
- **다른 애플리케이션 수준 서비스를 사용하여 최적화** Amazon Simple Queue Service(SQS), Amazon Simple Notification Service(SNS) 및 Amazon Simple Email Service(SES)를 사용합니다.

COST 2. Have you sized your resources to meet your cost targets?

현재 작업에 적합한 AWS 리소스 크기를 선택하도록 합니다. AWS는 선택한 리소스 유형이 워크로드에 최적화되도록 벤치마킹 평가를 사용하도록 권장합니다.

모범 사례:

- **측정치 기반 리소스 크기 결정** 성능 측정치를 활용해 적절한 크기/유형을 선택하여 비용에 대해 최적화합니다. Amazon EC2, Amazon DynamoDB, Amazon EBS(프로비저닝된 IOPS), Amazon RDS, Amazon EMR, 네트워킹 등의 서비스에 대해 처리량, 크기 및 스토리지를 적절하게 프로비저닝합니다.

COST 3. Have you selected the appropriate pricing model to meet your cost targets?

워크로드에 가장 적합한 요금 모델을 사용하여 비용을 최소화합니다. 최적의 배포는 완전한 온디맨드 인스턴스이거나, 온디맨드와 예약 인스턴스를 조합한 것일 수 있습니다. 또는 필요할 경우 스팟 인스턴스를 포함시킬 수 있습니다.

모범 사례:

- **예약 용량 및 약정 거래** 정기적으로 사용량을 분석하여 적절히 예약 인스턴스를 구매합니다(예: Amazon EC2, Amazon DynamoDB, Amazon S3, Amazon CloudFront 등).
- **스팟** 일부 워크로드(예: 배치, EMR 등)에 스팟 인스턴스(예: 스팟 블록, 플리트)를 사용합니다.
- **리전 비용 고려** 리전 선택 시 비용을 고려합니다.

공급과 수요의 균형

COST 4. How do you make sure your capacity matches but does not substantially exceed what you need?

비용과 성능 면에서 균형을 이룬 아키텍처를 구축하려면 지불하는 모든 요소를 사용하고 잘 사용하지 않는 인스턴스가 없도록 해야 합니다. 어느 쪽으로든 사용률 측정치가 편향되면 운영 비용(과다 사용으로 인한 성능 저하) 또는 (오버 프로비저닝으로 인한) AWS 비용이 과도하게 발생하여 비즈니스에 부정적인 영향을 미칠 수 있습니다.

모범 사례:

- **수요 기반 접근** Auto Scaling을 사용하여 변화하는 수요에 대응합니다.
- **버퍼 기반 접근** 작업을 버퍼링하여(예: Amazon Kinesis 또는 Amazon Simple Queue Service(SQS)를 사용) 작업을 처리하는 데 충분한 용량이 확보될 때까지 작업을 연기합니다.
- **시간 기반 접근** 시간 기반 접근 방식의 예로는 일조 기준, 주말 동안 개발 및 테스트 인스턴스 끄기, 분기 또는 연간 일정 기준(예: 블랙 프라이데이)이 있습니다.

비용 인지

COST 5. Did you consider data-transfer charges when designing your architecture?

데이터 전송 요금을 모니터링하여 이러한 비용 중 일부를 절감하는 설계 결정을 내릴 수 있도록 합니다. 예를 들어, Amazon S3 버킷에서 최종 사용자에게 직접 콘텐츠를 서비스하는 콘텐츠 공급업체인 경우 Amazon CloudFront 콘텐츠 전송 네트워크(CDN)로 콘텐츠를 푸시하면 비용을 대폭 절감할 수 있습니다. 작지만 효과적인 설계 변경으로 운영 비용을 크게 절감할 수 있습니다.

모범 사례:

- **최적화** 데이터 전송을 최적화하도록 설계합니다(애플리케이션 설계, WAN 가속화, 다중 AZ, 리전 선택 등).
- **CDN** 해당하는 경우 CDN을 사용합니다.
- **AWS Direct Connect** 상황을 분석하여 해당하는 경우 AWS Direct Connect를 사용합니다.

COST 6. How are you monitoring usage and spending?

정책과 절차를 수립하여 비용을 모니터링하고 관리하고 적절하게 할당합니다. AWS에서 제공하는 도구를 사용하여 사용자와 사용 항목 및 해당 비용을 확인합니다. 이를 통해 비즈니스 요구 사항과 팀에서 수행하는 작업을 보다 심도 있게 이해할 수 있습니다.

모범 사례:

- **모든 리소스에 태그 지정** 모든 태그 지정 가능 리소스에 태그를 지정하여 청구서의 변경 내용과 인프라 및 사용량의 변경 내용을 연결지을 수 있습니다.
- **결제 및 비용 관리 도구 활용** 결제 세부 보고서 또는 Cost Explorer를 로드하고 해석할 수 있는 표준 프로세스를 마련합니다. 해당하는 경우 Amazon CloudWatch 또는 타사 서비스(예: Cloudability, CloudCheckr, CloudHealth)를 사용하여 정기적으로 사용량 및 지출을 모니터링합니다.
- **알림** 팀의 주요 구성원으로 하여금 지출이 정의된 한도를 초과하는지 여부를 알 수 있도록 합니다.
- **자금 중심 과금(Charge Back)/확인(Show Back) 방식** 이를 사용하여 비용 센터에 인스턴스와 리소스를 할당합니다(예: 태그 지정).

COST 7. Do you decommission resources that you no longer need or stop resources that are temporarily not needed?

해당하는 경우 필요한 프로세스 변경 또는 향상을 확인할 수 있도록 프로젝트 시작에서 종료에 이르기까지 변경 제어 및 리소스 관리를 실행합니다. AWS Support와 공동으로 워크로드 프로젝트를 최적화하는 방식에 대한 권장 사항을 도출합니다(예: Auto Scaling, AWS OpsWorks, AWS Data Pipeline 또는 다른 Amazon EC2 프로비저닝 접근 방식을 사용할 시기 또는 Trusted Advisor 비용 최적화 권장 사항을 검토).

모범 사례:

- **자동화** 중요하지 않거나 필요하지 않으며 사용률이 낮은 인스턴스 또는 리소스를 확인하고 폐기할 때 리소스 종료를 적절하게 처리하도록 시스템을 설계합니다.
- **프로세스 정의** 불필요한 리소스를 확인하고 폐기하기 위한 프로세스를 마련합니다.

COST 8. What access controls and procedures do you have in place to govern AWS usage?

목표를 달성하는 동시에 적절한 비용이 발생하도록 정책과 메커니즘을 수립합니다. 태깅 및 IAM 관리를 통해 견제와 균형의 접근 방식을 채택하여 비용을 크게 들이지 않고도 혁신을 이룰 수 있습니다.

모범 사례:

- **그룹 및 역할 설정**(예: Dev/Test/Prod) 관리 메커니즘을 사용하여 인스턴스를 실행할 사람과 각 그룹의 리소스를 관리합니다. (이는 AWS 서비스 또는 타사 솔루션에 적용됩니다.)
- **프로젝트 수명 주기 추적** 프로젝트, 팀 및 환경의 수명 주기를 추적, 측정 및 감사하여 불필요한 리소스 사용 및 비용 지불을 방지합니다.

시간에 따른 최적화

COST 9. How do you manage and/or consider the adoption of new services?

AWS에서 여러 새로운 서비스와 기능을 출시함에 따라 기존 설계를 검토하고 그것이 여전히 비용 효율적인지 확인해 봐야 합니다.

모범 사례:

- **비용 최적화 기능 설정**
- **검토** 새로운 서비스, 리스소 유형 및 크기를 검토하는 프로세스를 마련합니다. 성능 테스트를 다시 실행하여 비용이 절감되는지 평가합니다.

Operational Excellence Pillar

Prepare

OPS 1. What best practices for cloud operations are you using?

운영 우수성을 추진하려면 효과적인 준비가 필요합니다. 운영 체크리스트를 사용하여 워크로드가 프로덕션 운영을 위한 준비가 되었는지 확인합니다. 체크리스트를 사용하면 효과적 준비 없이 의도치 않게 프로덕션으로 전환하는 것을 방지할 수 있습니다.

모범 사례:

- **운영 체크리스트** 워크로드를 운영할 준비가 되었으면 평가에 사용할 운영 체크리스트를 작성합니다.
- **사전 예방적 계획** 비즈니스(예: 명성, 자금)에 중대한 영향을 미칠 수 있는 기회 및 위험 모두에 대비하여 이벤트(예: 마케팅 캠페인, 반짝 세일)에 대한 사전 예방적 계획을 수립합니다.
- **보안 체크리스트** 워크로드를 안전하게 운영할 준비가 되었으면 평가에 사용할 보안 체크리스트를 작성합니다(예: 제로 데이, DDoS, 훼손된 키).

OPS 2. How are you doing configuration management for your workload?

환경, 아키텍처, 그리고 해당 리소스에 대한 구성 매개 변수는 추적 및 문제 해결을 위해 손쉽게 구성 요소를 식별할 수 있도록 문서화해야 합니다. 구성 변경 역시 추적이 가능하고 자동화되어야 합니다.

모범 사례:

- **리소스 추적** 워크로드 내에서 리소스 및 해당 기능을 식별하는 방법을 수립합니다(예: 메타데이터 사용, 태그 지정).
- **설명서** 아키텍처를 문서화합니다(예: 코드로서의 인프라, CMDB, 다이어그램, 릴리스 정보).
- **운영 관련 교훈 축적** 장기에 걸쳐 운영을 통해 얻은 교훈을 축적합니다(예: wiki, 기술 자료, 티켓).

- **변경 불가능한 인프라** 패치를 적용하지 않고 재배포하는 변경 불가능한 인프라를 설정합니다.
- **자동 변경 절차** 변경 절차를 자동화합니다.
- **구성 관리 데이터베이스(CMDB)** CMDB에서 모든 변경 사항을 추적합니다.

운영

OPS 3. How are you evolving your workload while minimizing the impact of change?

자동화, 잦은 소규모 변경, 정기적 품질 보증 테스트, 변경 사항을 추적, 감사, 롤백 및 검토하기 위해 정의된 메커니즘에 초점을 맞춰야 합니다.

모범 사례:

- **배포 파이프라인** CI/CD 파이프라인을 구축합니다(예: 소스 코드, 리포지토리, 빌드 시스템, 배포 및 테스트 자동화).
- **릴리스 관리 프로세스** 릴리스 관리 프로세스(수동 또는 자동)를 수립합니다.
- **소규모 증분식 변경** 시스템 구성 요소의 소규모 증분 버전을 릴리스할 수 있도록 합니다.
- **취소 가능 변경** 운영 문제를 유발한 변경 사항을 되돌릴 수 있도록 준비를 합니다(예: 롤백, 기능 전환).
- **위험 완화 전략** 블루/그린, Canary 및 A/B 테스팅과 같은 위험 완화 전략을 사용합니다.

OPS 4. How do you monitor your workload to ensure it is operating as expected?

시스템은 내부 및/또는 외부 요인으로 인해 시간이 지남에 따라 성능이 저하될 수 있습니다. 시스템 동작을 모니터링하여 이러한 성능 저하 요소를 식별하고 수정할 수 있습니다.

모범 사례:

- **모니터링** Amazon CloudWatch, 타사 또는 사용자 지정 모니터링 도구를 사용하여 성능을 모니터링합니다.
- **로그 집계** 여러 소스(예: 애플리케이션 로그, AWS 서비스별 로그, VPC 흐름 로그, CloudTrail)로부터 로그를 집계합니다.
- **경보 기반 알림** 측정치가 안전 범위를 벗어났을 경우 모니터링 시스템에서 자동 알림을 받습니다.
- **트리거 기반 작업** 경보를 통해 자동으로 문제에 대한 조치를 취하거나 에스컬레이션합니다.

응답

OPS 5. How do you respond to unplanned operational events?

예기치 못한 운영 이벤트에 대한 대응을 자동화할 준비를 합니다. 여기에는 경보뿐 아니라 완화, 수정, 롤백 및 복구도 포함됩니다.

모범 사례:

- **지침서** 준수할 지침서(예: 대기 프로세스, 워크플로 체인, 에스컬레이션 프로세스)를 작성하고 정기적으로 업데이트합니다.
- **RCA 프로세스** 문제를 해결, 문서화 및 수정하여 재발하지 않도록 하기 위한 RCA 프로세스를 마련합니다.
- **자동 대응** 예기치 못한 운영 이벤트를 자동 대응(예: Auto Scaling, Support API)을 통해 안정적으로 처리합니다.

OPS 6. How do you manage escalation when you respond to unplanned operational events?

예기치 못한 운영 이벤트에 대한 대응은 이해관계자, 에스컬레이션 프로세스 및 절차를 포함하는 사전 정의된 지침서를 따라야 합니다. 에스컬레이션 경로를 정의하되 기능 및 계층 에스컬레이션 기능을 모두 포함해야 합니다. 계층 에스컬레이션은 자동화해야 하며 에스컬레이션된 우선 순위가 이해관계자 알림을 생성해야 합니다.

모범 사례:

- **적절히 문서화 및 프로비저닝** 에스컬레이션이 발생할 경우 경보를 수신해야 할 이해관계자 및 시스템을 설정합니다.
- **대기열 기반 접근 방식의 기능 에스컬레이션** 우선순위, 영향 및 유입 메커니즘을 기반으로 적절한 기능 팀 사이에서 에스컬레이션합니다.
- **계층 에스컬레이션** 요구 또는 시간 기반 접근 방식을 사용합니다. 인시던트의 영향, 규모 또는 해결/복구 시간이 증가할수록 우선순위가 상승합니다.
- **외부 에스컬레이션 경로** 에스컬레이션 경로에 외부 지원, AWS Support, AWS 파트너 및 타사 지원 계약을 포함시킵니다.
- **계층 우선순위 에스컬레이션 자동화** 요구 또는 시간 임계값이 초과되면 우선순위가 자동으로 상승합니다.