

AWS Well-Architected Framework

운영 우수성 원칙



운영 우수성 원칙: AWS Well-Architected Framework

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

- 개요 및 소개 1
 - 소개 1
- 운영 우수성 3
 - 설계 원칙 3
 - 정의 4
- 조직 6
 - 조직 우선순위 6
 - OPS01-BP01 고객 요구 사항 평가 6
 - OPS01-BP02 내부 고객 요구 평가 8
 - OPS01-BP03 거버넌스 요구 사항 평가 9
 - OPS01-BP04 규정 준수 요구 사항 평가 11
 - OPS01-BP05 위협 환경 평가 14
 - OPS01-BP06 장점과 위험을 관리하면서 절충점 평가 16
- 운영 모델 19
 - 운영 모델 2x2 표현 20
 - 관계 및 소유권 28
- 조직 문화 46
 - OPS03-BP01 경영진의 후원 제공 47
 - OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여 50
 - OPS03-BP03 에스컬레이션 장려 52
 - OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션 55
 - OPS03-BP05 실험 장려 59
 - OPS03-BP06 팀원의 기술 역량 유지와 강화 장려 62
 - OPS03-BP07 팀에 적절한 리소스 제공 65
- 준비 68
 - 관찰성 구현 68
 - OPS04-BP01 핵심 성과 지표 파악 69
 - OPS04-BP02 애플리케이션 원격 측정 구현 71
 - OPS04-BP03 사용자 경험 원격 측정 구현 74
 - OPS04-BP04 종속성 원격 측정 구현 76
 - OPS04-BP05 분산 추적 구현 79
- 운영을 고려한 설계 81
 - OPS05-BP01 버전 관리 사용 82
 - OPS05-BP02 변경 사항 테스트 및 검증 83

| | |
|--|-----|
| OPS05-BP03 구성 관리 시스템 사용 | 87 |
| OPS05-BP04 빌드 및 배포 관리 시스템 사용 | 89 |
| OPS05-BP05 패치 관리 수행 | 92 |
| OPS05-BP06 설계 표준 공유 | 95 |
| OPS05-BP07 코드 품질 개선을 위한 사례 구현 | 97 |
| OPS05-BP08 여러 환경 사용 | 100 |
| OPS05-BP09 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용 | 101 |
| OPS05-BP10 통합 및 배포 완전 자동화 | 102 |
| 배포 위험 완화 | 104 |
| OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립 | 104 |
| OPS06-BP02 테스트 배포 | 106 |
| OPS06-BP03 안전한 배포 전략 채택 | 109 |
| OPS06-BP04 테스트 및 롤백 자동화 | 112 |
| 운영 준비 상태 및 변경 관리 | 115 |
| OPS07-BP01 직원의 역량 확보 | 116 |
| OPS07-BP02: 일관된 방식으로 운영 준비 상태 검토 | 117 |
| OPS07-BP03 런북을 사용한 절차 수행 | 121 |
| OPS07-BP04 플레이북을 사용하여 문제 조사 | 125 |
| OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정 | 129 |
| OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 활성화 | 130 |
| 운영 | 134 |
| 워크로드 관찰성 활용 | 134 |
| OPS08-BP01 워크로드 지표 분석 | 135 |
| OPS08-BP02 워크로드 로그 분석 | 137 |
| OPS08-BP03 워크로드 추적 데이터 분석 | 139 |
| OPS08-BP04 실행 가능한 알림 생성 | 142 |
| OPS08-BP05 대시보드 만들기 | 145 |
| 운영 상태 파악 | 148 |
| OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정 | 148 |
| OPS09-BP02 상태 및 추세를 전달하여 운영에 대한 가시성 확보 | 150 |
| OPS09-BP03 운영 지표 검토 및 개선 우선 순위 지정 | 151 |
| 이벤트 대응 | 153 |
| OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용 | 154 |
| OPS10-BP02 알림별 프로세스 마련 | 158 |
| OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정 | 162 |
| OPS10-BP04 에스컬레이션 경로 정의 | 165 |

OPS10-BP05 서비스에 영향을 미치는 이벤트에 대한 고객 커뮤니케이션 계획 정의 167

OPS10-BP06 대시보드를 통해 상태 전달 170

OPS10-BP07 이벤트 대응 자동화 172

개선 175

 학습, 공유 및 개선 175

 OPS11-BP01 지속적인 개선을 위한 프로세스 마련 175

 OPS11-BP02 인시던트 사후 분석 수행 177

 OPS11-BP03 피드백 루프 구현 179

 OPS11-BP04 지식 관리 수행 183

 OPS11-BP05 개선 추진 요인 정의 185

 OPS11-BP06 인사이트 검증 187

 OPS11-BP07 운영 지표 검토 수행 188

 OPS11-BP08 파악한 내용 문서화 및 공유 190

 OPS11-BP09 개선을 위한 시간 할애 192

결론 194

기여자 195

추가 자료 196

문서 개정 197

운영 우수성 원칙 - AWS Well-Architected Framework

게시 날짜: 2024년 6월 27일 ([문서 개정](#))

이 백서에서는 AWS Well-Architected Framework의 운영 우수성 원칙에 대해 중점적으로 설명합니다. 또한 AWS 워크로드 설계, 제공 및 유지 관리에 모범 사례를 적용할 때 참조할 수 있는 지침을 제공합니다.

소개

유요 [AWS Well-Architected Framework](#) 는 AWS에서 워크로드를 구축할 때 내리는 의사 결정의 이점과 위험성을 이해하는 데 도움이 됩니다. 이 프레임워크를 사용하면 클라우드에서 안정적이고 안전하며 효율적이고 경제적이면서 지속 가능한 워크로드를 설계하고 운영하기 위한 운영 및 설계 모범 사례를 알아볼 수 있습니다. 사용자는 모범 사례를 기준으로 운영 상태와 아키텍처를 일관적으로 측정하고 개선 영역을 식별할 수 있습니다. 운영을 염두에 두고 Well-Architected 워크로드를 제대로 설계하면 비즈니스 성공 가능성을 높일 수 있습니다.

이 프레임워크는 다음 6가지 원칙을 기반으로 합니다.

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

이 백서에서는 운영 우수성 원칙을 중점적으로 다루며 이 원칙을 Well-Architected 솔루션의 토대로 적용하는 방법을 중점적으로 설명합니다. 운영이 지원 대상 사업부 및 개발 팀과 분리된 고유한 부문으로 인식되는 환경에서는 운영 우수성을 달성하기가 어렵습니다. 이 백서의 사례를 도입하면 상태에 대한 인사이트를 제공하고, 효과적이고 효율적인 작업 및 이벤트 대응이 가능하며, 계속해서 비즈니스 목표를 개선하고 지원하는 아키텍처를 구축할 수 있습니다.

이 문서는 최고 기술 책임자(CTO), 아키텍트, 개발자와 같은 기술 업무 담당자와 운영 팀원을 대상으로 작성되었습니다. 이 백서의 내용을 읽고 나면 운영 우수성을 위한 클라우드 아키텍처를 설계할 때 사용할 AWS 모범 사례와 전략을 파악할 수 있습니다. 이 백서는 구현 세부 정보나 아키텍처 패턴을 제공하지 않습니다. 하지만 이 정보를 확인할 수 있는 적절한 리소스에 대한 참조는 포함되어 있습니다.

운영 우수성

Amazon에서는 소프트웨어를 올바르게 구축하는 동시에 우수한 고객 환경을 지속적으로 제공하기 위한 노력으로 운영 우수성을 정의합니다. 여기에는 팀 구성, 워크로드 설계, 규모에 따른 운영, 장기적 발전에 관한 모범 사례가 포함되어 있습니다. 팀은 운영 우수성을 통해 고객에게 도움이 되는 새로운 기능을 구축하는 데 더 많은 시간을 할애하고 유지 보수 및 급한 문제를 해결하는 데 걸리는 시간을 단축할 수 있습니다. Amazon에서는 올바른 구축을 위해 원활하게 운영되는 시스템, 조직과 팀을 위한 균형 잡힌 워크로드, 특히 우수한 고객 경험으로 이어지는 모범 사례를 고려합니다.

운영 우수성의 목표는 새로운 기능과 버그 수정을 빠르고 안정적으로 고객에게 제공하는 것입니다. 운영 우수성에 투자하는 조직은 새로운 기능을 구축하고, 변경하고, 장애를 처리하면서 고객이 지속적으로 만족하게 할 수 있습니다. 운영 우수성은 개발자가 고품질의 결과를 일관되게 달성할 수 있도록 지원함으로써 지속적인 통합과 지속적인 제공(CI/CD)을 지향합니다.

설계 원칙

클라우드에서 운영 우수성을 달성하기 위한 설계 원칙은 다음과 같습니다.

- **비즈니스 성과를 중심으로 팀 구성:** 팀의 비즈니스 성과를 달성할 수 있는 능력은 경영진의 비전, 효과적인 운영, 비즈니스에 부합하는 운영 모델에서 비롯됩니다. 경영진은 팀이 가장 효율적인 방식으로 운영하고 비즈니스 성과를 달성하도록 장려하는 적절한 클라우드 운영 모델을 활용하여 CloudOps 혁신에 전적으로 투자하고 전념해야 합니다. 적절한 운영 모델은 규모 조정 및 최적화로 생산성을 높이고 민첩성, 대응성, 적응을 통한 차별화를 위해 인력, 프로세스 및 기술 역량을 사용합니다. 조직의 장기적 비전은 목표에 반영되고 목표는 기업 전반의 이해관계자 및 클라우드 서비스 소비자에게 전달됩니다. 목표와 운영 KPI는 모든 수준에서 연계됩니다. 이러한 관행은 다음과 같은 설계 원칙을 구현함으로써 얻을 수 있는 장기적 가치를 뒷받침합니다.
- **실행 가능한 인사이트를 위한 관찰성 구현:** 워크로드 동작, 성능, 신뢰성, 비용 및 상태를 포괄적으로 이해합니다. 핵심 성과 지표(KPI)를 설정하고 관찰성 텔레메트리를 활용하여 정보에 입각한 결정을 내리고 비즈니스 성과가 위협에 처했을 때 즉각적인 조치를 취합니다. 실행 가능한 관찰성 데이터를 기반으로 성능, 신뢰성, 비용을 선제적으로 개선합니다.
- **자동화가 가능한 부분은 안전하게 자동화:** 클라우드에서는 애플리케이션 코드에 사용되었던 동일한 엔지니어링 원칙을 전체 환경에 적용할 수 있습니다. 전체 워크로드와 해당 작업(애플리케이션, 인프라, 구성, 프로시저)을 코드로 정의하고 업데이트할 수 있습니다. 그런 다음 이벤트에 대한 응답으로 워크로드 작업을 시작하여 워크로드 작업을 자동화할 수 있습니다. 클라우드에서는 속도 제어, 오류 임계값, 승인을 비롯한 가드레일을 구성하여 자동화 안전을 실현할 수 있습니다. 효과적인 자동화를 통해 이벤트에 일관되게 대응하고, 인적 오류를 제한하고, 작업자 수고를 줄일 수 있습니다.

- 되돌릴 수 있도록 변경 사항을 조금씩 자주 적용: 구성 요소를 정기적으로 업데이트할 수 있도록 확장 가능하고 느슨하게 결합된 워크로드를 설계합니다. 자동화된 배포 기법과 소규모의 점진적인 변경을 함께 사용하면 영향 반경을 줄이고 장애 발생 시 더 빠르게 되돌릴 수 있습니다. 이를 통해 품질을 유지하고 시장 상황의 변화에 신속하게 적응하면서 워크로드에 유익한 변화를 가져올 수 있다는 자신감이 높아집니다.
- 운영 절차를 자주 개선: 워크로드가 변경됨에 따라 운영도 적절하게 변화시킵니다. 운영 절차를 사용할 때 개선할 여지가 있는지 확인합니다. 정기적으로 검토하여 모든 절차가 효과적이며 팀이 이러한 절차에 익숙한지 확인하고 검증합니다. 격차가 확인되면 그에 따라 절차를 업데이트합니다. 절차 업데이트를 모든 이해관계자와 팀에 전달합니다. 운영을 게임화하여 모범 사례를 공유하고 팀을 교육합니다.
- 장애 예측: 장애 시나리오를 구동하여 워크로드의 위험 프로파일과 비즈니스 성과에 미치는 영향을 이해함으로써 운영의 성공을 극대화합니다. 시뮬레이션에서 확인한 장애에 대한 절차의 효과와 팀의 대응을 테스트합니다. 테스트를 통해 확인된 미해결 위험을 관리하기 위해 정보에 입각한 결정을 내립니다.
- 모든 운영 이벤트와 지표를 바탕으로 학습: 모든 운영 이벤트와 장애로부터 배운 내용을 바탕으로 개선합니다. 파악한 내용을 팀 전반과 조직 전체에 공유합니다. 파악한 내용에서 운영이 비즈니스 성과에 어떻게 기여하는지에 대한 데이터와 일화를 강조해야 합니다.
- 관리형 서비스 사용: 가능한 경우 AWS 관리형 서비스를 사용하여 운영 부담을 줄입니다. 해당 서비스와의 상호 작용을 중심으로 운영 절차를 구축합니다.

정의

클라우드의 운영 우수성에는 4가지 모범 사례 영역이 있습니다.

- 조직
- 준비
- 운영
- 개선

조직의 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. 바람직한 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.

조직

조직의 우선순위, 조직 구조, 그리고 팀원들이 비즈니스 성과를 뒷받침할 수 있도록 조직에서 팀원을 지원하는 방식을 파악해야 합니다.

운영 우수성을 실현하려면 다음을 파악해야 합니다.

주제

- [조직 우선순위](#)
- [운영 모델](#)
- [조직 문화](#)

조직 우선순위

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. 주기적으로 우선순위를 검토하여 조직 요구 사항의 변화에 따라 우선순위를 업데이트합니다.

모범 사례

- [OPS01-BP01 고객 요구 사항 평가](#)
- [OPS01-BP02 내부 고객 요구 평가](#)
- [OPS01-BP03 거버넌스 요구 사항 평가](#)
- [OPS01-BP04 규정 준수 요구 사항 평가](#)
- [OPS01-BP05 위협 환경 평가](#)
- [OPS01-BP06 장점과 위험을 관리하면서 절충점 평가](#)

OPS01-BP01 고객 요구 사항 평가

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

원하는 결과:

- 고객 성과에서 시작하여 역방향으로 작업합니다.
- 운영 관행이 비즈니스 성과 및 목표를 어떻게 지원하는지 이해합니다.
- 모든 관련 당사자를 참여시킵니다.
- 고객의 요구를 파악할 수 있는 메커니즘이 있습니다.

일반적인 안티 패턴:

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

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

외부 고객의 비즈니스 목표, 요구 사항 및 우선순위 검토: 실무 팀, 개발 팀, 운영 팀 등의 주요 이해관계
자와 함께 외부 고객의 목표, 요구 사항 및 우선순위를 논의합니다. 이렇게 하면 비즈니스 및 고객 성과
달성에 필요한 운영 지원을 철저히 파악할 수 있습니다.

공유된 이해 관계 수립: 워크로드의 비즈니스 기능과 워크로드 작동 과정에서 각 팀의 역할을 비롯하여
이러한 요소가 내외부 고객의 공동 비즈니스 목표를 지원하는 방식에 대한 공유된 이해 관계를 정립합
니다.

리소스

관련 모범 사례:

- [OPS11-BP03 피드백 루프 구현](#)

OPS01-BP02 내부 고객 요구 평가

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

원하는 결과:

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

일반적인 안티 패턴:

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

모범 사례 구현의 이점: 내부 고객 요구 사항을 평가하고 이해하면 비즈니스 가치를 제공하기 위한 노력의 우선 순위를 정하는 방법을 알 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

리소스

관련 모범 사례:

- [OPS11-BP03 피드백 루프 구현](#)

OPS01-BP03 거버넌스 요구 사항 평가

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

원하는 결과:

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

일반적인 안티 패턴:

- 조직에서는 루트 계정에서 다중 인증을 사용해야 합니다. 이 요구 사항을 구현하지 못했으며 루트 계정이 손상되었습니다.
- 워크로드를 설계하는 동안 IT 부서에서 승인하지 않은 인스턴스 유형을 선택합니다. 워크로드를 시작할 수 없으므로 재설계를 수행해야 합니다.
- 재해 복구 계획을 마련해야 합니다. 재해 복구 계획을 마련하지 않아 워크로드에 오랜 중단이 발생합니다.
- 팀에서 새 인스턴스를 사용하려고 하지만, 이를 허용하도록 거버넌스 요구 사항이 업데이트되지 않았습니다.

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

- [OPS01-BP04 규정 준수 요구 사항 평가](#) - 규정 준수는 거버넌스와 유사하지만, 조직 외부에서 이루어집니다.

관련 문서:

- [AWS Management and Governance Cloud Environment Guide](#)(AWS 관리 및 거버넌스 클라우드 환경 가이드)
- [다중 계정 환경에서의 AWS Organizations 서비스 제어 정책 모범 사례](#)

- [Governance in the AWS 클라우드: The Right Balance Between Agility and Safety](#)(AWS 클라우드의 거버넌스: 민첩성과 안전성 사이의 적절한 균형)
- [거버넌스, 위험 및 규정 준수\(GRC\)란 무엇입니까?](#)

관련 동영상:

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

관련 예시:

- [AWS Config 규정 준수 팩 샘플](#)

관련 서비스:

- [AWS Config](#)
- [AWS Organizations - 서비스 제어 정책](#)

OPS01-BP04 규정 준수 요구 사항 평가

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

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

- [SEC01-BP03 제어 목표 파악 및 검증](#) - 보안 제어 목표는 전반적인 규정 준수의 중요한 부분입니다.
- [SEC01-BP06 파이프라인에서 보안 제어의 테스트 및 검증 자동화](#) - 파이프라인의 일부로 보안 제어를 검증합니다. 또한, 새 변경 사항에 대한 규정 준수 문서를 생성할 수 있습니다.
- [SEC07-BP02 데이터 보호 제어 정의](#) - 많은 규정 준수 프레임워크는 데이터 처리 및 스토리지 정책에 기반을 두고 있습니다.
- [SEC10-BP03 포렌식 역량 확보](#) - 포렌식 기능을 종종 규정 준수 감사에 사용할 수 있습니다.

관련 문서:

- [AWS 규정 준수 센터](#)
- [AWS 규정 준수 리소스](#)
- [AWS 위험 및 규정 준수 백서](#)
- [AWS 공동 책임 모델](#)
- [규정 준수 프로그램 제공 AWS 범위 내 서비스](#)

관련 동영상:

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

관련 예시:

- [PCI DSS 및 AWS의 AWS 기초 보안 모범 사례](#)

관련 서비스:

- [AWS Artifact](#)
- [AWS Audit Manager](#)
- [AWS Compute Optimizer](#)
- [AWS Security Hub](#)

OPS01-BP05 위협 환경 평가

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

[Well-Architected 프레임워크](#)에서는 학습, 평가, 개선을 강조합니다. 아키텍처를 평가하고 시간에 따라 확장 가능한 설계를 구현하는 일관된 접근 방식을 제공합니다. AWS에서는 [AWS Well-Architected Tool](#)를 제공하여 개발 전 접근 방식, 프로덕션 환경에 적용하기 전의 워크로드 상태, 프로덕션 환경에서의 워크로드 상태를 검토할 수 있게 합니다. 이를 최신 AWS 아키텍처 모범 사례와 비교하고, 워크로드의 전반적인 상태를 모니터링하며, 잠재적 위협에 대한 인사이트를 얻을 수 있습니다.

AWS 고객은 AWS 모범 사례를 기준으로 하여 [아키텍처를 평가](#)할 수 있는 미션 크리티컬 워크로드에 대한 안내형 Well-Architected Review 서비스를 이용할 수 있습니다. Enterprise Support 고객에게는 클라우드 운영 방식의 격차를 파악하는 데 사용할 수 있는 [Operations Review](#) 서비스가 제공됩니다.

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

[AWS Trusted Advisor](#)는 우선순위 결정에 도움이 될 수 있는 최적화 방안을 알려주는 핵심 검사 세트에 액세스할 수 있는 도구입니다. [Business 및 Enterprise Support 고객](#)에게는 우선순위를 더욱 세밀하게 다듬는 데 사용할 수 있는 추가 검사 기능이 제공됩니다. 이러한 기능을 사용하면 보안, 안정성, 성능 및 비용 최적화 영역을 중점적으로 확인할 수 있습니다.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점: 조직과 워크로드에 대한 위협을 식별하고 이해하면 해결해야 할 위협, 우선 순위 및 이에 필요한 리소스를 결정하는 데 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

- 위협 환경 평가: 주력할 영역을 결정할 때 영향력을 고려할 수 있도록 업무상의 위협 요소(예: 경쟁, 업무상의 위협/책임, 운영상의 위험, 정보 보안 위협)를 평가합니다.
 - [AWS 최신 보안 공지](#)
 - [AWS Trusted Advisor](#)
- 위협 모델 유지 관리: 잠재적 위협, 계획되어 실행 중인 완화 조치 및 각 우선순위를 식별하는 위협 모델을 수립하고 유지 관리합니다. 인시던트로 나타나는 위협의 가능성, 해당 인시던트로부터 복구하

는 비용 및 예상되는 피해, 이러한 인시던트를 방지하는 비용을 검토합니다. 위협 모델의 내용이 변경됨에 따라 우선순위를 수정합니다.

리소스

관련 모범 사례:

- [SEC01-BP07 위협 모델을 사용하여 위협 식별 및 완화 조치의 우선순위 지정](#)

관련 문서:

- [AWS 클라우드 규정 준수](#)
- [AWS 최신 보안 공지](#)
- [AWS Trusted Advisor](#)

관련 동영상:

- [AWS re:Inforce 2023 - A tool to help improve your threat modeling](#)

OPS01-BP06 장점과 위험을 관리하면서 절충점 평가

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

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

AWS를 활용하면 선택한 방식이 워크로드에 미치는 영향을 효과적으로 파악하도록 팀에 AWS와 해당 서비스 관련 정보를 제공할 수 있습니다. [AWS Support](#)([AWS 지식 센터](#), [AWS 토론 포럼](#) 및 [AWS Support 센터](#))와 [AWS 설명서](#)에 나와 있는 리소스를 사용하여 팀을 교육해야 합니다. 더 궁금한 점이 있으면 AWS Support로 문의하세요.

AWS는 또한 [Amazon Builders' Library](#)의 운영 모범 사례와 패턴을 공유합니다. [AWS 블로그](#) 및 [공식 AWS 팟캐스트](#)에서도 기타 여러 가지 유용한 정보를 확인할 수 있습니다.

원하는 결과: 클라우드 제공 조직의 모든 수준에서 중요한 의사 결정을 용이하게 하기 위해 명확하게 정의된 의사 결정 거버넌스 프레임워크가 마련됩니다. 이 프레임워크에는 위험 관리 대장, 의사 결정 권한이 부여된 정의된 역할, 내릴 수 있는 각 의사 결정 수준에 대한 정의된 모델 등의 기능이 포함됩니다. 이 프레임워크는 갈등 해결 방법, 제시해야 하는 데이터, 옵션의 우선 순위 지정 방법을 미리 정의하므로 일단 결정을 내리면 지체 없이 실행할 수 있습니다. 의사 결정 프레임워크에는 모든 결정의 이점과 위험을 검토하고 평가하여 장단점을 이해하기 위한 표준화된 접근 방식이 포함됩니다. 여기에는 규정 준수 요구 사항 충족과 같은 외부 요인이 포함될 수 있습니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점: 이사회 차원의 비즈니스 우선 순위를 완벽하게 조정 및 지원하고, 성공 달성에 따르는 위험을 이해하며, 정보에 입각한 결정을 내리고, 위험이 성공 가능성을 저해할 때 적절하게 조치합니다. 의사 결정의 영향과 결과를 이해하면 선택의 우선 순위를 정하고 리더가 더 빨리 합의에 도달하여 비즈니스 성과를 개선할 수 있습니다. 선택을 통해 얻을 수 있는 이점을 파악하고 조직에 미칠 수 있는 위험을 인식하면 사례에 의존하지 않고 데이터에 기반한 결정을 내리는 데 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

구현 단계

1. 종합적인 클라우드 거버넌스 프레임워크 내에서 이익 측정 사례를 공식화하세요.
 - a. 의사 결정의 중앙 통제와 일부 의사 결정에 대한 분산된 권한 간의 균형을 유지합니다.
 - b. 모든 의사 결정에 부과되는 부담스러운 의사 결정 프로세스로 인해 속도가 느려질 수 있다는 점을 이해하세요.
 - c. 의사 결정 프로세스에 외부 요인(예: 규정 준수 요구 사항)을 통합하세요.
2. 이해 상충의 대상이 되는 의사 결정을 진행해야 하는 사람을 포함하여 다양한 수준의 의사 결정에 대해 합의된 의사 결정 프레임워크를 수립합니다.
 - a. 되돌릴 수 없는 단방향 결정을 중앙 집중화합니다.
 - b. 하위 조직 리더가 양방향 결정을 내릴 수 있도록 합니다.
3. 이점과 위험을 이해하고 관리합니다. 결정을 통해 제공되는 이점과 관련 위험을 적절하게 절충합니다.
 - a. 이점 파악: 비즈니스 목표, 필요 사항 및 우선순위에 따른 이점을 파악합니다. 비즈니스 사례에 미치는 영향, 출시 시기, 보안, 안정성, 성능, 비용 등을 예로 들 수 있습니다.
 - b. 위험 요소 파악: 비즈니스 목표, 필요 사항 및 우선순위에 따른 위험 요소를 파악합니다. 출시 리드타임, 보안, 안정성, 성능 및 비용이 이에 해당합니다.
 - c. 위험 대비 이점 평가 및 정보에 입각한 의사 결정: 실무/개발/운영 팀을 비롯한 주요 이해관계자의 목표, 필요 사항 및 우선순위에 따라 이점과 위험의 영향을 파악합니다. 위험 발생 가능성 및 해당 위험이 주는 영향의 비용 대비 이점의 가치를 평가합니다. 예를 들어, 안정성보다 시장 진입 속도를 강조하면 경쟁 우위를 제공할 수 있습니다. 하지만 안정성 문제가 있는 경우 가동 시간이 단축될 수 있습니다.
4. 규정 준수 요구 사항 준수를 자동화하는 주요 결정을 프로그래밍 방식으로 적용합니다.
5. 가치 흐름 분석 및 LEAN과 같은 알려진 업계 프레임워크와 기능을 활용하여 현재 상태 성과와 비즈니스 지표의 기준을 정하고 이러한 지표의 개선을 위한 진행 상황을 정의합니다.

구현 계획의 작업 수준: 중간-높음

리소스

관련 모범 사례:

- [OPS01-BP05 위험 환경 평가](#)

관련 문서:

- [Amazon 1일 차 문화의 요소 | 고품질의 빠른 의사 결정 내리기](#)
- [클라우드 거버넌스](#)
- [Management & Governance Cloud Environment](#)
- [Governance in the Cloud and in the Digital Age: Parts One & Two](#)

관련 동영상:

- [Podcast | Jeff Bezos | On how to make decisions](#)

관련 예시:

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

운영 모델

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 종사하는 업계, 소속된 조직, 팀 구성, 그리고 워크로드의 특성에 따라 결정됩니다. 당연히 단일 운영 모델로는 모든 팀과 해당 워크로드를 지원할 수 없습니다.

조직에 존재하는 운영 모델의 수는 개발 팀의 수에 따라 늘어날 수 있습니다. 여러 운영 모델을 조합하여 사용해야 할 수도 있습니다.

표준을 채택하고 서비스를 사용하면 운영 모델에서 지원 부담을 줄이고 운영을 간소화할 수 있습니다. 공유 표준을 개발하면 표준을 이미 채택했고 새로운 기능을 채택하려는 팀이 많을수록 그 효과가 더 커집니다.

팀의 활동을 지원하는 데 있어서는 표준의 추가, 변경 및 예외 처리를 요청하는 메커니즘을 갖추어야 합니다. 이 옵션이 없으면 표준이 혁신의 제약 요인이 됩니다. 이점과 위험을 평가한 후 요청이 적절한지 판단하고 실현 가능한 경우에 요청을 승인해야 합니다.

책임을 잘 정의하면 상충하거나 중복되는 작업의 빈도를 줄일 수 있습니다. 사업, 개발 및 운영 팀 간에 강력한 연계와 관계를 쌓으면 비즈니스 성과를 달성하기가 더 쉽습니다.

운영 모델 2x2 표현

이러한 운영 모델 2x2 표현을 살펴보면 조직의 환경 내에서 팀 간 관계를 이해하는 데 도움이 됩니다. 이 그림은 각 팀이 담당하는 역할과 팀 간 관계를 중점적으로 다루지만, 이 예제의 맥락에서 거버넌스와 의사 결정에 대해서도 설명합니다.

팀은 지원하는 워크로드에 따라 다양한 모델의 여러 부분에서 책임을 맡을 수 있습니다. 설명된 상위 영역보다 더 전문적인 분야로 세분화하기를 원할 수 있습니다. 활동을 분리 또는 합치하거나, 또는 팀을 추가하고 더 구체적인 세부 정보를 제공하면서 이러한 모델을 무한하게 변형시킬 수 있습니다.

여러 팀 간에 중복된 부분을 확인할 수 있거나, 추가적인 이점을 제공 또는 효율성을 높일 수 있지만 아직 알려지지 않은 역량이 있음을 깨달을 수 있습니다. 또한 조직에서 충족되지 않은 요구 사항을 확인 및 해결할 수도 있습니다.

조직의 변화를 평가할 때는 모델 간의 장단점은 무엇인지, 변화 전후의 모델에서 개별 팀은 어떤 역할을 맡는지, 팀의 관계와 책임이 어떻게 바뀌는지, 그리고 변화에 따른 이점이 조직에 영향을 미치는지 여부를 조사합니다.

다음과 같은 네 가지 각각의 운영 모델을 사용하면 성공적으로 운영할 수 있습니다. 일부 모델은 개발 과정의 특정 시점이나 특정 사용 사례에 더 적합할 수 있습니다. 이러한 모델 중 일부는 현재 조직 환경에서 사용하는 모델보다 더 유용할 수 있습니다.

주제

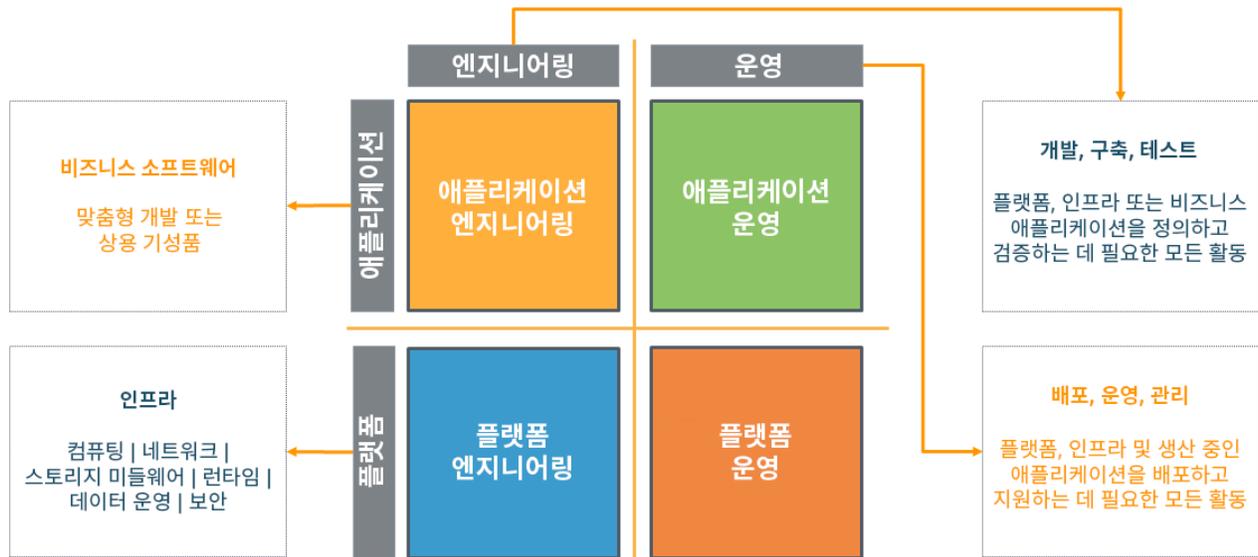
- [완전분리형 운영 모델](#)
- [중앙 집중식 거버넌스와 분리된 애플리케이션 엔지니어링 및 운영\(AEO\) 및 인프라 엔지니어링 및 운영\(IEO\)](#)
- [중앙 집중식 거버넌스 및 서비스 공급자와 분리된 AEO 및 IEO](#)
- [중앙 집중식 거버넌스 및 내부 서비스 공급자 컨설팅 파트너와 분리된 AEO 및 IEO](#)
- [분산형 거버넌스와 분리된 AEO 및 IEO](#)

완전분리형 운영 모델

다음 그림의 세로 축에는 애플리케이션과 인프라가 있습니다. 애플리케이션은 비즈니스 성과를 지원하는 워크로드를 말하며, 맞춤형으로 개발하거나 구매한 소프트웨어일 수 있습니다. 인프라는 물리적 인프라와 가상 인프라, 그리고 해당 워크로드를 지원하는 기타 소프트웨어를 말합니다.

가로 축에는 엔지니어링과 운영이 있습니다. 엔지니어링은 애플리케이션 및 인프라의 개발, 구축 및 테스트를 말합니다. 운영은 애플리케이션 및 인프라의 배포, 업데이트 및 지속적 지원을 말합니다.

기존 모델



많은 조직에 이러한 ‘완전분리형’ 모델이 존재합니다. 각 사분면의 활동은 별도의 팀이 수행합니다. 작업 요청, 작업 대기열 및 티켓과 같은 메커니즘 또는 ITSM(IT 서비스 관리 시스템)을 사용하여 팀 간에 작업이 전달됩니다.

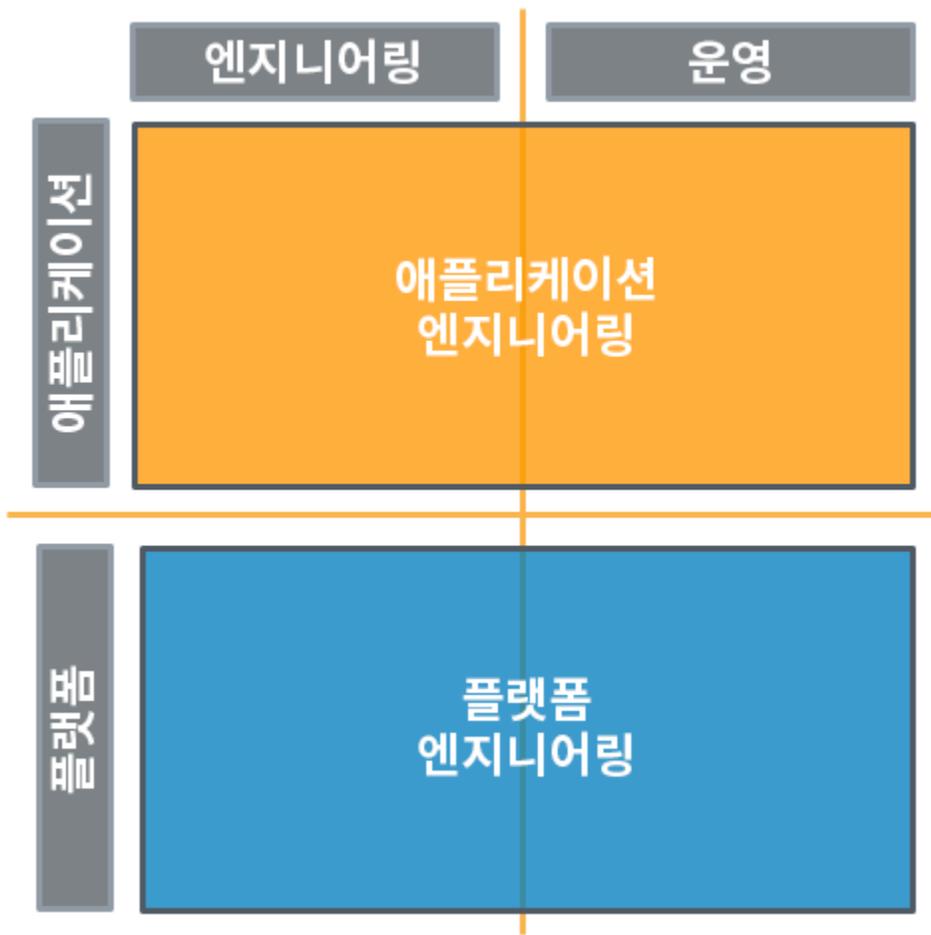
다른 팀으로 또는 여러 팀 간에 작업을 이양하면 복잡해지고 병목 현상과 지연이 발생합니다. 요청은 우선순위가 될 때까지 지연될 수 있습니다. 결함이 뒤늦게 발견되면 상당한 재작업이 필요하며, 동일한 팀과 부서의 검사를 다시 거쳐야 할 수도 있습니다. 엔지니어링 팀의 조치가 필요한 인시던트가 있다면 이미 착수된 활동으로 인해 대응이 지연됩니다.

수행 중인 활동이나 기능을 중심으로 사업, 개발 및 운영 팀을 구성하면 서로 조율되지 않을 위험이 큼니다. 이로 인해 팀이 사업 성과가 아니라, 맡은 책임에만 집중하게 될 수 있습니다. 팀의 전문 분야가 협소하거나, 팀이 물리적으로 또는 논리적으로 격리되어 있으면, 커뮤니케이션과 협업을 저해할 수 있습니다.

중앙 집중식 거버넌스와 분리된 애플리케이션 엔지니어링 및 운영(AEO) 및 인프라 엔지니어링 및 운영(IEO)

이 "분리된 AEO 및 IEO" 모델은 "자체적으로 구축하고 실행"하는 방법론을 따릅니다.

조직의 애플리케이션 엔지니어와 개발자가 워크로드의 운영과 엔지니어링을 모두 수행합니다. 마찬가지로, 인프라 엔지니어는 애플리케이션 팀을 지원하는 데 사용하는 플랫폼의 운영과 엔지니어링을 모두 수행합니다.



이 예제에서는 거버넌스를 중앙 집중식으로 처리하려고 합니다. 표준은 여러 애플리케이션 팀에 제공되거나 배포되거나 공유됩니다.

여러 계정에 걸쳐 환경을 중앙 집중식으로 관리할 수 있는 도구나 서비스를 사용해야 합니다(예: [AWS Organizations](#). 또한 [AWS Control Tower](#) 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.

"자체적으로 구축하고 실행"한다고 해서 애플리케이션 팀이 전체 스택, 도구 체인 및 플랫폼을 책임진다는 의미는 아닙니다.

플랫폼 엔지니어링 팀이 표준화된 서비스 세트(예: 개발 도구, 모니터링 도구, 백업 및 복구 도구, 네트워크)를 애플리케이션 팀에 제공합니다. 플랫폼 팀은 승인된 클라우드 공급자 서비스, 동일한 서비스의 특정 구성 또는 둘 모두에 대한 접근 권한을 애플리케이션 팀에 제공할 수도 있습니다.

승인된 서비스 및 구성을 배포하기 위한 셀프 서비스 기능을 제공하는 메커니즘(예: [Service Catalog](#))을 활용하면, 거버넌스를 적용하면서 이행 요청으로 인한 지연을 최소화할 수 있습니다.

플랫폼 팀은 전체 스택의 가시성을 지원하므로, 이를 통해 애플리케이션 팀은 애플리케이션 구성 요소에서 발생한 문제와 애플리케이션에 사용되는 서비스 및 인프라 구성 요소에서 발생한 문제를 서로 구분할 수 있습니다. 또한 플랫폼 팀은 이러한 서비스 구성을 지원하고 애플리케이션 팀의 운영을 개선하는 방법에 대한 지침을 제공할 수 있습니다.

앞서 언급한 바와 같이, 애플리케이션 팀이 팀의 활동과 애플리케이션 혁신을 지원하는 데 있어서 표준의 추가, 변경 및 예외 처리를 요청할 수 있는 메커니즘이 갖추어져야 합니다.

분리된 AEO IEO 모델은 애플리케이션 팀에 강력한 피드백 루프를 제공합니다. 일상적인 워크로드 운영에서는 지원 및 기능 요청을 통한 직접적 또는 간접적인 상호 작용으로 인해 고객과의 접촉이 늘어나기 마련입니다. 이와 같이 가시성이 높아지는 덕분에 애플리케이션 팀은 문제를 보다 신속하게 해결할 수 있습니다. 더 긴밀해지는 고객의 참여와 고객과의 관계는 고객 요구 사항에 대한 인사이트를 얻고 더 빠르게 혁신할 수 있게 해줍니다.

애플리케이션 팀을 지원하는 플랫폼 팀의 경우도 마찬가지입니다.

채택된 표준은 사전 승인을 받게 되므로, 프로덕션 환경에 적용하는 데 필요한 검토 작업량을 줄일 수 있습니다. 플랫폼 팀에서 제공하는 테스트를 거쳐 지원되는 표준을 사용하면 이러한 서비스 관련 문제의 발생 빈도가 줄어들 수 있습니다. 표준을 채택하면 애플리케이션 팀이 워크로드를 차별화하는 데 집중할 수 있습니다.

중앙 집중식 거버넌스 및 서비스 공급자와 분리된 AEO 및 IEO

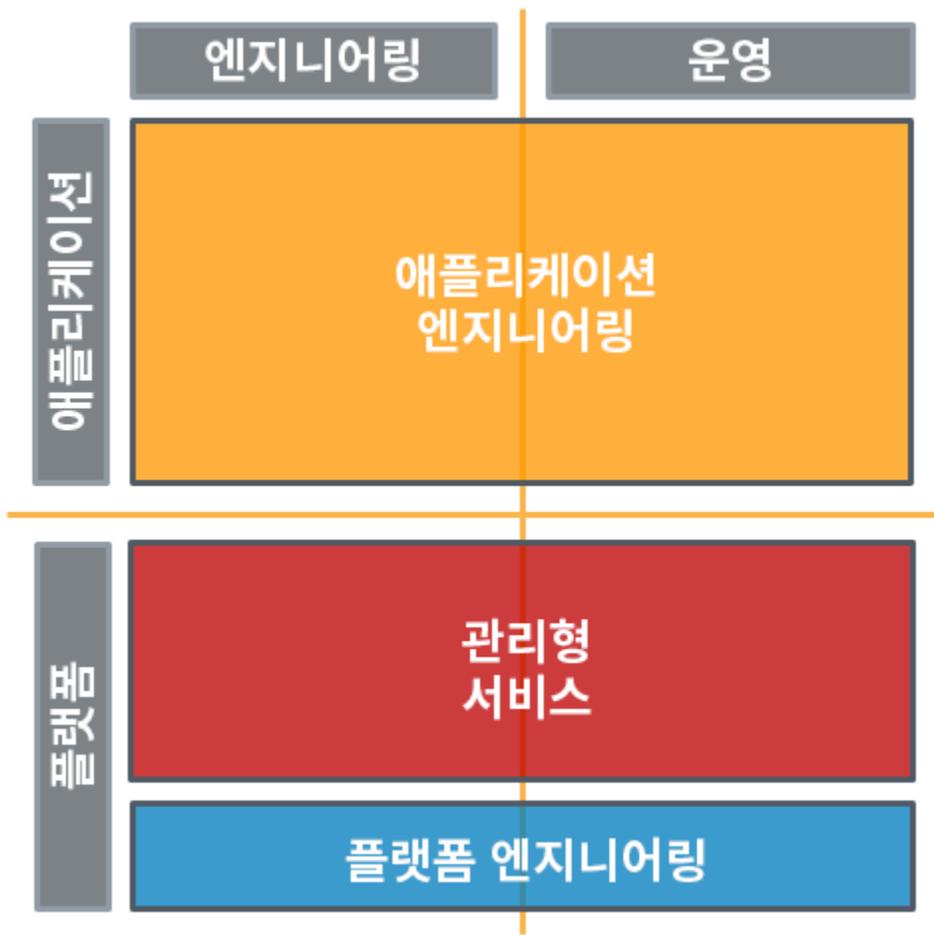
이 "분리된 AEO 및 IEO" 모델은 "자체적으로 구축하고 실행"하는 방법론을 따릅니다.

조직의 애플리케이션 엔지니어와 개발자가 워크로드의 운영과 엔지니어링을 모두 수행합니다.

조직에 전담 플랫폼 엔지니어링 및 운영 팀을 지원할 기술이나 팀원이 갖추어져 있지 않거나, 여기에 시간과 노력을 투자하고 싶지 않을 수도 있습니다.

또는 비즈니스를 차별화하는 기능을 제작하는 데 초점을 맞춘 플랫폼 팀을 두는 동시에 획일적인 일상 업무는 아웃소싱 업체에 맡기고 싶을 수도 있습니다.

관리형 서비스 공급자(예: [AWS Managed Services](#), [AWS Managed Services 파트너](#) 또는 관리형 서비스 공급자([AWS 파트너 네트워크](#)))는 클라우드 환경을 구현하는 전문성을 제공하며, 보안 및 규정 준수 요구 사항과 비즈니스 목표를 지원합니다.



이 모델의 경우 AWS Organizations 및 AWS Control Tower에서 계정 생성과 정책을 관리하고, 플랫폼 팀은 중앙 집중식으로 거버넌스를 관리하도록 구현하려고 합니다.

이 모델에서는 서비스 공급자의 업무에 맞추어 메커니즘을 수정해야 합니다. 이 모델에서는 서비스 공급자를 비롯한 팀 간의 작업 이양으로 인해 발생하는 병목 현상과 지연, 또는 뒤늦은 결함 발견과 관련된 잠재적 재작업 문제가 해결되지 않습니다.

공급자의 표준, 모범 사례, 프로세스 및 전문성을 활용하여 이 문제를 해결할 수 있습니다. 또한 이들이 지속적으로 개발하는 서비스 오퍼링의 이점을 얻을 수 있습니다.

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.

중앙 집중식 거버넌스 및 내부 서비스 공급자 컨설팅 파트너와 분리된 AEO 및 IEO

이 '분리된 AEO 및 IEO' 모델은 '자체적으로 구축하고 실행'하는 방법론을 확립하고자 합니다.

비즈니스에서는 애플리케이션 팀이 워크로드에 대한 엔지니어링 및 운영 활동을 수행하고, 문화와 같이 보다 많은 DevOps를 채택하기를 원합니다.

애플리케이션 팀이 마이그레이션, 클라우드 채택 또는 워크로드 현대화를 진행 중이더라도, 기존에 클라우드 및 클라우드 운영을 적절하게 지원할 기술은 보유하고 있지 않을 수도 있습니다. 애플리케이션 팀의 역량이나 숙련도가 부족하면 노력을 허사로 만드는 장애물이 될 수 있습니다.

이 문제를 해결하려면 질문을 던지고, 요구 사항을 논의하고, 솔루션을 파악할 수 있는 토론의 장이 되어 줄 클라우드 지원 센터 팀(CCoE)을 구성해야 합니다. 조직의 필요에 따라 CCoE는 전담 전문가 팀일 수도 있고, 조직 전체에서 선발한 참여자로 구성된 가상의 팀일 수도 있습니다. CCoE는 팀의 클라우드 전환을 지원하고, 중앙 집중식 클라우드 거버넌스를 수립하며, 계정 및 조직 관리 표준을 정의합니다. 또한 성공적인 참조 아키텍처와 엔터프라이즈용 패턴도 식별합니다.

AWS에서는 CCoE를 일반적인 클라우드 우수성 센터가 아닌 클라우드 지원 센터라고 지칭하며, 지원 팀의 성공과 비즈니스 성과 달성을 뒷받침하는 데 중점을 둡니다.

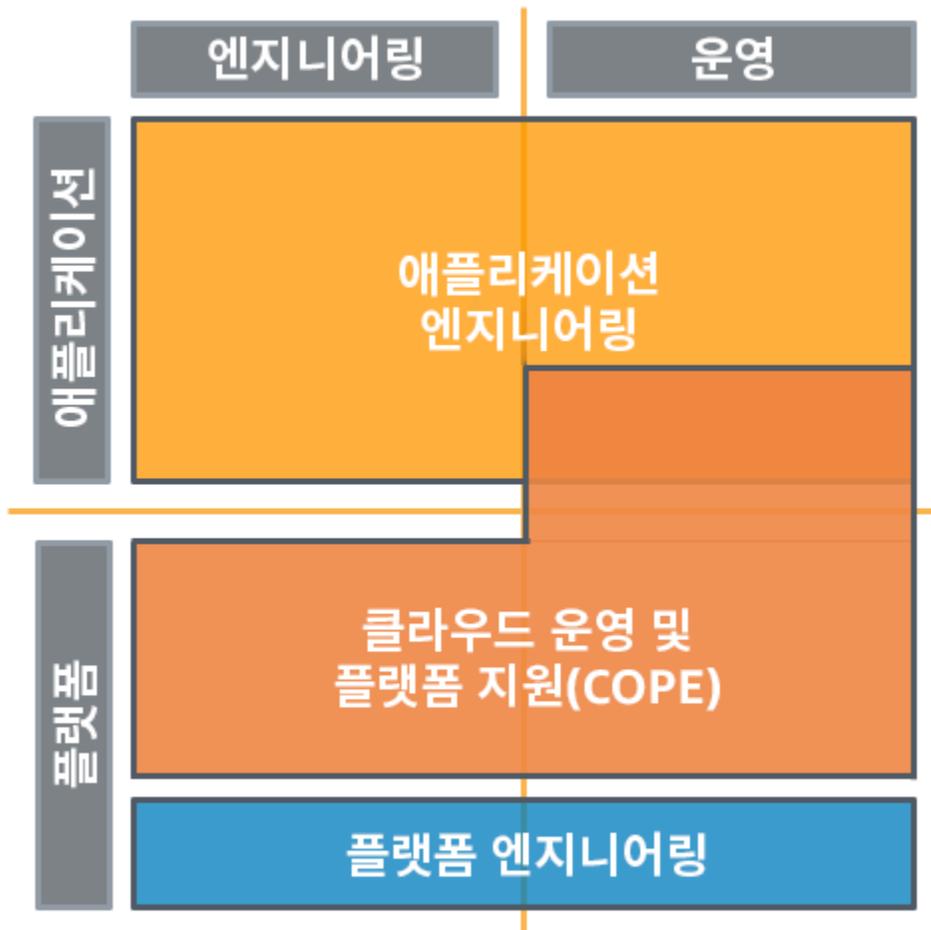
비즈니스의 플랫폼 엔지니어링 팀은 애플리케이션 팀이 채택할 수 있는 표준을 기반으로 핵심 공유 플랫폼 기능을 구축합니다. 또한 셀프 서비스 메커니즘을 통해 애플리케이션 팀에 제공되는 엔터프라이즈 참조 아키텍처와 패턴을 코드화합니다. AWS Service Catalog와 같은 서비스를 사용하면 애플리케이션 팀은 기본적으로 중앙 집중식 거버넌스 및 보안 표준을 준수하는 승인된 참조 아키텍처, 패턴, 서비스 및 구성을 배포할 수 있습니다.

나아가 플랫폼 엔지니어링 팀은 표준화된 서비스 집합(예: 개발 도구, 모니터링 도구, 백업 및 복구 도구, 네트워크)을 애플리케이션 팀에 제공합니다.

조직에는 표준화된 서비스를 관리 및 지원하고 참조 아키텍처와 패턴을 기반으로 클라우드 서비스를 구축하는 애플리케이션 팀을 지원하는 '내부 MSP 및 컨설팅 파트너'가 있습니다. '클라우드 운영 및 플랫폼 지원(COP)' 팀은 애플리케이션 팀과의 협력을 통해 애플리케이션 팀이 기본 운영 원칙을 세우도록 도우며, 시간이 흐르면서 시스템과 리소스에 대한 책임을 더 많이 부담하도록 합니다. COPE 팀은 CCoE 및 플랫폼 엔지니어링 팀과 함께 지속적인 개선을 추진하고 애플리케이션 팀을 지원하는 역할을 맡습니다.

애플리케이션 팀은 환경 조성, CI/CD 파이프라인, 변경 관리, 관찰성 및 모니터링은 물론 필요에 따라 기업의 팀과 통합된 COPE 팀과 함께 인시던트 및 이벤트 관리 프로세스를 정립하는 데 필요한 지원을 받습니다. COPE 팀은 애플리케이션 팀과 함께 이러한 운영 활동을 수행하는 데 참여하며, 시간이 지남에 따라 애플리케이션 팀에 소유권을 넘기며 참여 범위를 단계적으로 줄여나갑니다.

애플리케이션 팀은 COPE 팀으로부터 기술을 익히고 조직으로부터 교훈을 배우며, 중앙 집중식 거버넌스를 통해 구축된 가드레일의 보호를 받습니다. 애플리케이션 팀은 인정받은 성공을 기반으로 하며 채택한 조직 표준을 지속적으로 개발함으로써 이득을 얻습니다. 나아가 관찰성과 모니터링 수립 프로세스를 통해 워크로드의 운영에 대한 인사이트를 얻고, 변경이 워크로드에 미치는 영향을 보다 효과적으로 이해할 수 있습니다.



COPE 팀은 운영 활동을 뒷받침하고, 애플리케이션 팀 전반에서 엔터프라이즈 운영 보기를 제공하며, 중요한 인시던트 관리를 지원하는 데 필요한 액세스 권한을 보유하고 있습니다. COPE 팀은 획일적인 업무 부담으로 여겨지는 활동을 계속해서 담당하는데, 이러한 활동은 대규모로 지원되는 표준 솔루션을 통해 이루어집니다. 또한 애플리케이션 팀이 애플리케이션을 차별화하는 데 집중할 수 있도록 익히 알려진 프로그래밍 방식의 운영 활동과 자동화된 운영 활동을 지속적으로 관리합니다.

비즈니스는 팀의 성공에서 파생된 조직의 표준, 모범 사례, 프로세스 및 전문 지식을 활용할 수 있습니다. 클라우드를 채택하거나 클라우드에서 운영을 현대화하는 새로운 팀을 위해 이러한 성공적인 패턴을 재현할 수 있는 메커니즘을 수립해야 합니다. 이 모델은 애플리케이션 팀의 자립을 돕는 COPE 팀의 역량과 전환 지식, 아티팩트에 초점을 맞춥니다. 따라서 애플리케이션 팀의 운영상 부담을 덜어

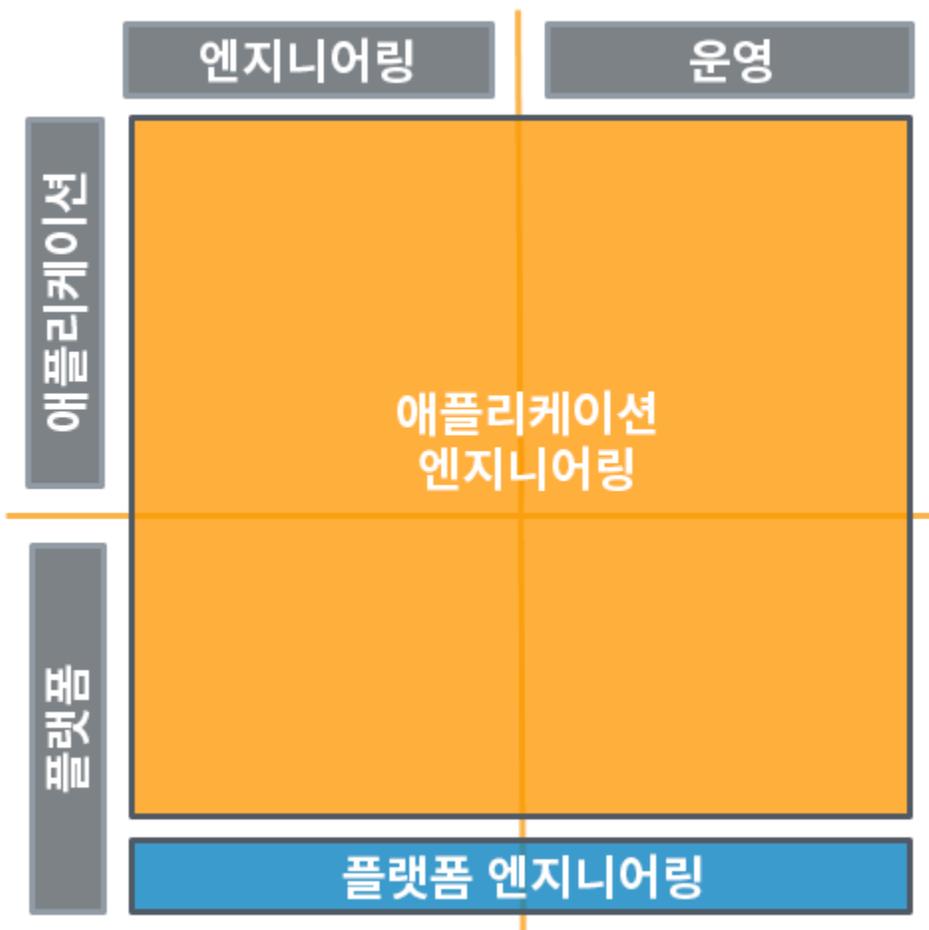
주지만, 동시에 애플리케이션 팀 대부분이 자립하지 못하게 되는 위험을 내포합니다. 이 모델은 또한 CCoE, COPE 및 애플리케이션 팀 간의 관계를 정립하며, 피드백 루프를 형성하여 발전과 혁신을 지원 합니다.

CCoE 및 COPE 팀을 구성하면서 조직 전반의 표준을 정의하면 클라우드 채택을 촉진하고 현대화 작 업을 지원할 수 있습니다. 애플리케이션 팀을 대상으로 컨설턴트 및 파트너 역할을 하는 COPE 팀의 추 가 지원을 통해 애플리케이션 팀의 유익한 클라우드 기능 채택을 가로막는 장벽을 허물 수 있습니다.

분산형 거버넌스와 분리된 AEO 및 IEO

이 "분리된 AEO 및 IEO" 모델은 "자체적으로 구축하고 실행"하는 방법론을 따릅니다.

조직의 애플리케이션 엔지니어와 개발자가 워크로드의 운영과 엔지니어링을 모두 수행합니다. 마찬가지로, 인프라 엔지니어는 애플리케이션 팀을 지원하는 데 사용하는 플랫폼의 운영과 엔지니어링을 모 두 수행합니다.



이 예제에서는 거버넌스를 분산형으로 처리하려고 합니다.

표준은 여전히 플랫폼 팀에 의해 애플리케이션 팀에 배포, 제공 또는 공유되지만, 애플리케이션 팀이 워크로드를 지원하는 데 있어서 새로운 플랫폼 기능을 자유롭게 운영하고 엔지니어링할 수 있습니다.

이 모델에서는 애플리케이션 팀의 제약이 더 적지만, 책임은 더 늘어납니다. 플랫폼 기능을 추가로 지원하려면 추가적인 기술이 필요하고 잠재적으로 팀원이 더 필요할 수도 있습니다. 기술 세트가 적절하지 않고 결함이 조기에 파악되지 않으면 재작업 발생의 위험이 상당히 커집니다.

애플리케이션 팀에 명시적으로 위임되지 않은 정책을 적용해야 합니다. 여러 계정에 걸쳐 환경을 중앙 집중식으로 관리할 수 있는 도구나 서비스를 사용합니다(예: [AWS Organizations](#). 또한 [AWS Control Tower](#) 와 같은 서비스는 계정 설정을 위한 청사진(운영 모델 지원)을 정의하고, AWS Organizations를 통해 지속적으로 거버넌스를 적용하며, 새로운 계정의 프로비저닝을 자동화할 수 있도록 이 관리 기능을 확장합니다.

애플리케이션 팀이 표준에 추가 및 변경을 요청할 수 있는 메커니즘을 갖추는 것이 좋습니다. 애플리케이션 팀이 새로운 표준을 만드는 데에도 기여할 수 있으며, 이는 다른 애플리케이션 팀에 도움이 될 수 있습니다. 플랫폼 팀은 이러한 추가 기능에 대한 직접적인 지원을 제공하는 것이 비즈니스 성과를 효과적으로 지원하는 방법이라고 판단할 수 있습니다.

이 모델은 기술 및 팀원에 대한 요구 사항이 상당히 높으므로 혁신에 있어서 제약이 덜합니다. 팀 간의 작업 이양으로 인해 발생하는 많은 병목 현상과 지연을 해결하는 동시에, 팀과 고객 간 관계 구축을 효과적으로 촉진할 수 있습니다.

관계 및 소유권

운영 모델은 팀 간의 관계를 정의하고 식별 가능한 소유권과 책임을 뒷받침합니다.

모범 사례

- [OPS02-BP01 리소스 소유자 식별](#)
- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)
- [OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별](#)
- [OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보](#)
- [OPS02-BP05 추가, 변경 및 예외를 요청하는 메커니즘](#)
- [OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임](#)

OPS02-BP01 리소스 소유자 식별

워크로드의 리소스에는 변경 제어, 문제 해결 및 기타 기능에 대한 소유자가 식별되어 있어야 합니다. 소유자는 워크로드, 계정, 인프라, 플랫폼 및 애플리케이션에 대해 할당됩니다. 소유권은 중앙 레지스

터 또는 리소스에 첨부된 메타데이터와 같은 도구를 사용하여 기록됩니다. 구성 요소의 비즈니스 가치는 구성 요소에 적용되는 프로세스와 절차를 알려 줍니다.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

고객 사례

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

구현 단계

1. 먼저 조직의 소유권을 정의합니다. 소유권은 리소스에 대한 위험을 부담하거나, 리소스를 변경하거나, 문제 해결 시 리소스를 지원하는 사람을 의미할 수 있습니다. 소유권은 리소스의 재정적 또는 관리적 소유권을 의미할 수도 있습니다.
2. [AWS Organizations](#)를 사용하여 계정을 관리하세요 계정의 대체 연락처를 중앙에서 관리할 수 있습니다.
 - a. 연락처 정보에 회사 소유의 이메일 주소와 전화번호를 사용하면 상급자가 조직을 퇴사한 경우에도 액세스할 수 있습니다. 예를 들어 청구, 운영 및 보안에 대한 별도의 이메일 배포 목록을 생성하고, 각 활성 AWS 계정에서 이를 청구, 보안 및 운영 연락처로 구성합니다. 누군가 휴가 중이거나 역할이 변경되거나 퇴사하더라도 여러 사람이 AWS 알림을 수신하고 응답할 수 있게 됩니다.
 - b. 계정을 [AWS Organizations](#)에서 관리하지 않는 경우 대체 계정 연락처가 AWS에서 필요한 경우 적절한 담당자와 연락할 수 있도록 도와줍니다. 계정의 대체 연락처가 개인이 아닌 그룹으로 설정되도록 구성합니다.
3. 태그를 사용하여 AWS 리소스 소유자를 식별합니다. 소유자와 해당 연락처 정보를 별도의 태그로 지정할 수 있습니다.
 - a. [AWS Config](#) 규칙을 사용하여 리소스에 필요한 소유권 태그가 있는지 확인할 수 있습니다.
 - b. 조직의 태그 지정 전략을 개발하는 방법에 대한 자세한 지침은 [AWS 태그 지정 모범 사례 백서](#)를 참조하세요.
4. [Amazon Q Business](#)는 생성형 AI를 사용하여 기업 시스템의 정보를 기반으로 직원 생산성을 높이고, 질문에 답하고, 작업을 완료하는 대화형 어시스턴트입니다.
 - a. Amazon Q Business를 회사의 데이터 소스에 연결합니다. Amazon Q Business는 Amazon Simple Storage Service(Amazon S3), Microsoft SharePoint, Salesforce 및 Atlassian Confluence를 포함하여 40개 이상의 지원되는 데이터 소스에 대한 사전 구축된 커넥터를 제공합니다. 자세한 내용은 [Amazon Q Business 커넥터](#)를 참조하세요.
5. 다른 리소스, 플랫폼 및 인프라의 경우 소유권을 식별하는 문서를 생성합니다. 모든 팀원이 여기에 액세스할 수 있어야 합니다.

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

리소스

관련 모범 사례:

- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)
- [OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보](#)

관련 문서:

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

관련 워크숍:

- [AWS Workshop - Tagging](#)

관련 예시:

- [AWS Config 규칙 - Amazon EC2 with required tags and valid values](#)

관련 서비스:

- [AWS Config 규칙 - required-tags](#)
- [AWS Organizations](#)

OPS02-BP02 프로세스 및 절차의 소유자 식별

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

- 프로세스 및 절차에서 정의를 담당하는 소유자를 식별했습니다.
 - 워크로드 지원을 위해 수행되는 운영 활동을 식별합니다. 검색 가능한 위치에 이러한 활동을 문서화합니다.
 - 활동 지정을 담당하는 개인 또는 팀을 고유하게 식별합니다. 이들은 올바른 권한, 액세스 및 도구와 적절한 기술을 갖춘 팀원이 활동을 성공적으로 수행할 수 있도록 할 책임이 있습니다. 해당 활동을 수행하는 데 문제가 있는 경우 활동을 수행하는 팀원은 활동 개선에 필요한 상세한 피드백을 제공할 책임이 있습니다.

- AWS Systems Manager 같은 서비스를 통해, 그리고 문서와 AWS Lambda를 통해 활동 아티팩트의 메타데이터에 대한 소유권을 파악합니다. 태그 또는 리소스 그룹을 사용하여 리소스 소유권을 캡처하고 소유권 및 연락처 정보를 지정합니다. AWS Organizations를 사용하여 태그 지정 정책을 생성하고 소유권 및 연락처 정보를 파악합니다.
- 시간이 지남에 따라 이러한 절차를 코드로 실행할 수 있도록 발전시켜 사람이 개입할 필요가 줄어들어야 합니다.
 - AWS Lambda 함수, CloudFormation 템플릿 또는 AWS Systems Manager 자동화 문서를 예로 들어 보겠습니다.
 - 적절한 리포지토리에서 버전 관리를 수행합니다.
 - 소유자와 문서를 쉽게 식별할 수 있도록 적절한 리소스 태그 지정을 포함합니다.

고객 사례

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

구현 단계

1. 기존 프로세스 및 절차를 문서화합니다.
 - a. 이를 검토하고 최신 상태로 유지합니다.
 - b. 각 프로세스 또는 절차의 소유자를 식별합니다.
 - c. 이에 대한 버전을 관리합니다.
 - d. 가능하면 아키텍처 설계를 공유하는 워크로드 및 환경 전반에서 프로세스와 절차를 공유합니다.
2. 피드백 및 개선을 위한 메커니즘을 수립합니다.
 - a. 프로세스를 검토해야 하는 빈도에 대한 정책을 정의합니다.
 - b. 검토자와 승인자를 위한 프로세스를 정의합니다.

- c. 피드백을 제공하고 추적할 수 있도록 문제 또는 티켓팅 대기열을 구현합니다.
 - d. 가능하면 프로세스 및 절차에는 변경 승인 위원회(CAB)의 사전 승인 및 위험 분류가 있어야 합니다.
3. 프로세스와 절차를 실행해야 하는 담당자가 프로세스와 절차에 액세스하고 검색할 수 있는지 확인합니다.
- a. 태그를 사용하여 워크로드의 프로세스 및 절차에 액세스할 수 있는 위치를 지정합니다.
 - b. 의미 있는 오류 및 이벤트 메시지를 사용하여 문제를 해결하기 위한 적절한 프로세스나 절차를 지정합니다.
 - c. Wiki 및 문서 관리를 사용하여 프로세스 및 절차를 조직 전체에서 일관되게 검색할 수 있도록 합니다.
4. 필요한 경우 자동화합니다.
- a. 서비스 및 기술 팀에서 API를 제공할 경우 자동화를 개발해야 합니다.
 - b. 프로세스에 대해 적절하게 교육합니다. 사용자 스토리와 요구 사항을 개발하여 이러한 프로세스를 자동화합니다.
 - c. 문제와 함께 프로세스 및 절차의 사용을 성공적으로 측정하여 지속적인 개선을 지원합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS02-BP01 리소스 소유자 식별](#)
- [OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보](#)
- [OPS11-BP04 지식 관리 수행](#)

관련 문서:

- [AWS 백서 - AWS 기반 DevOps 소개](#)
- [AWS 백서 - Best Practices for Tagging AWS Resources](#)
- [AWS 백서 - 여러 계정을 사용하여 AWS 환경 구성](#)
- [AWS 클라우드 운영 및 마이그레이션 블로그 - 운영 효율성을 위한 클라우드 자동화 사례 구축: AWS Managed Services의 모범 사례](#)

- [AWS 클라우드 Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations](#)
- [AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard](#)
- [AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines](#)

관련 워크숍:

- [AWS Well-Architected Operational Excellence Workshop](#)
- [AWS Workshop - Tagging](#)

관련 동영상:

- [에서 IT 운영을 자동화하는 방법 AWS](#)
- [AWS re:Invent 2020 - Automate anything with AWS Systems Manager](#)
- [AWS re:Inforce 2022 - Automating patch management and compliance using AWS \(NIS306\)](#)
- [AWS Supports You - Diving Deep into AWS Systems Manager](#)

관련 서비스:

- [AWS Systems Manager - Automation](#)
- [AWS Service Management Connector](#)

OPS02-BP03 운영 활동에서 성능을 담당하는 소유자 식별

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 누가 활동 수행을 담당하는지, 조치가 필요한 경우 누구에게 알릴 것인지, 누가 작업을 수행하고 결과를 검증하고 활동 소유자에게 피드백을 제공하는지 이해할 수 있습니다.
- 프로세스와 절차를 통해 워크로드 운영 작업을 강화할 수 있습니다.
- 새로운 팀원들은 더 빠르게 역량을 발휘할 수 있습니다.
- 인시던트를 완화하는 데 걸리는 시간을 줄일 수 있습니다.
- 팀마다 동일한 프로세스와 절차를 사용하여 일관된 방식으로 작업을 수행합니다.
- 팀이 반복 가능한 프로세스를 통해 프로세스를 확장할 수 있습니다.
- 표준화된 프로세스와 절차는 팀 간에 워크로드 책임을 이전하는 데 따르는 영향을 완화하는 데 도움이 됩니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

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

시간이 지남에 따라 이러한 절차를 코드로 실행할 수 있도록 발전시켜 사람이 개입할 필요가 줄어들어야 합니다. 예를 들어 프로시저는 AWS Lambda 함수, AWS CloudFormation 템플릿 또는 AWS Systems Manager 자동화 문서로 시작할 수 있습니다. 이러한 절차가 적절한 리포지토리에서 버전 관

리되는지 확인하고 팀에서 소유자와 문서를 쉽게 식별할 수 있도록 적절한 리소스 태그 지정을 포함하세요. 활동 수행에 대한 책임을 문서화한 다음 성공적인 시작 및 운영과 원하는 결과의 성과를 위한 자동화를 모니터링합니다.

고객 사례

AnyCompany Retail은 소유권을 공통 아키텍처 관행과 기술을 공유하는 애플리케이션 또는 애플리케이션 그룹에 대한 프로세스를 소유하는 팀 또는 개인으로 정의합니다. 처음에 회사는 문서 관리 시스템의 단계별 지침으로 프로세스와 절차를 문서화합니다. AWS 계정을 관리하는 AWS Organizations를 통해, 애플리케이션을 호스팅하는 AWS 계정 태그와 계정 내 특정 리소스 그룹의 태그를 사용하여 프로시저를 검색할 수 있도록 합니다. AnyCompany Retail은 시간이 지남에 따라 이러한 프로세스를 코드로 변환하고 CloudFormation 또는 AWS Cloud Development Kit (AWS CDK) 템플릿과 같은 서비스를 통해 코드형 인프라를 사용하여 리소스를 정의합니다. 운영 프로세스는 AWS Systems Manager 또는 AWS Lambda 기능의 자동화 문서가 되어, Amazon CloudWatch 경보 또는 Amazon EventBridge 이벤트와 같은 이벤트에 대한 대응으로 스케줄링된 작업으로 시작되거나 IT 서비스 관리(ITSM) 플랫폼 내의 요청에 의해 시작될 수 있습니다. 모든 프로세스에는 소유자를 식별하는 태그가 있습니다. 팀은 자동화 및 프로세스에 대한 문서를 프로세스의 코드 저장소에서 생성한 Wiki 페이지 내에서 관리합니다.

구현 단계

1. 기존 프로세스 및 절차를 문서화합니다.
 - a. 최신 상태인지 검토하고 확인합니다.
 - b. 각 프로세스 또는 프로시저에 소유자가 있는지 확인합니다.
 - c. 프로시저의 버전을 관리하세요.
 - d. 가능하면 아키텍처 설계를 공유하는 워크로드 및 환경 전반에서 프로세스와 절차를 공유합니다.
2. 피드백 및 개선을 위한 메커니즘을 수립합니다.
 - a. 프로세스를 검토해야 하는 빈도에 대한 정책을 정의합니다.
 - b. 검토자와 승인자를 위한 프로세스를 정의합니다.
 - c. 이슈 또는 티켓팅 대기열을 구현하여 피드백을 제공하고 추적합니다.
 - d. 가능하면 프로세스 및 절차에는 변경 승인 위원회(CAB)의 사전 승인 및 위험 분류가 있어야 합니다.
3. 프로세스 및 프로시저를 실행해야 하는 사용자가 이를 액세스하고 검색할 수 있도록 합니다.
 - a. 태그를 사용하여 워크로드의 프로세스 및 절차에 액세스할 수 있는 위치를 지정합니다.
 - b. 의미 있는 오류 및 이벤트 메시지를 사용하여 문제를 해결하기 위한 적절한 프로세스나 절차를 지정합니다.

- c. Wiki 또는 문서 관리를 사용하여 조직 전체에서 프로세스와 절차를 일관되게 검색할 수 있습니다.
4. 필요할 때 자동화하세요.
- a. 서비스와 기술이 API를 제공하는 경우 자동화를 개발합니다.
 - b. 프로세스를 제대로 이해하고 있는지 확인하고 해당 프로세스를 자동화하기 위한 사용자 스토리와 요구 사항을 개발합니다.
 - c. 반복적 개선을 지원하는 문제 추적을 통해 프로세스 및 절차의 성공적인 사용을 측정합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS02-BP01 리소스 소유자 식별](#)
- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)
- [OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보](#)
- [OPS02-BP05 책임과 소유권을 식별하는 메커니즘](#)
- [OPS11-BP04 지식 관리 수행](#)

관련 문서:

- [AWS 백서 | AWS 기반 DevOps 소개](#)
- [AWS 백서 | Best Practices for Tagging AWS Resources](#)
- [AWS 백서 | 여러 계정을 사용하여 AWS 환경 구성](#)
- [AWS 클라우드 운영 및 마이그레이션 블로그: 운영 효율성을 위한 클라우드 자동화 사례 구축: AWS Managed Services의 모범 사례](#)
- [AWS Workshop - Tagging](#)
- [AWS Service Management Connector](#)

관련 동영상:

- [AWS Knowledge Center Live | Tagging AWS Resources](#)
- [AWS re:Invent 2020 | Automate anything with AWS Systems Manager](#)

- [AWS re:Inforce 2022 | Automating patch management and compliance using AWS \(NIS306\)](#)
- [AWS Supports You | Diving Deep into AWS Systems Manager](#)

관련 예시:

- [AWS Well-Architected Operational Excellence Workshop](#)

OPS02-BP04 책임과 소유권을 관리하는 메커니즘 확보

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

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

원하는 결과: 조직 내 팀에는 리소스와 관련된 방식, 수행할 작업, 프로세스 및 절차를 포함하여 명확하게 정의된 책임이 있습니다. 이러한 책임은 팀의 책임과 목표는 물론 다른 팀의 책임과도 일치합니다. 일관되고 검색 가능한 방식으로 에스컬레이션 경로를 문서화하고 이러한 결정을 책임 매트릭스, 팀 정의 또는 Wiki 페이지와 같은 문서 아티팩트에 입력합니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

팀원의 역할과 책임을 식별하고 맡은 역할에 대한 기대치를 이해하도록 합니다. 조직의 구성원이 특정 요구 사항에 대해 연락할 팀이나 개인을 식별할 수 있도록 이 정보를 검색할 수 있게 설정합니다. 조직이 AWS에서 마이그레이션 및 현대화 기회를 활용하고자 함에 따라 역할과 책임도 달라질 수 있습니다. 팀과 구성원이 각자의 책임을 인식하도록 하고 이러한 변경 기간 동안 작업을 수행할 수 있도록 적절하게 교육하세요.

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

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

고객 사례

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

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

구현 단계

1. 기존 문서부터 시작하세요. 일반적인 소스 문서에는 다음이 포함될 수 있습니다.
 - a. 책임 또는 Responsible, Accountable, Consulted, and Informed(RACI) 매트릭스
 - b. 팀 정의 또는 Wiki 페이지
 - c. 서비스 정의 및 오퍼링
 - d. 역할 또는 직무 설명
2. 문서화된 책임을 검토하고 토론을 주최하세요.
 - a. 팀과 함께 검토하여 문서화된 책임과 팀이 일반적으로 수행하는 책임 간의 불일치 사항을 파악하세요.
 - b. 내부 고객이 제공하는 잠재적 서비스에 대해 논의하여 팀 간의 기대치 차이를 파악하세요.
3. 불일치를 분석하고 해결합니다.
4. 몇 가지 개선 기회를 파악합니다.
 - a. 자주 요구되고 리소스를 많이 사용하는 요청(일반적으로 강력한 개선 대상)을 식별합니다.
 - b. 모범 사례, 패턴, 권장 가이드를 찾아보고 이를 통해 개선을 간소화하고 표준화합니다.
 - c. 개선 기회를 기록하고 완료될 때까지 추적합니다.
5. 팀에 아직 책임 할당을 관리하고 추적할 책임이 없다면 팀에서 이 책임을 맡을 사람을 정합니다.
6. 팀의 책임 명시 요청 프로세스를 정의합니다.
 - a. 프로세스를 검토하고 명확하고 사용하기 쉬운지 확인합니다.
 - b. 누군가가 에스컬레이션을 담당하고 추적하여 결론에 도달하도록 하세요.
 - c. 운영 지표를 수립하여 효과를 측정합니다.
 - d. 피드백 메커니즘을 만들어 팀이 개선 기회를 강조할 수 있는지 확인하세요.
 - e. 주기적 검토를 위한 메커니즘을 구현합니다.
7. 검색 가능하고 액세스 가능한 위치에 문서화하세요.
 - a. 일반적으로는 Wiki 또는 문서 포털을 사용합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

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

관련 문서:

- [AWS 백서 - AWS 기반 DevOps 소개](#)
- [AWS 백서 - AWS 클라우드 Adoption Framework: 운영 관점](#)
- [AWS Well-Architected Framework Operational Excellence - Workload level Operating model topologies](#)
- [AWS Prescriptive Guidance - Building your Cloud Operating Model](#)
- [AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model](#)
- [AWS 클라우드 Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams](#)
- [AWS 클라우드 Operations & Migrations Blog - Why a Cloud Operating Model?](#)
- [AWS DevOps Blog - How organizations are modernizing for cloud operations](#)

관련 동영상:

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

OPS02-BP05 추가, 변경 및 예외를 요청하는 메커니즘

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

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

관련 문서:

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

관련 서비스:

- [AWS Systems Manager Change Manager](#)

OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임

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

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

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

관련 문서:

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

조직 문화

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

모범 사례

- [OPS03-BP01 경영진의 후원 제공](#)

- [OPS03-BP02 팀원에게 성과 달성이 위태로울 때 조치를 취할 수 있는 권한 부여](#)
- [OPS03-BP03 에스컬레이션 장려](#)
- [OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션](#)
- [OPS03-BP05 실험 장려](#)
- [OPS03-BP06 팀원의 기술 역량 유지와 강화 장려](#)
- [OPS03-BP07 팀에 적절한 리소스 제공](#)

OPS03-BP01 경영진의 후원 제공

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

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

일반적인 안티 패턴:

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

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

고객 사례

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

구현 단계

1. 단일 스레드 리더십을 구축하고 혁신을 주도하고 추진할 주 경영진 후원자를 지정합니다.
2. 혁신의 명확한 비즈니스 성과를 정의하고 소유권과 책임을 지정합니다. 주요 경영진에 중요한 결정을 이끌고 결론지을 수 있는 권한을 부여합니다.
3. 혁신 전략이 매우 명확하고 경영진 후원자가 조직의 모든 수준에 이를 폭넓게 전달하도록 확인합니다.
 - a. IT 및 클라우드 이니셔티브에 대해 명확하게 정의된 비즈니스 목표를 설정합니다.
 - b. IT 및 클라우드 혁신을 추진하기 위한 주요 비즈니스 지표를 문서화합니다.

- c. 전략의 일부를 담당하는 모든 팀과 개인에게 비전을 일관되게 전달합니다.
4. 특정 리더, 관리자 및 개별 기여자에게 전달해야 하는 메시지를 지정하는 커뮤니케이션 계획 매트릭스를 개발합니다. 이 메시지를 전달해야 하는 담당자나 팀을 지정합니다.
 - a. 커뮤니케이션 계획을 일관되고 안정적으로 이행합니다.
 - b. 정기적으로 대면 이벤트를 통해 기대치를 설정하고 관리합니다.
 - c. 커뮤니케이션의 효과에 대한 피드백을 수용하고 그에 따라 커뮤니케이션을 조정하고 계획을 세웁니다.
 - d. 커뮤니케이션 이벤트를 예약하여 팀의 문제를 사전에 파악하고 필요한 경우 과정을 수정할 수 있는 일관된 피드백 루프를 구축합니다.
5. 리더십 관점에서 각 이니셔티브에 적극적으로 참여하여 영향을 받는 모든 팀이 달성해야 할 성과를 이해하고 있는지 확인합니다.
6. 모든 현황 회의에서 경영진 후원자는 방해 요소를 찾고, 확립된 지표, 사례 또는 팀의 피드백을 검토하고, 목표를 향한 진행 상황을 측정해야 합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션](#)
- [OP11-BP01 지속적인 개선을 위한 프로세스 마련](#)
- [OPS11-BP07 운영 지표 검토 수행](#)

관련 문서:

- [Untangling Your Organisational Hairball: Highly Aligned](#)
- [The Living Transformation: Pragmatically approaching changes](#)
- [Becoming a Future-Ready Enterprise](#)
- [CCOE 구축 시 피해야 할 7가지 위험](#)
- [Navigating the Cloud: Key Performance Indicators for Success](#)

관련 동영상:

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

관련 예시:

- [Prosci: Primary Sponsor's Role & Importance](#)

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

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

예를 들어, Amazon은 [리더십 원칙](#)을 지침으로 삼아 직원들이 상황에 맞게 앞으로 나아가고, 문제를 해결하며, 갈등을 해결하고, 조치를 취할 수 있도록 바람직한 행동을 유도합니다.

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

일반적인 안티 패턴:

- 조직 내에는 위험이 식별되었을 때 어떤 조치를 취해야 하는지에 대한 명확한 지침이나 메커니즘이 없습니다. 예를 들어 피싱 공격을 감지한 직원은 보안팀에 보고하지 않아 조직의 상당 부분이 공격을 받게 됩니다. 이로 인해 데이터 침해가 발생합니다.
- 고객은 주로 배포 실패로 인한 서비스 가용성 장애에 대해 불평합니다. SRE 팀이 배포 도구를 담당하며 배포에 대한 자동 롤백은 장기 로드맵에 포함되어 있습니다. 최근 애플리케이션 출시에서 엔지니어 중 한 명이 애플리케이션을 이전 버전으로 자동으로 롤백하는 솔루션을 고안했습니다. 이들의 솔루션이 SRE 팀의 패턴이 될 수 있지만, 다른 팀은 이러한 개선 사항을 추적할 프로세스가 없기 때문에 이를 채택하지 않게 됩니다. 조직은 고객에게 영향을 미치고 부정적인 분위기를 야기하는 배포 실패로 인해 계속 어려움을 겪습니다.
- 규정을 준수하기 위해 Infosec 팀은 Amazon EC2 Linux 인스턴스에 연결하는 운영자를 대신하여 공유 SSH 키를 정기적으로 교체하기 위해 오랫동안 확립된 프로세스를 감독합니다. Infosec 팀이 키

교체를 완료하는 데 수일이 걸리며 해당 인스턴스에 연결할 수 없게 됩니다. Infosec 내부 또는 외부의 어느 누구도 동일한 결과를 얻기 위해 AWS에서 다른 옵션을 사용할 것을 제안하지 않습니다.

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS01-BP06 장점과 위험을 관리하면서 절충점 평가](#)
- [OPS02-BP05 책임과 소유권을 식별하는 메커니즘](#)

관련 문서:

- [AWS Blog Post | The agile enterprise](#)
- [AWS Blog Post | Measuring success: A paradox and a plan](#)
- [AWS Blog Post | Letting go: Enabling autonomy in teams](#)
- [Centralize or Decentralize?](#)

관련 동영상:

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

관련 예시:

- [Using architectural decision records to streamline technical decision-making for a software development project](#)

OPS03-BP03 에스컬레이션 장려

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

원하는 결과: 조직 전체적으로 개인은 문제를 직속 리더십에 에스컬레이션하는 데 익숙합니다. 경영진은 의도적, 의식적으로 팀이 문제를 에스컬레이션해도 괜찮다고 생각해야 한다는 기대를 설정했습니다. 조직 내 각 수준에 문제를 에스컬레이션하는 메커니즘이 있습니다. 직원이 관리자에게 에스컬레이션하면 영향 수준과 문제를 에스컬레이션할지 여부를 공동으로 결정합니다. 에스컬레이션을 시작하려

면 직원은 문제 해결을 위한 권장 작업 계획을 포함해야 합니다. 직속 경영진이 적시에 조치를 취하지 않을 경우, 직원들이 조직에 대한 위협으로 인해 에스컬레이션이 정당하다고 느낄 경우 최고 수준의 경영진에 문제를 제기하는 것이 좋습니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

구현 단계

1. 조직에 대한 정책, 표준 및 기대치를 정의합니다.

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

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례

OPS03-BP03 에스컬레이션 장려

- [OPS02-BP05 추가, 변경 및 예외를 요청하는 메커니즘](#)

관련 문서:

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

관련 동영상:

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

관련 예시:

- [Working with escalation plans in Incident Manager](#)

OPS03-BP04 시기 적절하고 명확하며 실행 가능한 커뮤니케이션

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

원하는 결과: 조직은 변화가 조직에 미치는 영향을 다루기 위한 커뮤니케이션 전략을 설계합니다. 팀들은 서로 적대적으로 일하기보다는 계속해서 서로 협력할 수 있도록 계속 정보를 얻고 동기를 부여받습니다. 개인은 명시된 목표를 달성하는 데 자신의 역할이 얼마나 중요한지 이해합니다. 이메일은 커뮤니케이션을 위한 수동적인 메커니즘일 뿐이며 적절하게 사용됩니다. 경영진은 개별 기여자와 시간을 보내어 이들이 자신의 책임, 완료해야 할 업무, 업무가 전체 사명에 기여하는 바를 이해하도록 돕습니다. 필요한 경우 리더는 규모가 작은 장소에 사람들을 직접 불러 메시지를 전달하고 이러한 메시지가 효과적으로 전달되고 있는지 확인합니다. 효과적인 커뮤니케이션 전략의 결과로 조직은 리더가 기대하는

수준 이상의 성과를 거둘 수 있습니다. 리더는 팀 내부 및 여러 팀에 걸쳐 다양한 의견을 내도록 장려하고 다양한 의견을 구합니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

이 모범 사례를 구현하려면 조직 전체의 이해관계자와 협력하여 통신 표준에 동의해야 합니다. 이러한 표준을 조직에 공개적으로 알리세요. 중요한 IT 전환이 발생할 경우, 계획 팀이 구성되어 있다면 이러한 관행을 무시하는 조직보다 변화가 직원에게 미치는 영향을 더 성공적으로 관리할 수 있습니다. 대규모 조직은 변화를 관리하기가 더 어려울 수 있습니다. 새로운 전략에 대해 모든 개별 기여자로부터 강

력한 동의를 얻는 것이 중요하기 때문입니다. 이러한 전환 계획 팀이 없는 경우 효과적인 커뮤니케이션에 대한 책임은 전적으로 리더에게 있습니다. 전환 계획 팀을 구성할 때는 모든 조직 리더와 협력하여 모든 수준에서 효과적인 커뮤니케이션을 정의하고 관리할 팀을 지정하세요.

고객 사례

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

구현 단계

1. 조직 내의 여러 수준에서 발생하는 변화에 대한 커뮤니케이션 계획을 수립하고 시작할 책임이 있는 핵심 팀을 조직 내에 구성합니다.
2. 단일 스레드 소유권을 도입하여 감독을 확보합니다. 독립적으로 혁신할 수 있는 역량을 개별 팀에 부여하고 독립성과 메커니즘의 일관된 사용 간에 균형을 유지합니다. 이렇게 하면 검사와 방향성을 적절한 수준으로 유지할 수 있습니다.
3. 조직 전체의 이해관계자와 협력하여 커뮤니케이션 표준, 관행 및 계획에 대해 합의를 이룹니다.
4. 핵심 커뮤니케이션 팀이 조직 및 프로그램 리더와 협력하여 리더를 대신해 적절한 직원에게 보낼 메시지를 작성하는지 확인합니다.
5. 팀원이 취해야 할 조치에 대해 적절한 기대치를 갖도록 공지, 일정 공유, 전사 회의, 대면 또는 일대일 방식을 통해 변화를 관리하는 전략적 커뮤니케이션 메커니즘을 구축합니다.
6. 조치가 필요한지 판단하는 데 필요한 맥락, 세부 정보 및 가능한 경우 시간을 안내합니다. 조치가 필요한 경우 필요한 조치와 그 영향을 알립니다.
7. 내부 채팅, 이메일, 지식 관리와 같은 전술적 커뮤니케이션을 촉진하는 도구를 도입합니다.
8. 모든 커뮤니케이션이 원하는 결과로 이어지는지 측정하고 검증하는 메커니즘을 구현합니다.
9. 모든 커뮤니케이션의 효과를 측정하는 피드백 루프를 구축합니다. 특히 커뮤니케이션이 조직 전반에서 변화에 대한 저항과 관련된 경우 피드백 루프가 더 중요합니다.
10. 모든 AWS 계정에서 청구, 보안 및 운영을 위한 [대체 연락처](#)를 설정합니다. 각 연락처를 개인의 연락처가 아닌 이메일 목록으로 설정하는 것이 가장 좋습니다.
11. 에스컬레이션 및 역에스컬레이션 커뮤니케이션 계획을 수립하여 내부 팀 및 AWS 지원 팀과 기타 타사 공급업체를 포함한 외부 팀과 협력합니다.
12. 각 혁신 프로그램의 전체 기간 동안 일관되게 커뮤니케이션 전략을 시작하고 수행합니다.
13. 가능한 경우 반복 가능한 작업에 높은 우선순위를 지정하여 대규모로 안전하게 자동화합니다.

14. 자동화된 작업이 포함된 시나리오에서 커뮤니케이션이 필요한 경우 커뮤니케이션은 팀에 정보 제공 또는 감사를 위한 것이거나 변경 관리 프로세스의 일부여야 합니다.
15. 알림 시스템의 커뮤니케이션을 분석하여 오탐이나 지속적으로 생성되는 알림이 있는지 확인합니다. 사람의 개입이 필요할 때 시작되도록 이러한 알림을 제거하거나 변경합니다. 알림이 시작되면 런북 또는 플레이북을 제공합니다.
- a. [AWS Systems Manager Documents](#)를 사용하여 알림용 플레이북과 런북을 작성할 수 있습니다.
16. 적절한 대응을 가능하게 하는 충분한 정보와 함께 명확하고 실행 가능한 방식으로 위험 또는 계획된 이벤트를 알리는 메커니즘이 마련되어 있습니다. 이메일 목록 또는 채팅 채널을 사용하여 계획된 이벤트 전에 알림을 보냅니다.
- a. [AWS Chatbot](#)은 조직 메시징 플랫폼에서 알림을 보내고 이벤트에 응답하는 데 사용할 수 있습니다.
17. 계획된 이벤트를 검색할 수 있는 액세스 가능한 정보 출처를 제공합니다. 동일한 시스템의 계획된 이벤트에 대한 알림을 제공합니다.
- a. [AWS Systems Manager Change Calendar](#)는 변경이 발생 가능한 경우 변경 기간을 설정하는 데 사용할 수 있습니다. 이 도구는 팀원에게 안전하게 변경할 수 있는 시점을 알려줍니다.
18. 취약성 알림과 패치 정보를 모니터링하여 워크로드 구성 요소와 관련된 잠재적 위험 및 취약성을 파악합니다. 팀원에게 조치를 취할 수 있도록 알림을 제공합니다.
- a. [AWS 보안 공지](#)를 구독하여 AWS의 취약성 알림을 받아볼 수 있습니다.
19. 다양한 의견과 관점 모색: 모든 사람이 기여하도록 장려합니다. 소외된 그룹에 커뮤니케이션의 기회를 제공합니다. 회의에서 역할과 책임을 교대로 말합니다.
- a. 역할 및 책임 확대: 다른 상황에서는 말할 수 없는 역할을 맡을 기회를 팀원에게 제공합니다. 팀원은 다른 상황에서는 불가능할 수 있는 새로운 팀원과의 상호 작용 및 역할에서 경험과 관점을 얻습니다. 또한 자신의 경험과 관점을 바탕으로 새로 맡은 역할을 수행하고 새로운 팀원과 상호 작용합니다. 관점이 발전함에 따라 새로 나타나는 비즈니스 기회나 새로운 개선 기회를 찾습니다. 팀 내에서 대개 다른 사람들이 수행하는 일반적인 업무를 팀원들이 번갈아 맡도록 하여 해당 업무 수행의 요구 사항과 영향을 이해하도록 합니다.
- b. 안전하고 환영받는 환경 제공: 조직 내 팀원의 정신적, 신체적 안전을 보호하는 정책과 규제 수단을 마련합니다. 팀원은 보복에 대한 두려움 없이 상호 작용할 수 있어야 합니다. 팀원들이 안전하고 환영 받는다고 느낄 때 참여와 생산성이 향상됩니다. 조직의 다양성이 높을수록 고객을 비롯하여 지원하는 인력을 더 잘 이해할 수 있습니다. 팀원들이 편하고 자유롭게 이야기할 수 있고 자신의 의견이 존중된다고 확신할 때 마케팅 기회, 접근성 요구 사항, 소외된 시장 부문, 환경에서 알려지지 않은 위험과 같은 귀중한 인사이트를 공유할 가능성이 더 커집니다.
- c. 팀원의 완전한 참여 장려: 직원이 모든 업무 관련 활동에 완전히 참여하는 데 필요한 리소스를 제공합니다. 일상적인 문제에 직면하는 팀원들은 이러한 문제를 해결할 수 있는 기술 역량을 개발

합니다. 고유하게 개발된 이러한 기술 역량은 조직에 상당한 이점을 가져올 수 있습니다. 팀원들에게 필요한 편의를 제공하면 그들의 조력을 통해 얻을 수 있는 혜택이 늘어납니다.

리소스

관련 모범 사례:

- [OPS03-BP01 경영진의 후원 제공](#)
- [OPS07-BP03 런북을 사용한 절차 수행](#)
- [OPS07-BP04 플레이북을 사용하여 문제 조사](#)

관련 문서:

- [AWS 블로그 게시물 | Accountability and empowerment are key to high-performing agile organizations](#)
- [AWS Executive Insights | 복잡성이 아닌 혁신을 확대하는 방법 배우기 | 단일 스레드 리더](#)
- [AWS 보안 공지](#)
- [Open CVE](#)
- [Slack 기반 AWS Support 앱 출시 - 서포트 케이스 생성 및 관리 기능](#)
- [Manage AWS resources in your Slack channels with AWS Chatbot](#)

관련 예시:

- [Well-Architected 실습: 인벤토리 및 패치 관리\(레벨 100\)](#)

관련 서비스:

- [AWS Chatbot](#)
- [AWS Systems Manager Change Calendar](#)
- [AWS Systems Manager Documents](#)

OPS03-BP05 실험 장려

실험은 새로운 아이디어를 제품과 기능으로 탈바꿈하는 촉매제입니다. 실험은 학습을 가속화하고 팀원의 관심과 참여를 유지합니다. 팀원은 혁신을 추진하기 위해 자주 실험하도록 장려됩니다. 원하지 않

는 결과가 나오더라도 하지 말아야 할 것을 알았다는 사실만으로 실험은 가치가 있습니다. 원치 않는 결과가 나온 성공한 실험에 대해 팀원에게 불이익을 가하지 않습니다.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

고객 사례

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

구현 단계

1. 조직 전체에서 경영진과 협력하여 실험을 지원합니다. 팀원이 안전한 방법으로 실험하도록 장려해야 합니다.

2. 팀원이 안전하게 실험할 수 있는 환경을 제공합니다. 프로덕션 환경과 같은 환경에 액세스할 수 있어야 합니다.
 - a. 별도의 AWS 계정을 사용하여 실험용 샌드박스 환경을 생성할 수 있습니다. [AWS Control Tower](#)를 사용하여 이러한 계정을 프로비저닝할 수 있습니다.
3. 기능 플래그 및 A/B 테스트를 사용하여 안전하게 실험하고 사용자 피드백을 수집합니다.
 - a. [AWS AppConfig 기능 플래그](#)는 기능 플래그를 만드는 기능을 제공합니다.
 - b. [Amazon CloudWatch Evidently](#)는 제한된 배포 환경에서 A/B 테스트를 실행하는 데 사용할 수 있습니다.
 - c. [AWS Lambda 버전](#)을 사용하여 베타 테스트를 위해 새로운 기능 버전을 배포할 수 있습니다.

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

리소스

관련 모범 사례:

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

관련 문서:

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

관련 동영상:

- [AWS On Air ft. Amazon CloudWatch Evidently | AWS Events](#) (AWS On Air - Amazon CloudWatch Evidently 소개 | AWS Events)
- [AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira](#) (AWS On Air San Fran Summit 2022 - Jira와 AWS AppConfig 기능 플래그 통합)
- [AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags \(BOA305-R\)](#)(AWS re:Invent 2022 - 배포는 릴리스가 아님: 기능 플래그를 사용한 출시 제어 (BOA305-R))
- [Programmatically Create an AWS 계정 with AWS Control Tower](#)(AWS Control Tower를 통해 프로그래밍 방식으로 AWS 계정 생성)
- [AWS Organizations의 모범 사례를 사용하는 다중 계정 AWS 환경 설정](#)

관련 예시:

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

관련 서비스:

- [Amazon CloudWatch Evidently](#)
- [AWS AppConfig](#)
- [AWS Control Tower](#)

OPS03-BP06 팀원의 기술 역량 유지와 강화 장려

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

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

[AWS Support](#), ([AWS re:Post](#), [AWS Support Center](#)) 및 [AWS 설명서](#)와 같은 리소스는 기술적 장애물을 제거하고 운영을 개선하는 데 도움이 됩니다. 문의 사항에 대해 지원을 받으려면 AWS Support 센터를 통해 AWS Support에 지원을 요청하세요.

AWS는 [Amazon Builders' Library](#)에서 AWS의 운영을 통해 학습한 모범 사례와 패턴을 공유하며, [AWS 블로그](#) 및 [The Official AWS Podcast](#)를 통해 도움이 되는 방대한 기타 교육 자료를 제공합니다.

[AWS 교육 and Certification](#)에서는 역할 또는 영역별 학습 계획과 함께 자습형 디지털 과정을 통한 무료 교육이 제공됩니다. 강사 주도형 교육에 등록하여 팀이 AWS 기술을 연마하도록 추가로 지원할 수도 있습니다.

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

일반적인 안티 패턴:

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

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

구현 단계

1. 구조화된 클라우드 옹호 프로그램 사용: [AWS Skills Guild](#)는 클라우드 기술 역량에 대한 자신감을 높이고 지속적인 학습의 문화를 활성화하기 위한 컨설팅 교육을 제공합니다.
2. 교육 리소스 제공: 교육 자료와 실습 리소스를 이용할 수 있는 체계적인 시간과 액세스를 전용으로 제공하고 교사와 동료에게서 배울 수 있는 기회를 제공하는 컨퍼런스 및 전문 조직에 참석할 수 있도록 지원합니다. 주니어 팀원들이 시니어 팀원들을 멘토로 만나게 하거나, 시니어가 일할 때 주니어 팀원들이 따라다니며 업무 방식과 기술 역량을 접할 수 있게 합니다. 보다 넓은 시야를 확보하기 위해 작업과 직접적으로 관련되지 않은 콘텐츠에 대해 학습하도록 장려합니다.
3. 전문 기술 리소스 활용 장려: [AWS re:Post](#)와 같은 리소스를 활용하여 선별된 지식을 얻고 활발한 커뮤니티에 참여합니다.
4. 최신 지식 리포지토리 구축 및 유지 관리: Wiki 및 런북과 같은 지식 공유 플랫폼을 사용합니다. [AWS re:Post Private](#)을 통해 재사용 가능한 전문 지식 소스를 만들어 협업을 간소화하고 생산성을 개선하며 직원 온보딩을 가속화합니다.
5. 팀 교육 및 팀 간 참여: 팀원의 지속적인 교육 요구 사항에 대비하여 계획을 세웁니다. 팀원이 다른 팀(임시로 또는 영구적으로)에 합류하여 전체 조직에 도움이 되는 기술과 모범 사례를 공유할 수 있는 기회를 제공합니다.
6. 업계 자격증 획득 및 유지 지원: 팀원이 배운 내용을 검증하고 그 성과를 인정하는 업계 자격증을 획득하고 유지하도록 지원합니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- [OPS03-BP01 경영진의 후원 제공](#)
- [OPS11-BP04 지식 관리 수행](#)

관련 문서:

- [AWS 백서 | Cloud Adoption Framework: People Perspective](#)
- [Investing in continuous learning to grow your organization's future](#)
- [AWS Skills Guild](#)
- [AWS 교육 and Certification](#)
- [AWS Support](#)
- [AWS re:Post](#)

- [AWS 시작하기 리소스 센터](#)
- [AWS 블로그](#)
- [AWS 클라우드 규정 준수](#)
- [AWS 설명서](#)
- [The Official AWS Podcast.](#)
- [AWS Online Tech Talks](#)
- [AWS 이벤트 및 웨비나](#)
- [AWS Well-Architected Labs](#)
- [Amazon Builders' Library](#)

관련 동영상:

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

OPS03-BP07 팀에 적절한 리소스 제공

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

원하는 결과:

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

일반적인 안티 패턴:

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

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

구현 단계

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

5. 현재 프로비저닝된 리소스가 여전히 충분한지, 추가 리소스가 필요한지 정기적으로 검토하고 팀에 맞게 조정하여 지원합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS03-BP06 팀원의 기술 역량 유지와 강화 장려](#)
- [OPS09-BP03 운영 지표 검토 및 개선 우선 순위 지정](#)
- [OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용](#)
- [OPS10-BP07 이벤트 대응 자동화](#)

관련 문서:

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

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.

운영 우수성 달성을 준비하기 위해 수행해야 하는 사항은 다음과 같습니다.

주제

- [관찰성 구현](#)
- [운동을 고려한 설계](#)
- [배포 위험 완화](#)
- [운영 준비 상태 및 변경 관리](#)

관찰성 구현

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

관찰성은 단순한 모니터링을 넘어서서 외부 출력을 기반으로 시스템의 내부 작동을 포괄적으로 이해할 수 있게 합니다. 지표, 로그 및 추적에 기반을 둔 관찰성은 시스템 동작 및 역학에 대한 심층적인 통찰력을 제공합니다. 효과적인 관찰성을 통해 팀은 패턴, 이상 및 추세를 식별하여 잠재적 문제를 사전에 해결하고 최적의 시스템 상태를 유지할 수 있습니다.

모니터링 활동과 비즈니스 목표를 일치시키기 위해서는 핵심성과 지표(KPI)를 식별하는 것이 매우 중요합니다. 이러한 조정을 통해 팀은 진정으로 중요한 메트릭을 사용하여 데이터 기반 결정을 내리고 시스템 성능과 비즈니스 결과를 모두 최적화할 수 있습니다.

또한 관찰성을 통해 기업은 사후 대응이 아닌 사전 대응이 가능합니다. 팀은 단순히 대응하는 데 그치지 않고 시스템 내의 인과 관계를 이해하여 문제를 예측하고 예방할 수 있습니다. 워크로드가 진화함에 따라 관찰성 전략을 재검토하고 개선하여 관련성과 효율성을 유지하는 것이 중요합니다.

모범 사례

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)

- [OPS04-BP03 사용자 경험 원격 측정 구현](#)
- [OPS04-BP04 종속성 원격 측정 구현](#)
- [OPS04-BP05 분산 추적 구현](#)

OPS04-BP01 핵심 성과 지표 파악

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

워크로드 KPI를 효과적으로 정의하려면:

1. 비즈니스 성과부터 시작하세요. 지표를 자세히 살펴보기 전에 원하는 비즈니스 성과를 파악하세요. 매출 증대, 사용자 참여 증대 또는 응답 시간 단축이 필요한가요?
2. 기술 지표와 비즈니스 목표의 상관 관계 파악: 모든 기술 지표가 비즈니스 성과에 직접적인 영향을 미치는 것은 아닙니다. 비즈니스 성과에 직접적인 영향을 미치는 기술 지표를 파악하세요. 하지만 비즈니스 KPI를 사용하여 문제를 식별하는 것이 더 간단한 경우가 많습니다.
3. Amazon CloudWatch [사용](#): CloudWatch를 사용하여 KPI를 나타내는 지표를 정의하고 모니터링하세요.
4. KPI를 정기적으로 검토하고 업데이트하세요. 워크로드와 비즈니스가 발전함에 따라 적절한 KPI를 유지하세요.
5. 이해 관계자 참여: KPI를 정의하고 검토하는 데 기술 팀과 비즈니스 팀 모두를 참여시키세요.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

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

관련 문서:

- [AWS 관찰성 모범 사례](#)
- [CloudWatch 사용 설명서](#)
- [AWS 관찰성 스킬 빌더 코스](#)

관련 동영상:

- [관찰성 전략 개발](#)

관련 예시:

- [One Observability Workshop](#)

OPS04-BP02 애플리케이션 원격 측정 구현

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

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

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

구현 단계

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

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

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- [OPS04-BP01 워크로드 KPI 정의](#)
- [OPS04-BP03 사용자 활동 원격 측정 구현](#)
- [OPS04-BP04 종속성 원격 측정 구현](#)
- [OPS04-BP05 트랜잭션 추적 기능 구현](#)

관련 문서:

- [AWS 관측성 모범 사례](#)
- [CloudWatch 사용 설명서](#)
- [AWS X-Ray 개발자 안내서](#)
- [운영 가시성을 위한 분산 시스템 계측](#)
- [AWS 관찰성 스킬 빌더 코스](#)
- [Amazon CloudWatch의 새로운 소식](#)
- [AWS X-Ray의 새로운 소식](#)

관련 동영상:

- [AWS re:Invent 2022 - Observability best practices at Amazon](#)

- [AWS re:Invent 2022 - Developing an observability strategy](#)

관련 예시:

- [One Observability Workshop](#)
- [AWS Solutions Library: Application Monitoring with Amazon CloudWatch](#)

OPS04-BP03 사용자 경험 원격 측정 구현

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

구현 단계

1. Amazon CloudWatch RUM 배포: 애플리케이션을 CloudWatch RUM과 통합하여 실제 사용자 데이터를 수집, 분석 및 제공합니다.
 - a. RUM을 애플리케이션과 통합하기 위해 [CloudWatch RUM 자바스크립트 라이브러리](#)를 사용하세요.
 - b. 대시보드를 설정하여 실제 사용자 데이터를 시각화하고 모니터링할 수 있습니다.
2. CloudWatch Synthetics 구성: 애플리케이션과 사용자 상호 작용을 시뮬레이션하는 카나리아 또는 스크립팅된 루틴을 만들 수 있습니다.
 - a. 중요 애플리케이션 워크플로 및 경로를 정의합니다.
 - b. 이러한 경로에 대한 사용자 상호 작용을 시뮬레이션 하기 위해 [CloudWatch Synthetics 스크립트](#)를 사용해 카나리아를 디자인합니다.
 - c. 카나리아가 지정된 간격으로 실행되도록 스케줄링하고 모니터링하여 일관된 성능 검사를 보장합니다.
3. 데이터 분석 및 조치: RUM 및 가상 트랜잭션의 데이터를 활용하여 통찰력을 얻고 이상이 감지되면 수정 조치를 취하세요. CloudWatch 대시보드와 경보를 사용하여 최신 정보를 확인하세요.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)
- [OPS04-BP04 종속성 원격 측정 구현](#)
- [OPS04-BP05 분산 추적 구현](#)

관련 문서:

- [Amazon CloudWatch RUM 가이드](#)
- [Amazon CloudWatch Synthetics 가이드](#)

관련 동영상:

- [Amazon CloudWatch RUM을 통해 최종 사용자 인사이트로 애플리케이션 최적화](#)
- [AWS On Air ft. Amazon CloudWatch을 위한 RUM\(실제 사용자 모니터링\)](#)

관련 예시:

- [One Observability Workshop](#)
- [Amazon CloudWatch RUM 웹 클라이언트용 Git 리포지토리](#)
- [Amazon CloudWatch Synthetics를 사용하여 페이지 로드 시간 측정](#)

OPS04-BP04 종속성 원격 측정 구현

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

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

일반적인 안티 패턴:

- 외부 종속성 간과: 내부 애플리케이션 지표에만 초점을 맞추고 외부 종속성과 관련된 지표는 무시합니다.
- 사전 모니터링 부족: 종속성 상태 및 성능을 지속적으로 모니터링하는 대신 문제가 발생할 때까지 기다립니다.
- 사일로화된 모니터링: 여러 개의 다른 모니터링 도구를 사용하여 종속성 상태에 대한 보기가 단편적이고 일관성이 없을 수 있습니다.

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

구현 단계

종속성 원격 측정을 효과적으로 구현하는 방법:

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

2. 모니터링 전략 수립: 외부 종속성을 명확하게 파악한 후에는 그에 맞는 모니터링 전략을 세웁니다. 여기에는 각 종속성의 중요도, 예상되는 동작, 관련 서비스 수준에 관한 계약 또는 대상(SLA 또는 SLT)을 이해하는 것이 포함됩니다. 사전 알림을 설정하여 상태 변경 또는 성능 편차에 대한 알림을 받습니다.
3. [네트워크 모니터링](#) 사용: 전 세계 인터넷 및 네트워크 상태에 대한 포괄적인 인사이트를 제공하는 [Internet Monitor](#) 및 [Network Monitor](#)를 사용합니다. 이러한 도구는 외부 종속성에 영향을 미치는 운영 중단, 장애 또는 성능 저하를 이해하고 이에 대응하는 데 도움이 됩니다.
4. [AWS Health Dashboard](#)로 [최신 정보 획득](#): AWS에 서비스에 영향을 줄 수 있는 이벤트가 발생할 경우 이를 알리고 수정 지침을 제공합니다.
 - a. [Amazon EventBridge 규칙을 사용하여 AWS Health 이벤트를](#) 모니터링하거나 AWS Health API와 프로그래밍 방식으로 통합하여 AWS Health 이벤트를 수신할 때 작업을 자동화합니다. 이러한 작업은 계획된 모든 수명 주기 이벤트 메시지를 채팅 인터페이스로 보내는 것과 같은 일반적인 작업이거나 IT 서비스 관리 도구에서 워크플로를 시작하는 것과 같은 구체적인 작업일 수 있습니다.
 - b. AWS Organizations를 사용하는 경우 전체 계정의 [AWS Health 이벤트를](#) 집계합니다.
5. [AWS X-Ray로 애플리케이션 계측](#): AWS X-Ray는 애플리케이션과 기본 종속성이 어떻게 수행되는지에 대한 인사이트를 제공합니다. 요청을 처음부터 끝까지 추적하여 애플리케이션이 의존하는 외부 서비스 또는 구성 요소의 병목 현상이나 장애를 식별할 수 있습니다.
6. [Amazon DevOps Guru 사용](#): 이 기계 학습 기반 서비스는 운영 문제를 식별하고, 중대한 문제가 발생할 수 있는 시기를 예측하고, 취해야 할 구체적인 조치를 제시합니다. 종속성에 대한 인사이트를 얻고 종속성에서 운영 문제가 발생하지 않도록 하는 데 매우 중요합니다.
7. 주기적 모니터링: 외부 종속성과 관련된 지표 및 로그를 지속적으로 모니터링합니다. 예상치 못한 동작이나 성능 저하에 대한 알림을 설정합니다.
8. 변경 후 검증: 외부 종속성이 업데이트되거나 변경될 때마다 성능을 검증하고 애플리케이션 요구 사항에 맞는 지 확인합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS04-BP01 워크로드 KPI 정의](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)
- [OPS04-BP03 사용자 활동 원격 측정 구현](#)

- [OPS04-BP05 트랜잭션 추적 기능 구현](#)
- [OP08-BP04 실행 가능한 알림 생성](#)

관련 문서:

- [Amazon Personal AWS Health Dashboard 사용 설명서](#)
- [AWS Internet Monitor 사용 설명서](#)
- [AWS X-Ray 개발자 안내서](#)
- [AWS DevOps Guru 사용 설명서](#)

관련 동영상:

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

관련 예시:

- [Gaining operational insights with AIOps using Amazon DevOps Guru](#)
- [AWS Health Aware](#)
- [Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale](#)

OPS04-BP05 분산 추적 구현

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

구현 단계

분산 추적을 효과적으로 구현하려면:

1. AWS X-Ray **채택**: X-Ray을 애플리케이션에 통합하여 애플리케이션 동작에 대한 인사이트를 얻고 성능을 이해하며 병목 현상을 정확히 찾아내세요. 자동 추적 분석을 위해 X-Ray Insights를 활용하세요.
2. 서비스 계측: 다음의 [AWS Lambda](#) 함수에서 [EC2 인스턴스에](#) 이르기까지 모든 서비스가 추적 데이터를 전송하는지 확인하세요. 더 많은 서비스를 계측할수록 엔드투엔드 뷰가 더 명확해집니다.
3. 실제 [CloudWatch 사용자 모니터링](#) 및 [합성 모니터링을 통합합니다](#): 실제 사용자 모니터링(RUM) 및 합성 모니터링과 X-Ray를 통합하세요. 이를 통해 실제 사용자 경험을 캡처하고 사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 식별할 수 있습니다.
4. CloudWatch 에이전트를 [사용하세요](#): 에이전트는 X-Ray 또는 OpenTelemetry 중 하나에서 트레이스를 전송하여 더 심도 깊은 통찰력을 얻을 수 있습니다.
5. Amazon DevOps Guru [사용](#): DevOps Guru는 X-Ray, CloudWatch, AWS Config 및 AWS CloudTrail의 데이터를 사용하여 실행 가능한 권장 사항을 제공합니다.
6. 트레이스 분석: 추적 데이터를 정기적으로 검토하여 애플리케이션 성능에 영향을 줄 수 있는 패턴, 이상 또는 병목 현상을 식별하세요.

7. 알림 설정: 비정상적인 패턴이나 지연 시간 연장에 대해 [CloudWatch에서](#) 알람을 구성하여 선제적으로 문제를 해결할 수 있습니다.
8. 지속적인 개선: 모든 관련 데이터 포인트를 캡처하도록 서비스가 추가 또는 수정되면 추적 전략을 다시 검토하세요.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)
- [OPS04-BP03 사용자 경험 원격 측정 구현](#)
- [OPS04-BP04 종속성 원격 측정 구현](#)

관련 문서:

- [AWS X-Ray 개발자 가이드](#)
- [Amazon CloudWatch 에이전트 사용 설명서](#)
- [Amazon DevOps Guru 사용 설명서](#)

관련 동영상:

- [Use AWS X-Ray Insights](#)
- [AWS On Air ft. Observability: Amazon CloudWatch and AWS X-Ray](#)

관련 예시:

- [AWS X-Ray을 사용하여 애플리케이션 계측](#)

운영을 고려한 설계

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.

AWS에서는 전체 워크로드(애플리케이션, 인프라, 정책, 거버넌스, 운영)를 코드로 확인할 수 있습니다. 즉, 코드를 사용하여 모든 워크로드를 정의하고 업데이트할 수 있습니다. 그러면 애플리케이션 코드에 사용하는 것과 같은 엔지니어링 원칙을 스택의 모든 요소에 적용할 수 있습니다.

모범 사례

- [OPS05-BP01 버전 관리 사용](#)
- [OPS05-BP02 변경 사항 테스트 및 검증](#)
- [OPS05-BP03 구성 관리 시스템 사용](#)
- [OPS05-BP04 빌드 및 배포 관리 시스템 사용](#)
- [OPS05-BP05 패치 관리 수행](#)
- [OPS05-BP06 설계 표준 공유](#)
- [OPS05-BP07 코드 품질 개선을 위한 사례 구현](#)
- [OPS05-BP08 여러 환경 사용](#)
- [OPS05-BP09 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용](#)
- [OPS05-BP10 통합 및 배포 완전 자동화](#)

OPS05-BP01 버전 관리 사용

버전 관리를 사용하면 변경 사항과 릴리스를 추적할 수 있습니다.

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

원하는 결과: 팀은 코드를 사용하여 협업합니다. 병합되면 코드가 일관되고 변경 내용이 손실되지 않습니다. 올바른 버전 관리를 통해 오류를 쉽게 되돌릴 수 있습니다.

일반적인 안티 패턴:

- 워크스테이션에서 코드를 개발하고 저장해 왔습니다. 워크스테이션에서 복구할 수 없는 스토리지 오류가 발생하여 코드가 손실되었습니다.
- 기존 코드를 변경 사항으로 덮어쓴 후 애플리케이션을 다시 시작하면 애플리케이션이 더 이상 작동하지 않습니다. 변경 사항을 되돌릴 수 없습니다.

- 다른 사람이 편집해야 하는 보고서 파일에 대한 쓰기 잠금이 있습니다. 태스크를 완료할 수 있도록 태스크 작업 중지를 요청하는 연락을 받습니다.
- 연구 팀은 향후 작업을 결정할 세부 분석을 수행해 왔습니다. 누군가 실수로 최종 보고서에 쇼핑 목록을 저장했습니다. 변경 사항을 되돌릴 수 없으며 보고서를 다시 생성해야 합니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

리소스

관련 모범 사례:

- [OPS05-BP04 빌드 및 배포 관리 시스템 사용](#)

관련 문서:

- [AWS CodeCommit란 무엇인가요?](#)

관련 동영상:

- [AWS CodeCommit 소개](#)

OPS05-BP02 변경 사항 테스트 및 검증

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

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

원하는 결과: 소프트웨어 변경 사항이 적용되기 전에 테스트됩니다. 개발자가 테스트 결과 및 검증에 액세스할 수 있습니다. 조직에 모든 소프트웨어 변경에 적용되는 테스트 표준이 있습니다.

일반적인 안티 패턴:

- 아무런 테스트 없이 새로운 소프트웨어 변경 사항을 배포했습니다. 프로덕션 환경에서 실행에 실패하면 가동 중단으로 이어집니다.
- 새로운 보안 그룹이 프로덕션 전 환경에서 테스트 없이 AWS CloudFormation을 사용하여 배포됩니다. 보안 그룹이 고객이 앱에 연결할 수 없도록 합니다.
- 메시지가 수정되었으나 단위 테스트가 수행되지 않습니다. 소프트웨어가 프로덕션 환경에 배포되면 장애가 발생합니다.

이 모범 사례 확립의 이점: 소프트웨어 배포의 변경 실패율이 감소합니다. 소프트웨어 품질이 개선됩니다. 개발자가 코드의 가시성에 대한 인식을 높였습니다. 조직의 규정 준수를 지원한다는 확신을 가지고 보안 정책을 롤아웃할 수 있습니다. 트래픽 수요를 충족하기 위해 Auto Scaling 정책 업데이트 등과 같은 인프라 변경을 사전에 테스트합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

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

고객 사례

지속적 통합 파이프라인의 일부로, AnyCompany Retail에서는 모든 소프트웨어 아티팩트에 대해 여러 가지 유형의 테스트를 수행합니다. 테스트 기반 개발을 수행하기 때문에 모든 소프트웨어에 단위 테스트가 있습니다. 아티팩트가 구축되면 엔드 투 엔드 테스트를 실행합니다. 테스트의 1차 라운드가 완료

된 후 알려진 취약점을 찾는 정적 애플리케이션 보안 검사를 실행합니다. 각 테스트 관문을 통과할 때마다 개발자에게 메시지가 전송됩니다. 모든 테스트가 완료되면 소프트웨어 아티팩트는 아티팩트 리포지토리에 저장됩니다.

구현 단계

1. 조직 내 이해관계자와 함께 소프트웨어 아티팩트를 위한 테스트 표준을 개발합니다. 모든 아티팩트가 어떤 표준 테스트를 통과해야 하나요? 테스트 범위에 포함해야 하는 규정 준수 또는 거버넌스 요구 사항이 있나요? 코드 품질 테스트를 수행해야 하나요? 테스트가 완료되면 누구에게 알려야 하나요?
 1. [AWS 배포 파이프라인 참조 아키텍처](#)에는 통합 파이프라인의 일부로 소프트웨어 아티팩트에 대해 수행할 수 있는 테스트 유형이 나열된 신뢰할 수 있는 목록이 포함되어 있습니다.
2. 소프트웨어 테스트 표준을 기준으로 필수 테스트를 통해 애플리케이션을 계측합니다. 각 테스트 세트는 10분 이내에 완료해야 합니다. 테스트는 통합 파이프라인의 일부로 실행되어야 합니다.
 - a. 단위 테스트 사례(경계 조건 포함)를 생성하고, 코드 및 주석을 사용하여 함수를 생성하고, 잘 알려진 알고리즘을 구현하는 데 도움이 되는 생성형 AI 도구인 [Amazon Q Developer](#)를 사용합니다.
 - b. [Amazon CodeGuru Reviewer](#)를 사용하여 애플리케이션 코드에 결함이 있는지 테스트합니다.
 - c. [AWS CodeBuild](#)를 사용하여 소프트웨어 아티팩트에 대한 테스트를 수행할 수 있습니다.
 - d. [AWS CodePipeline](#)에서는 소프트웨어 테스트를 파이프라인으로 오케스트레이션할 수 있습니다.

리소스

관련 모범 사례:

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

관련 문서:

- [테스트 기반 개발 접근 방식 채택](#)
- [Accelerate your Software Development Lifecycle with Amazon Q](#)
- [Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience](#)

- [The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE](#)
- [Shift-Left Workload, leveraging AI for Test Creation](#)
- [Amazon Q Developer Center](#)
- [10 ways to build applications faster with Amazon CodeWhisperer](#)
- [Looking beyond code coverage with Amazon CodeWhisperer](#)
- [Best Practices for Prompt Engineering with Amazon CodeWhisperer](#)
- [TaskCat 및 CodePipeline으로 자동화된 AWS CloudFormation 테스트 파이프라인](#)
- [오픈 소스 SCA, SAST 및 DAST 도구를 사용하여 엔드 투 엔드 AWS DevSecOps CI/CD 파이프라인 구축](#)
- [서버리스 애플리케이션 테스트로 시작하기](#)
- [CI/CD 파이프라인이 릴리스에 매우 중요함](#)
- [AWS에서 지속적인 통합 및 지속적인 전송 적용 백서](#)

관련 동영상:

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

관련 리소스:

- [Building applications using generative AI with Amazon CodeWhisperer](#)
- [Amazon CodeWhisperer Workshop](#)
- [AWS 배포 파이프라인 참조 아키텍처 - 애플리케이션](#)
- [AWS Kubernetes DevSecOps 파이프라인](#)
- [코드형 정책 워크숍 - 테스트 기반 개발](#)
- [를 사용하여 GitHub에서 Node.js 애플리케이션에 대한 단위 테스트 실행 AWS CodeBuild](#)
- [인프라 코드 테스트 기반 개발에 Serverspec 사용](#)

관련 서비스:

- [Amazon Q Developer](#)
- [Amazon CodeGuru Reviewer](#)
- [AWS CodeBuild](#)
- [AWS CodePipeline](#)

OPS05-BP03 구성 관리 시스템 사용

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

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

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

AWS에서는 [AWS Config를 사용하여](#) 계정과 리전 전반에서 AWS 리소스 구성을 [지속적으로 모니터링할 수 있습니다.](#) 이를 통해 구성 이력을 추적하고, 구성 변경이 다른 리소스에 어떤 영향을 미치는지 이해하며, 그리고 [AWS Config 규칙 및 AWS Config 규정 준수 팩을 사용해 예상 또는 원하는 구성과 비교하여 감사합니다.](#)

Amazon EC2인스턴스, AWS Lambda, 컨테이너, 모바일 애플리케이션 또는 IoT 기기에서 실행되는 애플리케이션에 동적 구성이 있는 경우 [AWS AppConfig](#)를 사용하여 환경 전반에서 애플리케이션을 구성, 검증, 배포 및 모니터링할 수 있습니다.

AWS에서는 다음과 같은 서비스를 사용하여 지속적 통합 및 지속적 배포(CI/CD) 파이프라인을 구축할 수 있습니다. [AWS 개발자 도구](#) (예를 들면 다음과 같습니다. [AWS CodeCommit](#), [AWS CodeBuild](#), [AWS CodePipeline](#), [AWS CodeDeploy](#) 및 [AWS CodeStar](#)).

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

일반적인 안티 패턴:

- 플릿 전체에서 웹 서버 구성을 수동으로 업데이트하면 업데이트 오류로 인해 여러 서버가 응답하지 않게 됩니다.
- 여러 시간 동안 애플리케이션 서버 플릿을 수동으로 업데이트합니다. 변경 중 구성 불일치로 인해 예기치 않은 동작이 발생합니다.
- 누군가가 보안 그룹을 업데이트했으며 웹 서버에 더 이상 액세스할 수 없습니다. 변경된 사항을 알지 못하면 문제를 조사하는 데 상당한 시간이 들어서 복구 시간이 늘어납니다.
- 검증 없이 CI/CD를 통해 사전 프로덕션 구성을 프로덕션 환경으로 푸시합니다. 사용자와 고객을 잘못된 데이터와 서비스에 노출시킵니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

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

리소스

관련 모범 사례:

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

관련 문서:

- [AWS Control Tower](#)
- [AWS Landing Zone Accelerator](#)
- [AWS Config](#)
- [AWS Config란 무엇인가요?](#)
- [AWS AppConfig를](#)
- [AWS CloudFormation란 무엇인가요?](#)
- [AWS 개발자 도구](#)

관련 동영상:

- [AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads](#)
- [AWS re:Invent 2020: Achieve compliance as code using AWS Config](#)
- [AWS AppConfig를 사용하여 애플리케이션 구성 관리 및 배포](#)

OPS05-BP04 빌드 및 배포 관리 시스템 사용

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

AWS에서는 다음과 같은 서비스를 사용하여 지속적 통합 및 지속적 배포(CI/CD) 파이프라인을 구축할 수 있습니다. [AWS 개발자 도구](#) (예: AWS CodeCommit, [AWS CodeBuild](#), [AWS CodePipeline](#), [AWS CodeDeploy](#) 및 [AWS CodeStar](#)).

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

일반적인 안티 패턴:

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

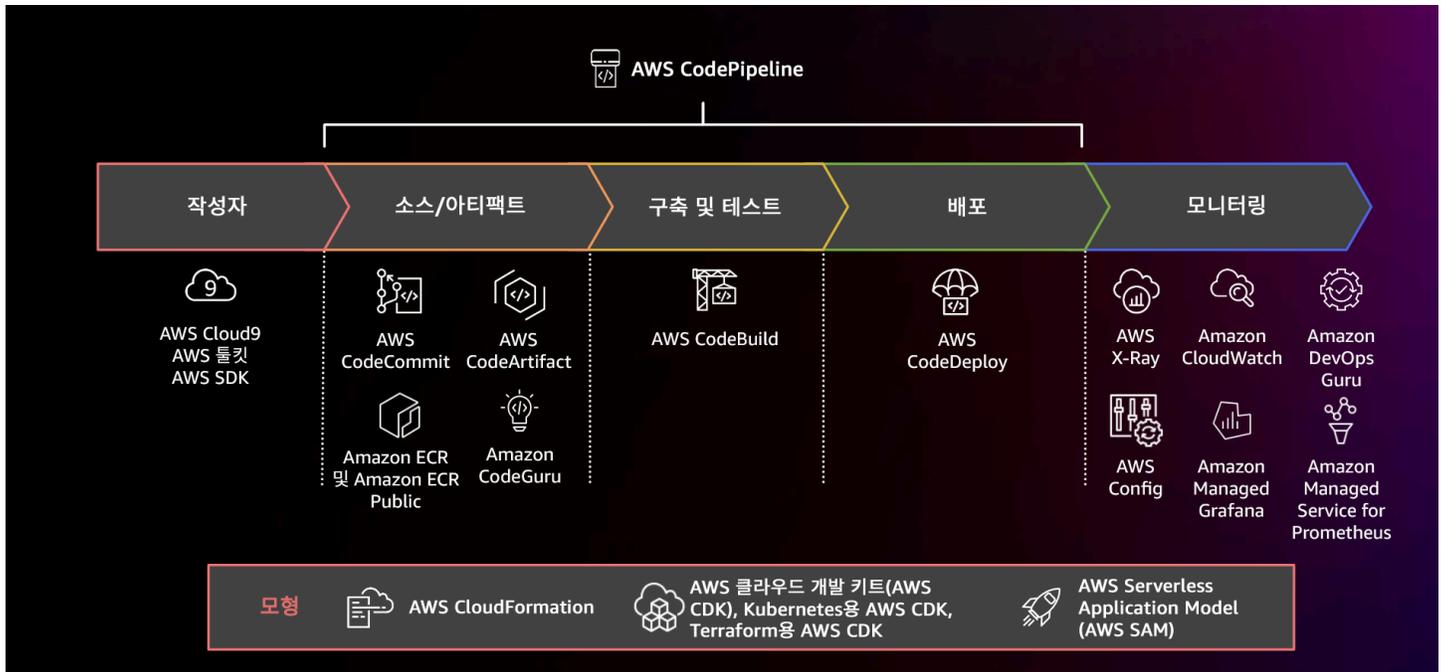
이 모범 사례 확립의 이점: 빌드 및 배포 활동을 관리하는 메커니즘을 제공하여 반복적인 작업 수행을 위한 작업량을 줄이고, 팀원들이 고가치 창조 작업에 집중할 수 있게 하고, 수동 절차에서 발생하는 오류의 도입을 제한할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계



AWS CodePipeline과 관련 서비스를 사용하는 CI/CD 파이프라인을 보여주는 다이어그램

1. 자산(예: 문서, 소스 코드, 바이너리 파일)의 버전을 관리하고, 저장하고, 관리하는 데 AWS CodeCommit를 사용합니다.
2. 소스 코드를 컴파일하고, 단위 테스트를 실행하고, 배포할 준비가 된 아티팩트를 생성하는 데 CodeBuild를 사용합니다.
3. CodeDeploy를 [Amazon EC2](#) 인스턴스, 온프레미스 인스턴스, [서버리스 AWS Lambda 기능](#) 또는 [Amazon ECS에 애플리케이션 배포를 자동화하는 배포 서비스로 사용하세요.](#)
4. 배포를 모니터링하세요.

리소스

관련 모범 사례:

- [OPS06-BP04 테스트 및 롤백 자동화](#)

관련 문서:

- [AWS 개발자 도구](#)
- [AWS CodeCommit란 무엇인가요?](#)

- [AWS CodeBuild란 무엇인가요?](#)
- [AWS CodeBuild](#)
- [AWS CodeDeploy란 무엇인가요?](#)

관련 동영상:

- [AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS](#)

OPS05-BP05 패치 관리 수행

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

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

[Amazon EC2 이미지 빌더](#)는 머신 이미지를 업데이트하기 위한 파이프라인을 제공합니다. 패치 관리의 일환으로 [AMI 이미지 파이프라인을 사용하는](#) Amazon Machine Image(AMI) 또는 [도커 이미지 파이프라인이 있는](#) 컨테이너 이미지를 [고려하세요](#). AWS Lambda는 취약점을 제거하기 위해 [사용자 지정 런타임 및 추가 라이브러리](#)를 위한 패턴을 제공합니다.

Amazon EC2 이미지 빌더를 사용하여 Linux용 [Amazon Machine Image](#) 또는 Windows Server 이미지에 대한 업데이트를 [관리해야 합니다](#). 이때 기존 파이프라인과 함께 [Amazon Elastic Container Registry \(Amazon ECR\)](#)를 사용하여 Amazon ECS와 Amazon EKS 이미지를 관리합니다. Lambda은 [버전 관리 기능을 포함합니다](#).

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

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

일반적인 안티 패턴:

- 2시간 내에 최신 보안 패치를 모두 적용해야 하는데 애플리케이션과 패치가 호환되지 않아 여러 번 중단될 수 있습니다.
- 패치가 적용되지 않은 라이브러리는 알 수 없는 당사자가 워크로드에 액세스하기 위해 해당 라이브러리의 취약성을 이용하므로 의도하지 않은 결과를 초래합니다.
- 개발자에게 알리지 않고 개발자 환경에 자동으로 패치를 적용합니다. 개발자가 환경이 예상대로 작동하지 않는다는 불만을 여러 번 제기합니다.
- 영구 인스턴스에 자체 상용 소프트웨어를 패치하지 않았습니다. 소프트웨어에 문제가 있어서 공급자에게 문의하면 해당 버전이 지원되지 않으며 지원을 받으려면 특정 수준으로 패치해야 한다는 답을 듣습니다.
- 사용한 암호화 소프트웨어에 대해 최근에 릴리스된 패치의 성능이 크게 향상되었습니다. 패치가 적용되지 않은 시스템에 성능 문제가 있습니다.
- 긴급 수정이 필요한 제로데이 취약성에 대한 알림을 받게 되며 모든 환경을 수동으로 패치해야 합니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

Amazon EC2 이미지 빌더를 위해서:

1. Amazon EC2 이미지 빌더를 사용하여 파이프라인 세부 정보를 지정합니다.
 - a. 이미지 파이프라인 생성 및 이름 지정
 - b. 파이프라인 일정 및 시간대 정의

- c. 모든 종속성 구성
2. 레시피 선택:
 - a. 기존 레시피 선택 또는 새 레시피 생성
 - b. 이미지 유형 선택
 - c. 레시피 이름 및 버전 지정
 - d. 기본 이미지 선택
 - e. 빌드 구성 요소 추가 및 대상 레지스트리에 추가
3. 선택 사항 - 인프라 구성을 정의합니다.
4. 선택 사항 - 구성 설정을 정의합니다.
5. 리뷰 설정.
6. 레시피 위생을 정기적으로 유지하십시오.

Systems Manager 패치 관리자를 위해서:

1. 패치 기준선을 생성합니다.
2. 경로 지정 작업 방법을 선택합니다.
3. 규정 준수 보고 및 스캔을 활성화합니다.

리소스

관련 모범 사례:

- [OPS06-BP04 테스트 및 롤백 자동화](#)

관련 문서:

- [Amazon EC2 이미지 빌더는 무엇인가요?](#)
- [Amazon EC2 이미지 빌더를 사용하여 이미지 파이프라인 생성](#)
- [컨테이너 이미지 파이프라인 생성](#)
- [AWS Systems Manager 패치 관리자](#)
- [패치 관리자 사용](#)
- [패치 규정 준수 보고서 사용](#)
- [AWS 개발자 도구](#)

관련 동영상:

- [AWS의 서버리스 애플리케이션용 CI/CD](#)
- [운영 설계를 염두에 두세요](#)

관련 예시:

- [Well-Architected 실습 - 인벤토리 및 패치 관리](#)
- [AWS Systems Manager 패치 관리자 튜토리얼](#)

OPS05-BP06 설계 표준 공유

여러 팀이 모범 사례를 공유하면 표준에 대한 인지도를 높이고 개발 작업의 이점을 극대화할 수 있습니다. 아키텍처가 변경됨에 따라 표준을 문서화하고 최신 상태를 유지합니다. 조직에 공유 표준이 적용되면 표준에 대한 추가, 변경 및 예외 처리를 요청하는 메커니즘을 확보해야 합니다. 이 옵션이 없으면 표준이 혁신의 제약 요인이 됩니다.

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

일반적인 안티 패턴:

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

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

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

관련 문서:

- [AWS Service Catalog로 AWS Backup 자동화](#)

- [AWS Service Catalog Account Factory 개선](#)
- [Expedia Group에서 AWS Service Catalog를 사용하여 DBaaS\(Database as a Service\) 제품을 구축한 방법](#)
- [Maintain visibility over the use of cloud architecture patterns\(클라우드 아키텍처 패턴 사용에 대한 가시성 유지\)](#)
- [AWS Organizations 설정에서 AWS Service Catalog 포트폴리오 공유 간소화](#)

관련 동영상:

- [AWS Service Catalog – 시작하기](#)
- [AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert](#)

관련 예시:

- [AWS Service Catalog 참조 아키텍처](#)
- [AWS Service Catalog 워크숍](#)

관련 서비스:

- [AWS Service Catalog](#)

OPS05-BP07 코드 품질 개선을 위한 사례 구현

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

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

일반적인 안티 패턴:

- 코드 검토 없이 애플리케이션의 기본 분기에 코드를 커밋합니다. 변경 사항은 프로덕션에 자동으로 배포되고 중단이 발생합니다.

- 단위, 엔드 투 엔드 또는 통합 테스트 없이 새 애플리케이션을 개발합니다. 배포 전에 애플리케이션을 테스트할 방법이 없습니다.
- 팀은 결함을 해결하기 위해 프로덕션에서 수동으로 변경합니다. 변경 사항은 테스트 또는 코드 검토 단계를 거치지 않으며 지속적 통합 및 전달 프로세스를 통해 캡처되거나 로깅되지 않습니다.

이 모범 사례 확립의 이점: 코드 품질을 개선하기 위한 방법을 채택하면 프로덕션 환경에서 발생하는 문제를 최소화할 수 있습니다. 코드 품질은 페어 프로그래밍, 코드 검토, AI 생산성 도구 도입과 같은 모범 사례의 사용을 촉진합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

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

고객 사례

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

구현 단계

1. 테스트 기반 개발, 코드 검토, 페어 프로그래밍과 같은 코드 품질 관련 사례를 지속적 통합 및 전달 프로세스에 도입합니다. 이러한 기법을 사용하여 소프트웨어 품질을 개선합니다.
 - a. [Amazon Q Developer](#)를 사용합니다. 이 생성형 AI 도구는 단위 테스트 사례(경계 조건 포함)를 생성하고, 코드 및 주석을 사용하여 함수를 생성하고, 잘 알려진 알고리즘을 구현하고, 코드의 보안 정책 위반 및 취약성을 탐지하고, 보안 암호를 탐지하고, 코드형 인프라(IaC)를 스캔하고, 코드를 문서화하고, 타사 코드 라이브러리를 더 빠르게 학습하는 데 도움이 됩니다.
 - b. [Amazon CodeGuru Reviewer](#)는 기계 학습을 사용하여 Java 및 Python 코드에 대한 프로그래밍 권장 사항을 제공할 수 있습니다.
 - c. 코드 개발에 대해 협업할 수 있는 공유 개발 환경은 [AWS Cloud9](#)으로 생성할 수 있습니다.

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

리소스

관련 모범 사례:

- [OPS05-BP02 변경 사항 테스트 및 검증](#)
- [OPS05-BP06 설계 표준 공유](#)

관련 문서:

- [테스트 기반 개발 접근 방식 채택](#)
- [Accelerate your Software Development Lifecycle with Amazon Q](#)
- [Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience](#)
- [The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE](#)
- [Shift-Left Workload, leveraging AI for Test Creation](#)
- [Amazon Q Developer Center](#)
- [10 ways to build applications faster with Amazon CodeWhisperer](#)
- [Looking beyond code coverage with Amazon CodeWhisperer](#)
- [Best Practices for Prompt Engineering with Amazon CodeWhisperer](#)
- [Agile Software Guide](#)
- [CI/CD 파이프라인이 릴리스에 매우 중요함](#)
- [Amazon CodeGuru Reviewer를 사용한 코드 검토 자동화](#)
- [테스트 기반 개발 접근 방식 채택](#)
- [DevFactory가 Amazon CodeGuru를 사용하여 더 나은 애플리케이션을 구축하는 방법](#)
- [On Pair Programming](#)
- [RENGA Inc., Amazon CodeGuru로 코드 검토 자동화](#)
- [The Art of Agile Development: Test-Driven Development](#)
- [코드 검토가 중요한 이유\(그리고 시간이 절약되는 이유\)](#)

관련 동영상:

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

관련 서비스:

- [Amazon Q Developer](#)
- [Amazon CodeGuru Reviewer](#)
- [Amazon CodeGuru Profiler](#)
- [AWS Cloud9](#)

OPS05-BP08 여러 환경 사용

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

원하는 결과: 규정 준수 및 거버넌스 요구 사항을 반영하는 여러 환경이 있습니다. 프로덕션 단계에 있는 환경을 통해 코드를 테스트하고 홍보합니다.

일반적인 안티 패턴:

- 공유 개발 환경에서 개발을 수행하고 있으며 다른 개발자가 코드 변경 사항을 덮어씁니다.
- 공유 개발 환경에 대한 제한적인 보안 제어로 인해 새로운 서비스와 기능을 실험할 수 없습니다.
- 프로덕션 시스템에서 로드 테스트를 수행하고 사용자를 중단시킵니다.
- 프로덕션 환경에서 데이터 손실을 일으키는 심각한 오류가 발생했습니다. 데이터 손실이 어떻게 발생했는지 파악하고 다시 발생하지 않도록 프로덕션 환경에서 데이터 손실을 일으키는 조건을 재현하려고 합니다. 테스트 중 추가 데이터 손실을 방지하기 위해 사용자가 애플리케이션을 사용할 수 없도록 해야 합니다.
- 멀티 테넌트 서비스를 운영 중이며 전용 환경에 대한 고객 요청을 지원할 수 없습니다.
- 항상 테스트하지는 않지만 테스트할 때는 프로덕션 환경에서 테스트합니다.
- 단일 환경의 단순성이 환경 내 변경 사항의 영향 범위를 우선한다고 생각합니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

다중 환경을 사용하고 실험이 가능한 최소한의 제어 기능이 있는 샌드박스 환경을 개발자에게 제공합니다. 개별 개발 환경을 제공하면 병렬 작업이 가능하므로 개발을 더 빠르게 진행할 수 있습니다. 프로덕션 환경과 인접한 환경에는 더욱 엄격한 제어 기능을 구현하여 개발자가 혁신을 이룰 수 있도록 합니다. 코드형 인프라 및 구성 관리 시스템을 사용하여 프로덕션 환경의 제어 기능과 일치하는 방식으로 구성된 환경을 배포합니다. 그러면 배포된 시스템이 정상적으로 작동합니다. 사용되고 있지 않은 환경은 유휴 리소스 관련 비용이 발생하지 않도록 해제합니다. 예를 들어 개발 시스템은 야간 시간과 주말에 해제합니다. 로드 테스트 시에는 올바른 결과를 얻을 수 있도록 프로덕션 환경에 상응하는 환경을 배포합니다.

리소스

관련 문서:

- [AWS의 인스턴스 스케줄러](#)
- [AWS CloudFormation란 무엇인가요?](#)

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

되돌릴 수 있는 소규모 변경 작업을 자주 수행하면 변경의 영향과 범위가 감소합니다. 변경 관리 시스템, 구성 관리 시스템, 빌드 및 제공 시스템과 함께 사용할 경우 자주 발생하는 작고 되돌릴 수 있는 변경으로 인해 변경의 범위와 영향이 줄어듭니다. 그러면 문제를 더 쉽게 해결할 수 있으며 변경 사항 롤백 옵션을 사용해 문제 해결 시간을 단축할 수 있습니다.

일반적인 안티 패턴:

- 분기별로 애플리케이션의 새 버전을 배포하며, 변경 기간은 코어 서비스가 해제되었음을 의미합니다.
- 관리 시스템의 변경 내용을 추적하지 않고 데이터베이스 스키마를 변경하는 경우가 많습니다.
- 수동 내부 업데이트를 수행하고 기존 설치 및 구성을 덮어쓰며 명확한 롤백 계획이 없습니다.

이 모범 사례 확립의 이점: 작은 변경 사항을 자주 배포하면 개발 작업이 더 빨라집니다. 변경 사항이 작으면 의도하지 않은 결과가 있는지 파악하기가 훨씬 더 쉽고 되돌리기도 더 쉽습니다. 변경 사항을 되돌릴 수 있는 경우 복구가 간소화됨에 따라 변경 사항 구현의 위험이 줄어듭니다. 변경 프로세스를 수행하면 위험이 줄어들고 변경 실패로 인한 영향도 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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

리소스

관련 모범 사례:

- [OPS05-BP03 구성 관리 시스템 사용](#)
- [OPS05-BP04 빌드 및 배포 관리 시스템 사용](#)
- [OPS06-BP04 테스트 및 롤백 자동화](#)

관련 문서:

- [AWS에서 마이크로서비스 구현](#)
- [마이크로서비스 - 관찰성](#)

OPS05-BP10 통합 및 배포 완전 자동화

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

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

원하는 결과: 개발자는 도구를 사용하여 코드를 제공하고 프로덕션으로 승격합니다. 개발자는 업데이트를 제공하기 위해 AWS Management Console에 로그인할 필요가 없습니다. 변경 및 구성에 대한 전

체 감사 추적이 있어 거버넌스 및 규정 준수 요구 사항을 충족합니다. 프로세스는 반복 가능하며 팀 간에 표준화됩니다. 개발자는 자유롭게 개발 및 코드 푸시에 집중할 수 있어 생산성이 향상됩니다.

일반적인 안티 패턴:

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

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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

리소스

관련 모범 사례:

- [OPS05-BP03 구성 관리 시스템 사용](#)
- [OPS05-BP04 빌드 및 배포 관리 시스템 사용](#)

관련 문서:

- [AWS CodeBuild란 무엇인가요?](#)
- [AWS CodeDeploy란 무엇인가요?](#)

관련 동영상:

- [AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS](#)

배포 위험 완화

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.

워크로드 설계에는 워크로드를 배포, 업데이트 및 운영하는 방식이 모두 포함되어 있어야 합니다. 결함을 줄이고 신속하고 안전하게 결함을 수정하는 엔지니어링 방식을 구현해야 합니다.

모범 사례

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

OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립

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

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

일반적인 안티 패턴:

- 배포를 수행했으며 애플리케이션이 불안정해졌지만 시스템에 활성 사용자가 있는 것 같습니다. 변경 사항을 롤백하고 활성 사용자에게 영향을 줄 것인지 아니면 사용자에게 영향을 줄 수 있으므로 기다렸다가 롤백할 것인지를 결정해야 합니다.

- 루틴을 변경한 후에는 새 환경에 액세스할 수 있지만, 서브넷 중 하나에 연결할 수 없게 됩니다. 전부를 롤백할지 아니면 액세스할 수 없는 서브넷을 수정할지 결정해야 합니다. 이러한 결정을 내리는 동안 서브넷에는 계속 연결할 수 없습니다.
- 시스템이 소규모 릴리스로 업데이트할 수 있는 방식으로 설계되지 않았습니다. 따라서 실패한 배포 중에 이러한 대량 변경 사항을 되돌리기가 어렵습니다.
- 코드형 인프라(IaC)를 사용하지 않으며 인프라가 수동으로 업데이트되어 원치 않는 구성을 초래했습니다. 수동 변경 사항을 효과적으로 추적하고 되돌릴 수 없습니다.
- 배포 빈도의 증가를 측정하지 않았으므로, 변경 사항의 크기를 줄이고 각 변경에 대한 롤백 계획을 개선하도록 팀을 장려하지 못하여 위험이 늘어나고 실패율이 증가합니다.
- 부적절한 변경으로 인한 운영 중단의 총 기간을 측정하지 않습니다. 팀이 배포 프로세스와 복구 계획 효율성의 우선 순위를 지정하고 개선할 수 없습니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

구현 단계

1. 팀이 지정된 기간 내에 변경 사항을 되돌릴 수 있는 효과적인 계획을 수립하도록 요구하는 정책을 문서화합니다.
 - a. 정책에는 수정 사항이 허용되는 시기가 명시되어야 합니다.
 - b. 관련된 모든 사람이 액세스할 수 있도록 롤백 계획을 문서화해야 합니다.
 - c. 롤백 요구 사항을 지정합니다(예: 무단 변경 사항이 배포된 것으로 확인된 경우).
2. 워크로드의 각 구성 요소와 관련된 모든 변경의 영향 수준을 분석합니다.
 - a. 반복 가능한 변경 사항이 변경 정책을 적용하는 일관된 워크플로를 따르는 경우 표준화 및 템플릿화되고 사전 승인되도록 허용합니다.
 - b. 복구에 들이는 시간을 줄이고 비즈니스에 미치는 영향을 줄일 수 있도록 변경 크기를 줄여 변경 사항의 잠재적 영향을 줄입니다.

- c. 가능한 경우 롤백 프로시저가 코드를 알려진 정상 상태로 되돌려 사고가 발생하지 않도록 합니다.
3. 도구와 워크플로를 통합하여 정책을 프로그래밍 방식으로 적용합니다.
4. 변경 사항에 대한 데이터를 다른 워크로드 책임자가 볼 수 있도록 하여 롤백할 수 없는 실패한 변경 사항의 진단 속도를 개선합니다.
 - a. 가시적인 변경 데이터를 사용하여 이러한 관행의 성공을 측정하고 반복적인 개선 사항을 파악합니다.
5. 모니터링 도구를 사용하여 배포의 성공 또는 실패를 확인하여 롤백에 대한 의사 결정 속도를 높입니다.
6. 변경이 부적절한 경우 운영 중단 기간을 측정하여 복구 계획을 지속적으로 개선합니다.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS06-BP04 테스트 및 롤백 자동화](#)

관련 문서:

- [AWS Builders Library | 배포 중 롤백 안전 보장](#)
- [AWS 백서 | 클라우드에서 변경 관리](#)

관련 동영상:

- [re:Invent 2019 | Amazon's approach to high-availability deployment](#)

OPS06-BP02 테스트 배포

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

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

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

일반적인 안티 패턴:

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

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

고객 사례

AnyCompany Retail은 지속적 통합 및 지속적 전달(CI/CD) 파이프라인의 일환으로 프로덕션과 유사한 환경에서 고객을 위한 인프라 및 소프트웨어 업데이트를 릴리스하는 데 필요한 정의된 단계를 수행합니다. 이 파이프라인은 배포 전에 리소스의 드리프트를 감지(IaC 외부에서 수행된 리소스의 변경을 감지)하기 위한 사전 검사와 IaC 시작 시 취하는 작업에 대한 검증으로 구성됩니다. 로드 밸런서에 다시

등록하기 전에 특정 파일 및 구성이 제자리에 있고 서비스가 실행 상태이고 로컬 호스트의 상태 확인에 올바르게 응답하는지 확인하는 등, 배포 단계를 검증합니다. 또한 모든 변경 사항은 기능, 보안, 회귀, 통합 및 로드 테스트 등의 여러 자동 테스트에 플래그를 지정합니다.

구현 단계

1. 설치 전 검사를 수행하여 사전 프로덕션 환경을 프로덕션에 미러링합니다.
 - a. 그리고 [드리프트 감지](#) 를 사용하여 자원이 AWS CloudFormation 외부에서 변경된 시점을 감지합니다.
 - b. 또한 [변경 세트](#) 를 사용하여 스택 업데이트의 의도가 변경 세트가 시작될 때 AWS CloudFormation이 취하는 작업과 일치하는지 확인합니다.
2. 이렇게 하면 [AWS CodePipeline](#) 에서 수동 승인 단계가 트리거되어 사전 프로덕션 환경에 대한 배포를 승인합니다.
3. 그리고 [AWS CodeDeploy AppSpec](#) 파일과 같은 배포 구성을 사용하여 배포 및 검증 단계를 정의합니다.
4. 해당하는 경우 [AWS CodeDeploy를 다른 AWS 서비스와 통합하거나](#) 또는 [AWS CodeDeploy를 파트너 제품 및 서비스와 통합](#) 단축할 수 있습니다.
5. [또한](#) Amazon CloudWatch, AWS CloudTrail 및 Amazon SNS 이벤트 알림을 사용하여 배포를 모니터링하세요.
6. 기능, 보안, 회귀, 통합 및 로드 테스트를 포함하여 배포 후 자동화된 테스트를 수행합니다.
7. [문제 해결](#) 배포 문제
8. 이전 단계에 대한 검증이 성공적으로 끝나면 프로덕션으로의 배포를 승인하는 수동 승인 워크플로가 시작됩니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- [OPS05-BP02 변경 사항 테스트 및 검증](#)

관련 문서:

- [AWS Builders' Library | 안전한 자동 배포 자동화 | 테스트 배포](#)
- [AWS 백서 | AWS에서의 지속적 통합 및 지속적 전달 사례](#)

- [The Story of Apollo - Amazon's Deployment Engine](#)
- [코드 전달 전에 AWS CodeDeploy를 로컬로 테스트 및 디버깅하는 방법](#)
- [Integrating Network Connectivity Testing with Infrastructure Deployment](#)

관련 동영상:

- [re:Invent 2020 | Testing software and systems at Amazon](#)

관련 예시:

- [자습서 | 유효성 검사 테스트를 통한 Amazon ECS 서비스 배포](#)

OPS06-BP03 안전한 배포 전략 채택

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

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

일반적인 안티 패턴:

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

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

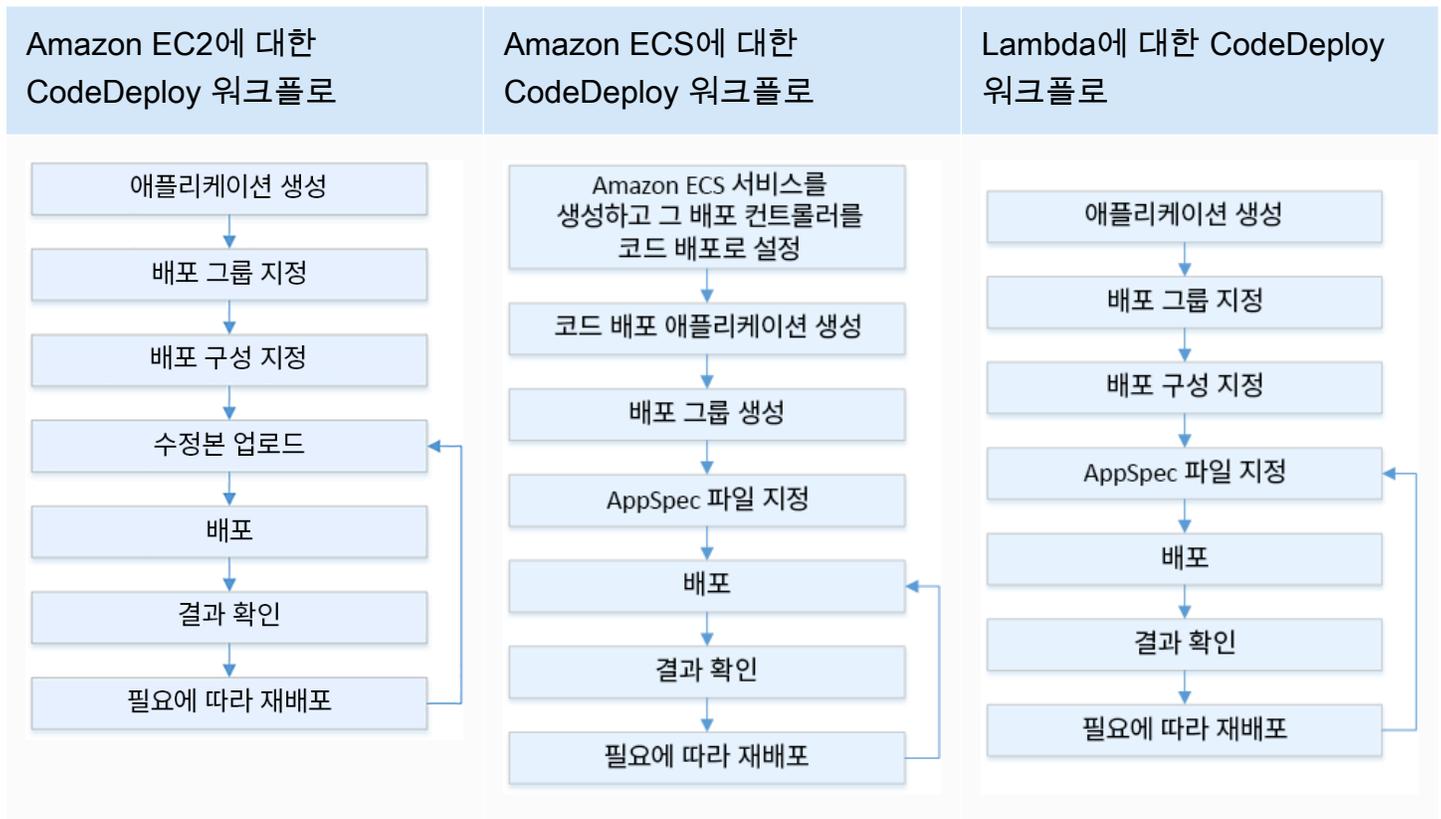
이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

고객 사례

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



구현 단계

1. 승인 워크플로를 사용하여 프로덕션 단계로 진입할 때 프로덕션 롤아웃 단계의 순서를 시작합니다.

2. 이때 [AWS CodeDeploy](#)과 같은 자동화된 배포 시스템을 사용하세요. AWS CodeDeploy [배포 옵션에는](#) EC2/온프레미스에 대한 인플레이스 배포와 EC2/온프레미스에 대한 블루/그린 배포, AWS Lambda 및 Amazon ECS(위의 워크플로 다이어그램 참조)를 포함합니다.
 - a. 해당하는 경우 [AWS CodeDeploy를 다른 AWS 서비스와 통합](#) 또는 [AWS CodeDeploy를 파트너 제품 및 서비스와 통합](#)합니다.
3. 그리고 [Amazon Aurora](#) 및 [Amazon RDS](#)와 같은 데이터베이스에는 블루/그린 배포를 사용합니다.
4. [배포 모니터링에는](#) Amazon CloudWatch, AWS CloudTrail 및 Amazon Simple Notification Service(Amazon SNS) 이벤트를 알림을 사용합니다.
5. 기능, 보안, 회귀, 통합 및 로드 테스트를 비롯한 자동화된 배포 후 테스트를 수행합니다.
6. [문제 해결](#): 배포 문제

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS05-BP02 변경 사항 테스트 및 검증](#)
- [OPS05-BP09 되돌릴 수 있는 작은 단위의 변경 내용 자주 적용](#)
- [OPS05-BP10 통합 및 배포 완전 자동화](#)

관련 문서:

- [AWS Builders' Library | 안전한 자동 배포 자동화 | 프로덕션 배포](#)
- [AWS Builders Library | 릴리스를 이끄는 CI/CD 파이프라인 | 안전한 자동 프로덕션 릴리스](#)
- [AWS 백서 | AWS에서의 지속적 통합 및 지속적 전달 사례 | 배포 방법](#)
- [AWS CodeDeploy 사용 설명서](#)
- [AWS CodeDeploy에서 배포 구성 사용](#)
- [API Gateway Canary 릴리스 배포 설정](#)
- [Amazon ECS 배포 유형](#)
- [Amazon Aurora 및 Amazon RDS 내 완전 관리형 블루/그린 배포](#)
- [AWS Elastic Beanstalk을 사용한 블루/그린 배포](#)

관련 동영상:

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

관련 예시:

- [AWS CodeDeploy에서 샘플 블루/그린 배포 시도](#)
- [워크숍 | Building CI/CD pipelines for Lambda canary deployments using AWS CDK](#)
- [워크숍 | Blue/Green and Canary Deployment for EKS and ECS](#)
- [워크숍 | Building a Cross-account CI/CD Pipeline](#)

OPS06-BP04 테스트 및 롤백 자동화

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

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

일반적인 안티 패턴:

- 시스템이 소규모 릴리스로 업데이트할 수 있는 방식으로 설계되지 않았습니다. 따라서 실패한 배포 중에 이러한 대량 변경 사항을 되돌리기가 어렵습니다.
- 배포 프로세스가 일련의 수동 단계로 구성되어 있습니다. 변경 사항을 워크로드에 배포한 후, 배포 후 테스트를 시작합니다. 테스트 후 워크로드가 작동하지 않으며 고객 연결이 끊어짐을 알게 됩니다. 그런 다음 이전 버전으로 롤백을 시작합니다. 이러한 모든 수동 단계는 전체 시스템 복구를 지연시키고 고객에게 장기적인 영향을 미칩니다.

- 애플리케이션에서 자주 사용되지 않는 기능에 대한 자동화된 테스트 사례를 개발하는 데 시간을 투자하여 자동화된 테스트 기능에 대한 투자 수익을 최소화했습니다.
- 릴리스가 서로 독립적인 애플리케이션, 인프라, 패치 및 구성 업데이트로 구성되어 있습니다. 하지만 모든 변경 사항을 한 번에 전달하는 단일 CI/CD 파이프라인이 있습니다. 한 구성 요소에 실패가 발생하면 모든 변경 사항을 되돌려야 하므로 롤백이 복잡하고 비효율적입니다.
- 팀이 스프린트 1에서 코딩 작업을 완료하고 스프린트 2 작업을 시작하지만 스프린트 3까지의 테스트는 계획에 포함되지 않았습니다. 그 결과, 자동화된 테스트를 통해 스프린트 2 산출물에 대한 테스트를 시작하기 전에 해결해야 했던 결함이 스프린트 1에서 드러났으며, 전체 릴리스가 지연되어 자동 테스트의 가치가 떨어졌습니다.
- 프로덕션 릴리스의 자동 회귀 테스트 사례는 완료되었지만 워크로드 상태를 모니터링하고 있지는 않습니다. 서비스 재시작 여부를 확인할 수 없으므로 롤백이 필요한지 또는 이미 발생했는지 확실하지 않습니다.

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

1. 요구 사항 계획부터 테스트 사례 개발, 도구 구성, 자동화된 테스트, 테스트 사례 종료에 이르기까지 테스트 프로세스의 각 단계를 정의하는 개발 수명 주기에 대한 테스트 수명 주기를 설정합니다.
 - a. 전체 테스트 전략을 바탕으로 워크로드별 테스트 접근 방식을 만듭니다.
 - b. 개발 수명 주기 전반에 걸쳐 적절한 경우 지속적 테스트 전략을 고려합니다.

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

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립](#)
- [OPS06-BP02 테스트 배포](#)

관련 문서:

- [AWS Builders Library | 배포 시 롤백 안전 보장](#)
- [AWS CodeDeploy로 재배포 및 배포 롤백](#)
- [AWS CloudFormation을 사용하여 배포를 자동화할 때의 8가지 모범 사례](#)

관련 예시:

- [Selenium, AWS Lambda, AWS Fargate \(Fargate\) 및 AWS 개발자 도구를 사용한 서버리스 UI 테스트](#)

관련 동영상:

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

운영 준비 상태 및 변경 관리

워크로드, 프로세스, 절차 및 직원의 운영 준비 상태를 평가하여 워크로드와 관련된 운영 위험을 파악합니다. 환경으로 이어지는 변경 흐름을 관리합니다.

수동 또는 자동화된 체크리스트를 비롯한 일관된 프로세스를 사용하여 워크로드 또는 변경에 대응하는 준비 여부를 확인해야 합니다. 이렇게 하면 문제 해결 계획을 세워야 하는 영역도 파악할 수 있습니다. 일상 활동을 문서화한 런북과 문제 해결 프로세스를 안내하는 플레이북이 제공됩니다. 비즈니스 가치 제공을 지원하고 변화와 관련된 위험을 완화하는 데 도움이 되는 변경 사항을 관리하는 메커니즘을 사용할 수 있습니다.

모범 사례

- [OPS07-BP01 직원의 역량 확보](#)

- [OPS07-BP02 일관된 방식으로 운영 준비 상태 검토](#)
- [OPS07-BP03 런북을 사용한 절차 수행](#)
- [OPS07-BP04 플레이북을 사용하여 문제 조사](#)
- [OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정](#)
- [OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 활성화](#)

OPS07-BP01 직원의 역량 확보

워크로드를 지원하기 위해 적절한 수의 숙련된 인력이 있는지 확인하는 메커니즘을 확보합니다. 워크로드를 구성하는 플랫폼과 서비스에 대해 교육을 받아야 합니다. 워크로드를 운영하는 데 필요한 지식을 제공합니다. 워크로드의 정상 작동을 지원하고 발생하는 인시던트 문제를 해결할 수 있도록 충분한 교육을 받은 직원이 있어야 합니다. 번아웃을 방지하기 위해 업무 대기 기간 및 휴가 기간 동안 다른 인력이 순환할 수 있도록 충분한 인원을 확보합니다.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

고객 사례

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

구현 단계

1. 대기 근무를 비롯하여 워크로드를 운영하고 지원하기에 적절한 수의 직원을 할당합니다.
2. 워크로드를 구성하는 소프트웨어 및 플랫폼에 대해 직원을 교육합니다.
 - a. [AWS Training and Certification](#) 페이지에는 AWS에 대한 교육 과정 라이브러리가 있습니다. 무료 및 유료의 온라인, 오프라인 과정을 제공합니다.
 - b. AWS 전문가로부터 배울 수 있는 [이벤트 및 웨비나](#)를 AWS가 주최합니다.
3. 운영 조건 및 워크로드 변화에 따라 팀 규모와 기술을 정기적으로 평가합니다. 운영 요구 사항에 맞게 팀 규모와 기술을 조정합니다.

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

리소스

관련 모범 사례:

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

관련 문서:

- [AWS 이벤트 및 웨비나](#)
- [AWS Training and Certification](#)

OPS07-BP02 일관된 방식으로 운영 준비 상태 검토

ORR(운영 준비 상태 검토)을 사용하여 워크로드를 운영할 수 있는지 검증할 수 있습니다. ORR은 팀에서 워크로드를 안전하게 운영할 수 있는지 검증할 수 있도록 Amazon에서 개발한 메커니즘입니다. ORR은 요구 사항의 체크리스트를 사용한 검토 및 검사 프로세스입니다. ORR은 팀이 자체 워크로드

를 인증하는 데 사용하는 셀프 서비스 경험입니다. ORR에는 다년간의 소프트웨어 구축을 통해 얻은 교훈을 바탕으로 한 모범 사례가 포함되어 있습니다.

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

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

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

고객 사례

몇 번의 프로덕션 인시던트 후 AnyCompany Retail은 ORR 프로세스를 구현하기로 했습니다. 이를 위해 모범 사례, 거버넌스 및 규정 준수 요구 사항, 그리고 중단으로부터 얻은 교훈을 통해 구성된 체크리스트를 구축했습니다. 새 워크로드를 시작하기 전에 ORR을 수행합니다. 모든 워크로드는 ORR 체크리스트에 추가되는 새로운 모범 사례 및 요구 사항을 통합하기 위해 모범 사례의 하위 집합이 포함된 연간 ORR을 수행합니다. 시간이 지나면서 AnyCompany Retail은 [AWS Config](#) 를 사용하여 일부 모범 사례를 탐지하고 ORR 프로세스의 속도를 높였습니다.

구현 단계

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

1. 보안, 운영 및 개발 담당자를 포함한 핵심 이해 관계자를 한 자리에 모읍니다.
2. 각 이해 관계자가 한 가지 이상의 요구 사항을 제공하도록 합니다. 첫 반복의 경우 항목의 수를 30개 이하로 제한합니다.
 - [부록 B: ORR 질문 예시](#) 는 ORR(운영 준비 상태 검토) 백서에 수록되어 있으며 시작 시 사용할 수 있는 샘플 질문이 포함되어 있습니다.
3. 요구 사항을 스프레드시트에 수집합니다.
 - 이때 [사용자 지정 렌즈](#) ([AWS Well-Architected Tool](#) 에 포함)를 사용하여 ORR을 개발하고 계정 및 AWS Organization 전체에서 이를 공유할 수 있습니다.
4. ORR을 수행할 하나의 워크로드를 식별합니다. 출시 전 워크로드나 내부 워크로드가 가장 좋습니다.

5. ORR 체크리스트를 실행하고 탐색 내용을 기록합니다. 완화 조치가 적용된 경우 탐색 결과가 좋지 않을 수 있습니다. 완화 조치가 부족한 탐색 결과에 대해서는 항목의 백로그에 이를 추가하고 시작 전에 구현합니다.
6. 시간이 지나는 동안 ORR 체크리스트에 모범 사례 및 요구 사항을 계속 추가합니다.

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

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

리소스

관련 모범 사례:

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

관련 문서:

- [AWS Control Tower - AWS Control Tower의 가이드라인](#)
- [AWS Well-Architected Tool - 사용자 지정 렌즈](#)
- [Adrian Hornsby의 운영 준비 상태 검토 템플릿](#)

- [ORR\(운영 준비 상태 검토\) 백서](#)

관련 동영상:

- [AWS Supports You | 효과적인 ORR\(운영 준비 상태 검토\) 구축](#)

관련 예시:

- [샘플 ORR\(운영 준비 상태 검토\) 렌즈](#)

관련 서비스:

- [AWS Config](#)
- [AWS Control Tower](#)
- [AWS Security Hub](#)
- [AWS Well-Architected Tool](#)

OPS07-BP03 런북을 사용한 절차 수행

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

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

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

원하는 결과: 팀에 워크로드 작업을 수행하기 위한 단계별 가이드 모음이 있습니다. 런북에는 원하는 결과, 필요한 도구, 권한 및 오류 처리 지침이 들어 있습니다. 런북이 중앙 위치(버전 관리 시스템)에 저

장되고 자주 업데이트됩니다. 예를 들어, 런북을 통해 팀은 애플리케이션 경보, 운영 문제 및 계획된 수명 주기 이벤트 중에 중요한 계정에 대한 AWS Health 이벤트를 모니터링하고, 전달하고, 이에 대응할 수 있습니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

텍스트 런북은 조직이 성숙함에 따라 자동화되어야 합니다. [AWS Systems Manager Automation](#)과 같은 서비스를 사용하면 플랫폼 텍스트를 워크로드에 대해 실행할 수 있는 자동화로 변환할 수 있습니다. 이러한 자동화를 이벤트에 대응하여 실행해 워크로드 유지를 위한 운영 부담을 줄일 수 있습니다. AWS Systems Manager Automation은 자동화 런북을 보다 쉽게 만들 수 있는 로우코드 [시각적 디자인 경험](#)도 제공합니다.

고객 사례

AnyCompany Retail은 소프트웨어를 배포하는 중 데이터베이스 스키마 업데이트를 수행해야 합니다. 클라우드 운영 팀은 데이터베이스 관리 팀과 협력하여 이러한 변경 사항을 수동으로 배포하기 위한 런

북을 빌드했습니다. 이 런북에는 프로세스의 각 단계를 체크리스트 형식으로 나열되어 있습니다. 또한 문제 발생 시 오류 처리에 대한 섹션이 포함되어 있습니다. 팀은 내부 Wiki에 다른 런북과 함께 이 런북을 게시했습니다. 클라우드 운영 팀은 향후 스프린트에서 런북을 자동화할 계획입니다.

구현 단계

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

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last Updated | Escalation POC |
|-----|-----|-----|-----|-----|-----|-----|
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

1. 기존 문서 리포지토리 또는 Wiki가 없는 경우 버전 관리 시스템에서 새로운 버전 관리 리포지토리를 생성합니다.
2. 런북이 없는 프로세스를 파악합니다. 이상적인 프로세스는 반규칙적으로 수행되며 단계 수가 적고 장애 영향이 적은 프로세스입니다.
3. 문서 리포지토리에서 템플릿을 사용하여 새로운 Markdown 문서 초안을 작성합니다. 런북 제목 및 런북 정보 아래의 필수 필드를 입력합니다.
4. 첫 번째 단계부터 시작하여 런북의 단계 부분을 채웁니다.
5. 팀원에게 런북을 제공하고 런북을 사용하여 단계를 확인하도록 합니다. 누락된 부분이 있거나 명확히 설명해야 할 부분이 있다면 런북을 업데이트합니다.
6. 내부 문서 저장소에 런북을 게시합니다. 게시한 다음, 팀 및 다른 이해관계자에게 알립니다.
7. 시간이 흐르면 런북 라이브러리를 빌드합니다. 라이브러리가 커지면 런북 자동화 작업을 시작합니다.

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

리소스

관련 모범 사례:

- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)
- [OPS07-BP04 플레이북을 사용하여 문제 조사](#)
- [OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용](#)
- [OPS10-BP02 알림별 프로세스 마련](#)
- [OPS11-BP04 지식 관리 수행](#)

관련 문서:

- [AWS Well-Architected Framework: Concepts: Runbook development](#)
- [Achieving Operational Excellence using automated playbook and runbook](#)
- [AWSSystems Manager: 런북을 사용한 작업](#)
- [AWS 대규모 마이그레이션을 위한 마이그레이션 플레이북 - 작업 4: 마이그레이션 런북 개선](#)
- [AWS Systems Manager 자동화 런북을 사용하여 운영 작업 해결](#)

관련 동영상:

- [AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response](#)
- [AWS에서 IT 운영을 자동화하는 방법 | Amazon Web Services](#)
- [AWS Systems Manager로 스크립트 통합](#)

관련 예시:

- [Well-Architected 실습: 플레이북 및 런북으로 운영 자동화](#)
- [AWS 블로그 게시물: 운영 효율성을 위한 클라우드 자동화 사례 구축: AWS Managed Services의 모범 사례](#)
- [AWS Systems Manager: 자동화 시연](#)
- [AWS Systems Manager: 최신 스냅샷 런북에서 루트 볼륨 복원](#)
- [Jupyter Notebook 및 CloudTrail Lake를 사용하여 AWS 인시던트 대응 런북 빌드](#)
- [Gitlab - Runbooks](#)
- [Rubix - Jupyter Notebook에서 런북을 빌드하기 위한 Python 라이브러리](#)

- [Document Builder를 사용하여 사용자 지정 런북 만들기](#)

관련 서비스:

- [AWS Systems Manager Automation](#)

OPS07-BP04 플레이북을 사용하여 문제 조사

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

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

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

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

워크로드에 발생하는 일반적인 인시던트를 리스팅하여 플레이북의 구축을 시작할 수 있습니다. 시작하려면 위험성이 낮고 근본 원인이 몇 가지 문제로 좁혀진 인시던트에 대한 플레이북을 선택합니다. 간단한 시나리오에 대한 플레이북을 갖춘 후에는 근본 원인이 잘 알려지지 않았고 위험성이 더 높은 시나리오로 넘어가도록 합니다.

텍스트 플레이북은 조직이 성숙해지면 자동화되어야 합니다. [AWS Systems Manager Automation](#)과 같은 서비스를 사용하면 플랫폼 텍스트를 자동화로 변환할 수 있습니다. 이러한 자동화를 워크로드에 대해 실행함으로써 조사 속도를 높일 수 있습니다. 이벤트에 대한 대응으로 이러한 자동화를 활성화하여 인시던트를 발견하고 해결하는 데 걸리는 평균 시간을 단축할 수 있습니다.

고객은 [AWS Systems Manager Incident Manager](#)를 사용하여 인시던트에 대응할 수 있습니다. 이 서비스는 인시던트를 분류하고, 복구 및 완화 과정에서 이해관계자에게 이를 알리고, 인시던트 전반에서 협업할 수 있는 단일 인터페이스를 제공합니다. 또한 AWS Systems Manager Automation을 사용하여 탐지 및 복구 속도를 높입니다.

고객 사례

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

구현 단계

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

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

1. 기존 문서 리포지토리 또는 Wiki가 없는 경우 버전 관리 시스템에서 플레이북에 대한 새로운 버전 관리 리포지토리를 생성합니다.
2. 조사가 필요한 일반적인 문제를 파악합니다. 근본 원인이 몇 가지 문제로 한정되어 있고 해결 방법의 위험성이 낮은 시나리오여야 합니다.
3. Markdown 템플릿을 사용하여 플레이북 이름 섹션과 플레이북 정보 아래의 필드를 작성합니다.
4. 문제 해결 단계를 작성합니다. 수행해야 하는 작업 또는 조사해야 하는 영역을 최대한 명확하게 작성합니다.
5. 팀원에게 플레이북을 전달하여 살펴보고 확인할 수 있도록 합니다. 누락되거나 명확하지 않은 사항이 있는 경우 플레이북을 업데이트합니다.
6. 문서 리포지토리에 플레이북을 게시하고 팀과 모든 이해관계자에게 이를 알립니다.
7. 더 많은 플레이북을 추가할수록 이 플레이북 라이브러리는 더 발전하게 됩니다. 여러 플레이북이 있다면 플레이북의 자동화와 동기화를 유지할 수 있도록 AWS Systems Manager Automation과 같은 도구를 사용하여 자동화를 시작합니다.

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

리소스

관련 모범 사례:

- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)
- [OPS07-BP03 런북을 사용한 절차 수행](#)
- [OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용](#)
- [OPS10-BP02 알림별 프로세스 마련](#)
- [OPS11-BP04 지식 관리 수행](#)

관련 문서:

- [AWS Well-Architected Framework: Concepts: Playbook development](#)
- [Achieving Operational Excellence using automated playbook and runbook](#)
- [AWSSystems Manager: 런북을 사용한 작업](#)
- [AWS Systems Manager 자동화 런북을 사용하여 운영 작업 해결](#)

관련 동영상:

- [AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response \(SEC318-R1\)](#)
- [AWS Systems Manager Incident Manager - AWS Virtual Workshops](#)
- [AWS Systems Manager로 스크립트 통합](#)

관련 예시:

- [AWS Customer Playbook Framework](#)
- [AWS Systems Manager: 자동화 시연](#)
- [Jupyter Notebook 및 CloudTrail Lake를 사용하여 AWS 인시던트 대응 런북 빌드](#)
- [Rubix - Jupyter Notebook에서 런북을 빌드하기 위한 Python 라이브러리](#)
- [Document Builder를 사용하여 사용자 지정 런북 만들기](#)
- [Well-Architected 실습: 플레이북 및 런북으로 운영 자동화](#)
- [Well-Architected Labs: Incident response playbook with Jupyter](#)

관련 서비스:

- [AWS Systems Manager Automation](#)
- [AWS Systems Manager Incident Manager](#)

OPS07-BP05 정보에 입각하여 시스템 및 변경 사항 배포 결정

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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

고객 사례

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

구현 단계

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

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

리소스

관련 모범 사례:

- [OPS01-BP03 거버넌스 요구 사항 평가](#) - 거버넌스 요구 사항은 변경 사항 배포 여부를 결정하는 핵심 요소입니다.
- [OPS06-BP01 변경이 적절하지 못한 경우에 대한 계획 수립](#) - 배포 실패를 완화하기 위한 계획을 수립하고 사전 분석을 사용하여 이를 검증합니다.
- [OPS06-BP02 테스트 배포](#) - 프로덕션 결함을 줄이기 위해 배포 전에 모든 소프트웨어 변경 사항을 적절하게 테스트해야 합니다.
- [OPS07-BP01 직원의 역량 확보](#) - 워크로드를 지원할 수 있는 충분한 교육을 받은 인력을 확보하는 것은 정보에 입각한 시스템 변경 사항 배포 결정을 내리는 데 필수적입니다.

관련 문서:

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

OPS07-BP06 프로덕션 워크로드에 대한 지원 플랜 활성화

프로덕션 워크로드가 의존하는 모든 소프트웨어 및 서비스에 대한 지원을 활성화합니다. 프로덕션 서비스 수준 요구 사항을 충족하는 적절한 지원 수준을 선택합니다. 이러한 종속성 지원 플랜은 서비스 중단 또는 소프트웨어 문제가 생긴 경우에 필요합니다. 모든 서비스 및 소프트웨어 공급업체에 대한 지

원 플랜과 지원 요청 방법을 문서화합니다. 지원 담당 연락처가 최신 상태로 유지되는지 확인하는 메커니즘을 구현합니다.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

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

고객 사례

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

구현 단계

1. 조직의 이해관계자와 협력하여 워크로드가 의존하는 소프트웨어 및 서비스 공급업체를 식별합니다. 이러한 종속성을 문서화합니다.
2. 워크로드에 대한 서비스 수준 요구 사항을 결정합니다. 이에 맞는 지원 플랜을 선택합니다.
3. 상용 소프트웨어 및 서비스의 경우 공급업체와 함께 지원 플랜을 수립합니다.
 - a. 모든 프로덕션 계정에 대해 AWS Business Support 이상을 구독하면 AWS Support로부터 더 빠르게 응답을 받을 수 있으므로, AWS Business Support 이상을 구독할 것을 강력히 권장합니다. 프리미엄 지원을 받지 못하는 경우 AWS Support의 도움이 필요한 문제를 처리할 수 있는 실행 플랜을 마련해야 합니다. AWS Support는 사용자가 성능을 최적화하고, 비용을 절감하고, 혁신 속도를 높이는 데 적극적으로 도움이 될 수 있도록 설계된 다양한 도구 및 기술, 인력, 프로그램을 제공합니다. AWS Business Support는 AWS Trusted Advisor 및 AWS Personal Health Dashboard에 액세스하고 응답 시간을 단축하는 등의 추가적인 이점을 제공합니다.
4. 지식 관리 도구에 지원 플랜을 문서화합니다. 지원 요청 방법, 지원 사례가 접수될 경우 통지 대상, 인시던트 발생 시의 에스컬레이션 방법을 포함합니다. Wiki는 지원 프로세스나 연락처의 변경 사항을 알게 된 누구나 문서를 필요에 따라 업데이트할 수 있도록 지원하는 좋은 메커니즘입니다.

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

리소스

관련 모범 사례:

- [OPS02-BP02 프로세스 및 절차의 소유자 식별](#)

관련 문서:

- [AWS Support 플랜](#)

관련 서비스:

- [AWS Business Support](#)
- [AWS Enterprise Support](#)

운영

성공은 직접 정의한 지표를 기준으로 측정된 비즈니스 성과를 달성했음을 의미합니다. 워크로드 및 운영 상태를 이해하면 조직 및 비즈니스 성과가 지금 위험한지, 아니면 앞으로 위험해질 것인지를 파악하고 적절히 대응할 수 있습니다.

성공하려면 다음을 수행할 수 있어야 합니다.

주제

- [워크로드 관찰성 활용](#)
- [운영 상태 파악](#)
- [이벤트 대응](#)

워크로드 관찰성 활용

관찰성을 활용하여 워크로드 상태를 최적화하세요. 관련 지표, 로그, 추적을 활용하여 워크로드 성능을 종합적으로 파악하고 문제를 효율적으로 해결하세요.

관찰성을 통해 의미 있는 데이터에 집중하고 워크로드의 상호 작용과 결과를 이해할 수 있습니다. 필수 인사이트에 집중하고 불필요한 데이터를 제거함으로써 워크로드 성능을 이해하기 위한 간단한 접근 방식을 유지할 수 있습니다.

데이터를 수집하는 것뿐만 아니라 데이터를 올바르게 해석하는 것도 중요합니다. 명확한 기준을 정의하고, 적절한 경고 임계값을 설정하고, 편차를 적극적으로 모니터링합니다. 특히 다른 데이터와 상관관계가 있는 경우 주요 지표의 변화는 특정 문제 영역을 정확히 찾아낼 수 있습니다.

관찰성을 사용하면 잠재적 문제를 더 잘 예측하고 해결하여 워크로드가 원활하게 운영되고 비즈니스 요구 사항을 충족할 수 있습니다.

AWS는 다음과 같은 특정 도구를 제공합니다. [Amazon CloudWatch](#) 모니터링 및 로깅용 [AWS X-Ray](#) 분산 추적용. 이러한 서비스는 다양한 AWS 리소스와 손쉽게 통합되므로 효율적인 데이터 수집이 가능하고, 사전 정의된 임계값을 기반으로 알림을 설정하고, 대시보드에 데이터를 표시하여 쉽게 해석할 수 있습니다. 이러한 통찰력을 활용하면 운영 목표에 부합하는 정보에 입각한 데이터 기반 결정을 내릴 수 있습니다.

모범 사례

- [OPS08-BP01 워크로드 지표 분석](#)
- [OPS08-BP02 워크로드 로그 분석](#)
- [OPS08-BP03 워크로드 추적 데이터 분석](#)
- [OPS08-BP04 실행 가능한 알림 생성](#)
- [OPS08-BP05 대시보드 만들기](#)

OPS08-BP01 워크로드 지표 분석

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

Amazon CloudWatch 같은 도구를 활용하여 지표 분석을 수행합니다. AWS Cost Anomaly Detection 및 Amazon DevOps Guru 같은 AWS 서비스는 특히 정적 임계값을 알 수 없거나 행동 패턴이 이상 탐지에 더 적합한 경우 이상을 탐지하는 데 사용할 수 있습니다.

구현 단계

1. 분석 및 검토: 워크로드 지표를 정기적으로 검토하고 해석하세요.
 - a. 순전히 기술적인 지표보다 비즈니스 성과 지표를 우선시하세요.
 - b. 데이터의 급증, 하락 또는 패턴의 중요성을 이해하세요.
2. 활용 Amazon CloudWatch: 중앙 집중식 보기 및 심층 분석에 Amazon CloudWatch를 사용합니다.
 - a. 지표를 시각화하고 시간 경과에 따라 비교하도록 CloudWatch 대시보드를 구성하세요.
 - b. 또한 [CloudWatch의 백분위수를 사용하면](#) 지표 분포를 명확하게 파악하여 SLA를 정의하고 이상치를 이해하는 데 도움이 될 수 있습니다.
 - c. 애플리케이션 레이어를 통해 트래픽을 추적하고 구성 요소와 종속 요소 간 지연 시간을 파악할 수 있도록 [AWS Cost Anomaly Detection](#) 정적 임계값에 의존하지 않고 비정상적 패턴을 식별할 수 있습니다.
 - d. 호출 오류율, 대기 시간에 대한 서비스 수준 목표, 대기 시간 이상값에 대한 [CloudWatch계정 간 관찰 가능성](#) 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.
 - e. 또한 [CloudWatch 지표 인사이트를 사용하여](#) 계정 및 리전 전반의 지표 데이터를 쿼리하고 분석하여 추세와 이상 현상을 식별합니다.
 - f. CloudWatch 지표 수식을 [적용하여](#) 지표를 변환, 집계 또는 계산을 수행하여 심층적인 통찰력을 확보할 수 있습니다.
3. Amazon DevOps Guru 채택: 기계 학습으로 강화된 이상 탐지 기능 [Amazon DevOps Guru을 통합하여](#) 서버리스 애플리케이션의 운영 문제의 초기 징후를 식별하고 고객에게 영향을 미치기 전에 문제를 해결하세요.
4. 인사이트를 기반으로 최적화: 지표 분석을 기반으로 정보에 입각한 결정을 내려 워크로드를 조정하고 개선하세요.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)

관련 문서:

- [Wheel 블로그 - 지표를 지속적으로 검토하는 것의 중요성 강조](#)
- [백분위수는 중요합니다](#)
- [AWS Cost Anomaly Detection 사용](#)
- [CloudWatch 계정 간 관찰 가능성](#)
- [지표 인사이트로 CloudWatch 지표를 쿼리하세요](#)

관련 동영상:

- [에서 계정 간 오피저버빌리티 활성화 Amazon CloudWatch](#)
- [Amazon DevOps Guru 소개](#)
- [AWS Cost Anomaly Detection를 사용하여 지표를 지속적으로 분석합니다.](#)

관련 예시:

- [One Observability Workshop](#)
- [AIOps를 사용하여 운영 인사이트 확보 Amazon DevOps Guru](#)

OPS08-BP02 워크로드 로그 분석

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 운영 병목 현상, 보안 위협 및 기타 잠재적 문제를 사전에 식별합니다.
- 지속적인 애플리케이션 최적화를 위해 로그 데이터를 효율적으로 활용합니다.

- 애플리케이션 동작에 대한 이해도를 높여 디버깅 및 문제 해결을 지원합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

[Amazon CloudWatch Logs](#)는 로그 분석을 위한 강력한 도구입니다. CloudWatch Logs Insights 및 Contributor Insights와 같은 통합 기능을 사용하면 로그에서 의미 있는 정보를 직관적이고 효율적으로 도출할 수 있습니다.

구현 단계

1. CloudWatch Logs 설정: 로그를 CloudWatch Logs로 전송할 애플리케이션 및 서비스를 구성합니다.
2. 로그 이상 탐지 사용: [Amazon CloudWatch Logs 이상 탐지](#)를 활용하여 비정상적인 로그 패턴을 자동으로 식별하고 알립니다. 이 도구를 사용하면 로그의 이상 현상을 사전에 관리하고 잠재적 문제를 조기에 발견할 수 있습니다.
3. CloudWatch Logs Insights 설정: [CloudWatch Logs Insights](#)를 사용하여 대화형 방식으로 로그 데이터를 검색하고 분석합니다.
 - a. 쿼리를 만들어 패턴을 추출하고, 로그 데이터를 시각화하고, 실행 가능한 인사이트를 도출합니다.
 - b. [CloudWatch Logs Insights 패턴 분석](#)을 사용하여 빈번한 로그 패턴을 분석하고 시각화합니다. 이 기능은 로그 데이터의 일반적인 운영 추세와 잠재적 이상값을 이해하는 데 도움이 됩니다.
 - c. [CloudWatch Logs 비교\(diff\)](#)를 사용하여 서로 다른 기간 간 또는 여러 로그 그룹 간의 차이 분석을 수행합니다. 이 기능을 사용하여 변경 사항을 정확히 찾아내고 시스템 성능 또는 동작에 미치는 영향을 평가할 수 있습니다.
4. Live Tail을 통한 실시간 로그 모니터링: [Amazon CloudWatch Logs Live Tail](#)을 사용하여 로그 데이터를 실시간으로 확인합니다. 애플리케이션의 운영 활동이 발생할 때 이를 적극적으로 모니터링할 수 있으므로 시스템 성능 및 잠재적 문제를 즉시 파악할 수 있습니다.
5. Contributor Insights 활용: [CloudWatch Contributor Insights](#)를 사용하여 IP 주소 또는 사용자 에이전트와 같은 높은 카디널리티 차원에서 볼륨이 높은 항목을 식별할 수 있습니다.
6. CloudWatch Logs 지표 필터 구현: [CloudWatch Logs 지표 필터](#)를 구성하여 로그 데이터를 실행 가능한 지표로 변환합니다. 이를 통해 경보를 설정하거나 패턴을 추가로 분석할 수 있습니다.
7. [CloudWatch 교차 계정 관찰성](#) 구현: 한 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.
8. 정기적 검토 및 개선: 정기적으로 로그 분석 전략을 검토하여 모든 관련 정보를 캡처하고 애플리케이션 성능을 지속적으로 최적화합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS04-BP02 애플리케이션 원격 측정 구현](#)
- [OPS08-BP01 워크로드 지표 분석](#)

관련 문서:

- [CloudWatch Logs Insights를 사용한 로그 데이터 분석](#)
- [CloudWatch Contributor Insights 사용](#)
- [CloudWatch 로그 지표 필터 생성 및 관리](#)

관련 동영상:

- [CloudWatch Logs Insights를 사용한 로그 데이터 분석](#)
- [Use CloudWatch Contributor Insights to Analyze High-Cardinality Data](#)

관련 예시:

- [CloudWatch Logs 샘플 쿼리](#)
- [One Observability Workshop](#)

OPS08-BP03 워크로드 추적 데이터 분석

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

[AWS X-Ray](#)는 추적 데이터 분석을 위한 포괄적인 제품군을 제공하여 서비스 상호 작용에 대한 전체적인 보기를 제공하고, 사용자 활동을 모니터링하고, 성능 문제를 탐지합니다. ServiceLens, X-Ray Insights, X-Ray Analytics 및 Amazon DevOps Guru와 같은 기능은 추적 데이터에서 더 심도 깊은 실행 가능한 인사이트를 얻을 수 있습니다.

구현 단계

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

1. AWS X-Ray 통합: 애플리케이션에 X-Ray를 통합하여 추적 데이터를 캡처할 수 있도록 합니다.
2. X-Ray 지표 분석: [서비스 맵](#)으로 애플리케이션 상태를 모니터링하여 지연 시간, 요청률, 장애율, 응답 시간 분포와 같은 X-Ray 추적에서 파생된 지표를 자세히 살펴봅니다.
3. ServiceLens 사용: [ServiceLens 맵](#)을 활용하여 서비스 및 애플리케이션의 관찰성을 향상합니다. 이를 통해 추적, 지표, 로그, 경보 및 기타 건강 정보를 통합적으로 볼 수 있습니다.
4. X-Ray Insights 활성화:
 - a. [X-Ray Insights](#)를 켜면 추적에서 이상 징후를 자동으로 탐지할 수 있습니다.
 - b. 인사이트를 검토하여 패턴을 정확히 찾아내고 장애율 또는 지연 시간 증가와 같은 근본 원인을 파악합니다.
 - c. 인사이트 타임라인을 참조하여 탐지된 문제를 시간순으로 분석합니다.

5. X-Ray Analytics 사용: [X-Ray Analytics](#)를 사용하면 추적 데이터를 철저히 탐색하고, 패턴을 정확히 찾아내며, 인사이트를 추출할 수 있습니다.
6. X-Ray에서 그룹 사용: X-Ray에서 그룹을 생성하여 긴 지연 시간과 같은 기준에 따라 추적을 필터링하여 보다 표적화된 분석을 수행할 수 있습니다.
7. Amazon DevOps Guru 통합: [Amazon DevOps Guru](#)를 통해 추적에서 운영 이상 징후를 찾아내는 기계 학습 모델의 이점을 활용합니다.
8. CloudWatch Synthetics 사용: [CloudWatch Synthetics](#)를 사용하여 엔드포인트 및 워크플로를 지속적으로 모니터링하기 위한 canary를 만듭니다. 이러한 canary를 X-Ray와 통합하여 테스트 대상 애플리케이션의 심층 분석을 위한 추적 데이터를 제공할 수 있습니다.
9. 실제 사용자 모니터링(RUM) 사용: [AWS X-Ray 및 CloudWatch RUM](#)을 사용하면 애플리케이션의 최종 사용자부터 시작하여 다운스트림 AWS 관리형 서비스까지 요청 경로를 분석하고 디버그할 수 있습니다. 이를 통해 최종 사용자에게 영향을 미치는 지연 추세 및 오류를 식별할 수 있습니다.
10. 로그와의 상관관계 파악: 애플리케이션 동작을 세부적으로 파악할 수 있도록 X-Ray 추적 보기 내에서 [추적 데이터와 관련 로그](#)의 상관관계를 분석합니다. 이렇게 하면 추적된 트랜잭션과 직접 관련된 로그 이벤트를 볼 수 있습니다.
11. [CloudWatch 교차 계정 관찰성](#) 구현: 한 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS08-BP01 워크로드 지표 분석](#)
- [OPS08-BP02 워크로드 로그 분석](#)

관련 문서:

- [ServiceLens를 사용하여 애플리케이션 상태 모니터링](#)
- [X-Ray Analytics을 통한 추적 데이터 탐색](#)
- [X-Ray Insights을 통한 트레이스의 이상 징후 감지](#)
- [CloudWatch Synthetics를 사용한 지속적인 모니터링](#)

관련 동영상:

- [Amazon CloudWatch Synthetics와 AWS X-Ray를 사용한 애플리케이션 분석 및 디버깅](#)
- [AWS X-Ray Insights 사용](#)

관련 예시:

- [One Observability Workshop](#)
- [AWS Lambda를 사용하여 X-Ray 구현하기](#)
- [CloudWatch Synthetics Canary 템플릿](#)

OPS08-BP04 실행 가능한 알림 생성

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

구현 단계

1. 핵심 성과 지표(KPI) 결정: 애플리케이션의 KPI를 식별합니다. 알림을 이러한 KPI와 연계하여 비즈니스에 미치는 영향을 정확하게 반영해야 합니다.
2. 이상 탐지 구현:
 - Amazon CloudWatch 이상 탐지 사용: 비정상적인 패턴을 자동으로 탐지하도록 [Amazon CloudWatch 이상 탐지](#) 기능을 설정하면 실제 이상 징후가 있을 때만 알림을 생성할 수 있습니다.
 - AWS X-Ray Insights 사용:
 - a. [X-Ray Insights](#)를 설정하여 추적 데이터에서 이상 징후를 탐지합니다.
 - b. [X-Ray Insights](#)가 탐지된 문제에 대해 알림을 받도록 알림을 구성합니다.
 - Amazon DevOps Guru와 통합:
 - a. [Amazon DevOps Guru](#)의 기계 학습 기능을 활용하여 기존 데이터로 운영 이상 징후를 탐지합니다.
 - b. DevOps Guru에서 [알림 설정](#)으로 이동하여 이상 알림을 설정합니다.
3. 실행 가능한 알림 구현: 즉각적인 조치를 취할 수 있도록 적절한 정보를 제공하는 알림을 설계합니다.
 1. [Amazon EventBridge 규칙을 사용하여 AWS Health 이벤트를](#) 모니터링하거나 AWS Health API와 프로그래밍 방식으로 통합하여 AWS Health 이벤트를 수신할 때 작업을 자동화합니다. 이러한 작업은 계획된 모든 수명 주기 이벤트 메시지를 채팅 인터페이스로 보내는 것과 같은 일반적인 작업이거나 IT 서비스 관리 도구에서 워크플로를 시작하는 것과 같은 구체적인 작업일 수 있습니다.
4. 알림 피로 감소: 중요하지 않은 알림을 최소화합니다. 대수롭지 않은 알림으로 팀이 부담을 느끼면 중요한 문제를 감독하지 못할 수 있고 결과적으로 알림 메커니즘의 전반적인 효율성이 떨어질 수 있습니다.
5. 복합 경고 설정: [Amazon CloudWatch 복합 경고](#)를 사용하여 여러 경보를 통합합니다.
6. 알림 도구와 통합: [Ops Genie](#) 및 [PagerDuty](#)와 같은 도구를 통합합니다.
7. AWS Chatbot 활용: [AWS Chatbot](#)을 통합하여 Amazon Chime, Microsoft Teams, Slack에 알림을 전달합니다.
8. 로그 기반 알림: CloudWatch에서 [로그 지표 필터](#)를 사용하여 특정 로그 이벤트를 기반으로 경보를 생성합니다.

9. 검토 및 반복: 정기적으로 알림 구성을 재검토하고 수정합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

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

관련 문서:

- [Amazon CloudWatch 경보 사용](#)
- [복합 경보 생성](#)
- [이상 탐지를 기반으로 CloudWatch 경보 생성](#)
- [DevOps Guru Notifications](#)
- [X-ray insights notifications](#)
- [AWS 상호작용형 ChatOps로 리소스 모니터링, 운영 및 문제 해결](#)
- [Amazon CloudWatch 통합 가이드 | PagerDuty](#)
- [OpsGenie와 Amazon CloudWatch 통합](#)

관련 동영상:

- [Create Composite Alarms in Amazon CloudWatch](#)
- [AWS Chatbot Overview](#)
- [AWS On Air ft. Mutative Commands in AWS Chatbot](#)

관련 예시:

- [Amazon CloudWatch을 통해 클라우드에서의 경보, 인시던트 관리 및 문제 해결](#)
- [자습서: AWS Chatbot에 알림을 보내는 Amazon EventBridge 규칙 만들기](#)
- [One Observability Workshop](#)

OPS08-BP05 대시보드 만들기

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

비즈니스 중심 대시보드

비즈니스 KPI에 맞게 조정된 대시보드는 다양한 이해관계자를 참여시킵니다. 이러한 개인은 시스템 지표에 관심이 없을 수도 있지만 이러한 수치가 비즈니스에 미치는 영향을 이해하는 데 관심이 있습니다.

비즈니스 중심 대시보드를 사용하면 모니터링 및 분석되는 모든 기술 및 운영 지표가 중요한 비즈니스 목표와 동기화됩니다. 이러한 정렬은 모든 사람이 무엇이 필수적이고 무엇이 아닌지에 대해 동일한 이해를 가질 수 있도록 명확성을 제공합니다. 또한 비즈니스 KPI를 강조하는 대시보드는 실행 가능성이 더 높은 경향이 있습니다. 이해 관계자는 운영 상태, 주의가 필요한 영역, 비즈니스 성과에 미치는 잠재적 영향을 빠르게 이해할 수 있습니다.

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

Amazon CloudWatch 대시보드는 CloudWatch 콘솔의 맞춤형 홈페이지로, 다른 AWS 리전과 계정에 분산되어 있는 리소스까지 포함하여 모든 리소스를 단일 보기에서 모니터링하는 데 이용할 수 있습니다.

구현 단계

1. 기본 대시보드 생성: [CloudWatch에서 새 대시보드](#)를 만들고 이 대시보드를 설명하는 이름을 지정합니다.
2. Markdown 위젯 사용: 지표를 자세히 살펴보기 전에 [Markdown 위젯을 사용](#)하여 대시보드 상단에 텍스트로 컨텍스트를 추가합니다. 여기에는 대시보드에서 다루는 내용, 표시된 지표의 중요성이 설명되어야 하며 다른 대시보드 및 문제 해결 도구에 대한 링크도 포함될 수 있습니다.
3. 대시보드 변수 생성: 동적이고 유연한 대시보드 보기를 위해 적절한 곳에 [대시보드 변수를 추가](#)합니다.
4. 지표 위젯 생성: 애플리케이션이 내보내는 다양한 지표를 시각화하도록 [지표 위젯을 추가](#)하고 시스템 상태 및 비즈니스 결과가 효과적으로 표시되도록 위젯을 조정합니다.
5. Log Insights 쿼리: [CloudWatch Log Insights](#)를 활용하여 로그에서 실행 가능한 지표를 도출하고 대시보드에 이러한 인사이트를 표시합니다.
6. 경보 설정: [CloudWatch 경보](#)를 대시보드에 통합하여 임계값을 위반하는 모든 지표를 빠르게 확인합니다.
7. Contributor Insights 사용: [CloudWatch Contributor Insights](#)를 통합하여 카디널리티가 높은 필드를 분석하고 리소스를 가장 많이 사용하는 항목을 더 명확하게 파악할 수 있습니다.
8. 사용자 지정 위젯 설계: 표준 위젯으로는 충족되지 않는 특정 요구 사항에 대해서는 [사용자 지정 위젯](#)을 만드는 것이 좋습니다. 사용자 지정 위젯은 다양한 데이터 소스에서 데이터를 가져오거나 고유한 방식으로 데이터를 표현할 수 있습니다.
9. AWS Health Dashboard 사용: [AWS Health Dashboard](#)를 사용하여 계정 상태, 이벤트, 서비스 및 리소스에 영향을 미칠 수 있는 향후 변경 사항에 대한 심층적인 인사이트를 얻습니다. 또한 AWS

Organizations 내의 상태 이벤트를 한 곳에서 확인하거나 사용자 지정 대시보드를 만들 수 있습니다 (자세한 내용은 관련 예시 참조).

10.반복 및 개선: 애플리케이션이 발전함에 따라 정기적으로 대시보드를 다시 방문하여 관련성을 확인합니다.

리소스

관련 모범 사례:

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

관련 문서:

- [운영 가시성을 위한 대시보드 구축](#)
- [Amazon CloudWatch 대시보드 사용](#)

관련 동영상:

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

관련 예시:

- [One Observability Workshop](#)
- [Amazon CloudWatch 애플리케이션 모니터링](#)
- [AWS Health Events Intelligence Dashboards and Insights](#)
- [Visualize AWS Health events using Amazon Managed Grafana](#)

운영 상태 파악

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

팀이 운영 상태를 쉽게 파악할 수 있어야 합니다. 운영 팀의 비즈니스 목표를 정의하고, 이를 반영하는 핵심 성과 지표를 식별한 다음, 이를 사용하여 운영 결과를 기반으로 지표를 개발하여 유용한 통찰력을 확보해야 합니다. 이러한 지표를 사용하여 비즈니스 및 기술적 시각이 반영된 대시보드를 구현해야 합니다. 팀원들은 이러한 대시보드를 통해 정보를 파악하여 적절한 결정을 내릴 수 있습니다.

AWS에서는 운영 로그를 더욱 쉽게 취합하고 분석할 수 있습니다. 이를 통해 지표를 생성하고 운영 상태를 확인하며, 시간이 지남에 따라 운영으로부터 인사이트를 확보할 수 있습니다.

모범 사례

- [OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정](#)
- [OPS09-BP02 상태 및 추세를 전달하여 운영에 대한 가시성 확보](#)
- [OPS09-BP03 운영 지표 검토 및 개선 우선 순위 지정](#)

OPS09-BP01 지표를 통한 운영 목표 및 KPI 측정

조직의 운영 성공을 정의하는 목표와 KPI를 확보하고 지표가 이를 반영하는지 결정하세요. 기준선을 참조 지점으로 설정하고 정기적으로 재평가하세요. 평가를 위해 팀으로부터 이러한 메트릭을 수집하는 메커니즘을 개발하세요.

원하는 결과:

- 조직 운영 팀의 목표 및 KPI가 게시되고 공유됩니다.
- 이러한 KPI를 반영하는 지표가 설정됩니다. 예시에는 다음이 포함될 수 있습니다.
 - 티켓 대기열 길이 또는 티켓의 평균 수명
 - 문제 유형별로 그룹화된 티켓 수
 - 표준화된 운영 절차(SOP)를 사용하거나 사용하지 않고 문제를 해결하는 데 소요된 시간
 - 실패한 코드 푸시를 복구하는 데 소요된 시간
 - 통화 음량

일반적인 안티 패턴:

- 개발자가 문제 해결 작업을 수행할 수 밖에 없기 때문에 배포 기한을 놓치는 경우가 있습니다. 개발 팀은 더 많은 인력을 확보하기 위해 노력하고 있지만 소요되는 시간을 측정할 수 없기 때문에 필요한 인원을 정량화할 수 없습니다.
- 티어1 데스크는 사용자 통화를 처리하도록 설정됩니다. 시간이 지나면서 더 많은 워크로드가 추가되었지만 Tier 1 데스크에는 인력이 할당되지 않습니다. 통화 시간이 늘어나고 해결 없이 문제가 더 길어지면서 고객 만족도가 떨어지지만 경영진은 그러한 지표를 발견하지 못해 조치를 취하지 못합니다.
- 문제가 되는 워크로드는 유지 관리를 위해 별도의 운영 팀에 전달되었습니다. 다른 워크로드와 달리 이 새 워크로드는 적절한 설명서 및 런북과 함께 제공되지 않습니다. 따라서 팀은 문제를 해결하고 장애를 해결하는 데 더 많은 시간을 할애합니다. 그러나 이를 문서화하는 지표가 없기 때문에 책임 소재를 찾기가 어렵습니다.

모범 사례 확립의 이점: 워크로드 모니터링을 통해 애플리케이션 및 서비스의 상태를 확인할 수 있는 반면, 모니터링 운영 팀은 소유자에게 변화하는 비즈니스 요구와 같은 워크로드 소비자 간의 변화에 대한 통찰력을 제공합니다. 운영 상태를 반영할 수 있는 지표를 만들어 이러한 팀의 효율성을 측정하고 비즈니스 목표와 비교하여 평가합니다. 지표를 통해 지원 문제를 강조하거나 서비스 수준 목표에서 벗어나는 편차가 발생하는 시점을 파악할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

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

리소스

관련 문서:

- [Amazon QuickSight- KPI 사용](#)

- [Amazon CloudWatch - 지표 사용](#)
- [대시보드 구축](#)
- [KPI 대시보드로 비용 최적화 KPI를 추적하는 방법](#)

OPS09-BP02 상태 및 추세를 전달하여 운영에 대한 가시성 확보

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

원하는 결과:

- 운영 책임자는 팀이 얼마만큼의 통화량을 받고 있는지, 배포와 같이 어떤 작업을 진행 중인지 한눈에 파악할 수 있습니다.
- 정상 운영에 영향이 발생할 경우 이해 관계자와 사용자 커뮤니티에 알림이 전달됩니다.
- 조직 경영진과 이해 관계자는 경고 또는 영향에 대응하여 상태 페이지를 확인하고 연락처, 티켓 정보, 예상 복구 시간 등 운영 이벤트와 관련된 정보를 얻을 수 있습니다.
- 경영진 및 기타 이해 관계자에게 보고서를 제공하여 일정 기간 동안의 통화량, 사용자 만족도 점수, 미결 티켓 수 및 연령과 같은 운영 통계를 보여줍니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점: 비즈니스에 영향을 미치는 운영 이벤트 중에는 상황을 파악하기 위해 노력하는 여러 팀의 정보를 쿼리하느라 많은 시간과 에너지가 낭비될 수 있습니다. 널리 보급된 상태 페이지와 대시보드를 구축함으로써 이해관계자들은 문제가 감지되었는지 여부, 문제의 주체가 누구인지, 정상 운영 상태로 돌아갈 것으로 예상되는 시기와 같은 정보를 신속하게 얻을 수 있습니다. 이렇게 하면 팀원들이 다른 사람에게 상태를 전달하는 데 너무 많은 시간을 소비하지 않고 문제를 해결하는 데 더 많은 시간을 할애할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

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

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

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

리소스

관련 문서:

- [진행 상황 측정](#)
- [운영 가시성을 위한 대시보드 구축](#)

관련 솔루션:

- [데이터 작업](#)

OPS09-BP03 운영 지표 검토 및 개선 우선 순위 지정

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

원하는 결과:

- 운영 책임자와 직원은 정기적으로 만나 지정된 보고 기간 동안의 지표를 검토합니다. 도전 과제를 전달하고, 긍정적인 결과를 축하하고, 배운 교훈을 공유합니다.

- 이해 관계자와 비즈니스 리더는 운영 현황에 대해 정기적으로 브리핑을 받고 목표, KPI 및 향후 이니셔티브에 대한 의견을 요청받습니다. 서비스 제공, 운영 및 유지 관리 간의 장단점을 논의하고 상황에 맞게 적용합니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점: 일부 조직에서는 서비스 제공과 신제품 또는 서비스에 동일한 시간과 관심을 할당하는 것이 어려울 수 있습니다. 이 경우 예상 서비스 수준이 서서히 저하되어 업무 부서에 문제가 발생할 수 있습니다. 비즈니스가 성장해도 운영은 변화하지 않고 발전하지 않으며 곧 뒤쳐질 수 있기 때문입니다. 운영 팀에서 수집한 인사이트를 정기적으로 검토하지 않으면 비즈니스에 미치는 위험은 너무 늦었을 때만 가시화될 수 있습니다. 운영 담당자와 경영진 모두에게 지표와 절차를 검토하는데 시간을 할당함으로써 운영팀이 수행하는 중요한 역할을 가시화하고 위험 수준이 위험한 수준에 도달하기 훨씬 전에 위험을 식별할 수 있습니다. 운영 팀은 임박한 비즈니스 변경 및 이니셔티브를 더 잘 파악하여 사전 조치를 취할 수 있습니다. 운영 지표에 대한 리더십 가시성은 이러한 팀이 내부 및 외부 모두에서 고객 만족도에서 수행하는 역할을 보여주고, 팀이 우선 순위에 대한 선택을 더 잘 판단하거나 운영팀이 새로운 비즈니스 및 워크로드 이니셔티브를 통해 변화하고 발전하는 데 필요한 시간과 리소스를 확보할 수 있도록 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

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

리소스

관련 문서:

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

이벤트 대응

계획된 운영 이벤트(예: 판매 프로모션, 배포 및 장애 테스트)와 계획되지 않은 운영 이벤트(예: 사용자 및 구성 요소 장애의 급증)를 모두 예상해야 합니다. 알림에 대응할 때는 일관된 결과가 제공되도록 기존 런북과 플레이북을 사용해야 합니다. 대응과 에스컬레이션을 담당하는 팀이나 역할이 정의된 알림을 소유해야 합니다. 또한 시스템 구성 요소가 업무에 주는 영향을 확인하고, 해당 정보를 활용해 필요 시의 작업 대상을 지정할 수 있습니다. 이벤트 후에는 근본 원인 분석(RCA)을 수행해야 하며 장애 재발을 방지하거나 해결 방법을 문서로 작성해야 합니다.

AWS에서는 워크로드와 운영의 모든 측면을 코드로 지원하는 도구가 제공되므로 이벤트에 손쉽게 대응할 수 있습니다. 이러한 도구를 사용하면 운영 이벤트 관련 대응을 스크립트로 작성할 수 있으며, 모니터링 데이터에 대한 응답으로 스크립트 실행을 시작할 수 있습니다.

AWS에서는 장애가 발생한 구성 요소의 복구를 시도하는 대신 알려진 정상 버전으로 교체함으로써 복구 시간을 줄일 수 있습니다. 그런 후에는 장애가 발생한 대역 외 리소스에 대한 분석을 수행할 수 있습니다.

모범 사례

- [OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용](#)

- [OPS10-BP02 알림별 프로세스 마련](#)
- [OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정](#)
- [OPS10-BP04 에스컬레이션 경로 정의](#)
- [OPS10-BP05 서비스에 영향을 미치는 이벤트에 대한 고객 커뮤니케이션 계획 정의](#)
- [OPS10-BP06 대시보드를 통해 상태 전달](#)
- [OPS10-BP07 이벤트 대응 자동화](#)

OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

이벤트, 인시던트 및 문제에 대한 이해

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

구현 단계

이벤트

1. 이벤트 모니터링:

- [관찰성을 구현](#) 하고 [워크로드 관찰성을 활용](#)합니다.
- 사용자, 역할 또는 AWS 서비스에서 수행한 모니터링 작업은 [AWS CloudTrail](#)에서 이벤트로 기록됩니다.
- 애플리케이션의 운영 변화에 실시간으로 대응하기 위해 [Amazon EventBridge](#)를 사용합니다.
- 리소스 구성 변경 사항을 지속적으로 평가, 모니터링 및 기록하기 위해 [AWS Config](#)를 사용합니다.

2. 프로세스 생성:

- 어떤 이벤트가 중요하고 모니터링이 필요한지 평가하는 프로세스를 개발합니다. 여기에는 정상 및 비정상 활동에 대한 임계값 및 매개변수 설정이 포함됩니다.
- 이벤트를 인시던트로 에스컬레이션하는 기준을 결정합니다. 심각도, 사용자에게 미치는 영향 또는 예상 행동과의 차이를 토대로 결정할 수 있습니다.
- 이벤트 모니터링 및 대응 프로세스를 정기적으로 검토합니다. 여기에는 과거 인시던트 분석, 임계값 조정, 경고 메커니즘 개선이 포함됩니다.

인시던트

1. 인시던트에 대응:

- 관찰성 도구의 인사이트를 사용하여 인시던트를 빠르게 식별하고 이에 대응합니다.
- 그리고 [AWS Systems Manager 운영 센터](#) 를 구현하여 운영 항목 및 인시던트를 집계하고 체계화 하며 우선 순위를 지정합니다.
- 심층 분석 및 문제 해결을 위해 [Amazon CloudWatch](#) 및 [AWS X-Ray](#) 와 같은 서비스를 사용합니다.
- 사전 예방, 예방 및 탐지 기능을 활용하여 인시던트 관리를 개선하기 위해 [AWS Managed Services\(AMS\)](#) 를 고려합니다. AMS는 모니터링, 인시던트 탐지 및 대응, 보안 관리와 같은 서비스를 통해 운영 지원을 확대합니다.
- 엔터프라이즈 지원 고객은 다음을 사용할 수 있습니다. [AWS 인시던트 탐지 및 대응](#) 프로덕션 워크로드에 대한 지속적인 사전 모니터링 및 인시던트 관리를 제공합니다.

2. 인시던트 관리 프로세스 만들기:

- 명확한 역할, 커뮤니케이션 프로토콜, 해결 단계를 포함한 구조화된 인시던트 관리 프로세스를 수립합니다.
- 인시던트 관리를 [AWS Chatbot](#) 와 같은 도구와 통합하여 효율적으로 대응하고 조정합니다.
- 각 카테고리에 대해 사전 정의된 [인시던트 대응 계획](#) 을 사용하여 심각도를 기준으로 인시던트를 분류합니다.

3. 학습 및 개선:

- 근본 원인을 이해하고 해결 방법의 효과를 확인하기 위해 [인시던트 사후 분석](#) 을 수행합니다.
- 검토 및 발전하는 관행을 토대로 대응 계획을 지속적으로 업데이트하고 개선합니다.
- 팀 전반에서 학습한 내용을 문서화하고 공유하여 운영 복원력을 개선합니다.
- Enterprise Support 고객은 기술 지원 관리자의 [인시던트 관리 워크숍](#) 을 요청할 수 있습니다. 이 안내 워크숍에서는 기존 인시던트 대응 계획을 테스트하고 개선할 수 있는 영역을 식별하도록 돕습니다.

문제

1. 문제 파악:

- 이전 인시던트의 데이터를 사용하여 심층적인 시스템 문제를 시사하는 반복 패턴을 식별합니다.
- 그리고 [AWS CloudTrail](#) 및 [Amazon CloudWatch](#) 와 같은 도구를 활용하여 추세를 분석하고 근본적인 문제를 파악합니다.
- 운영, 개발, 사업부를 비롯한 여러 팀이 참여하여 근본 원인에 대한 다양한 관점을 확보합니다.

- 빠른 해결보다는 장기적인 해결책에 초점을 맞춰 체계적인 문제 관리 프로세스를 개발합니다.
 - 근본 원인 분석(RCA) 기술을 통합하여 인시던트의 근본 원인을 조사하고 이해합니다.
 - 결과를 기반으로 운영 정책, 절차 및 인프라를 업데이트하여 재발을 방지합니다.
3. 지속적인 개선:
- 지속적인 학습과 개선의 문화를 조성하여 팀이 잠재적인 문제를 사전에 식별하고 해결하도록 독려합니다.
 - 진화하는 비즈니스 및 기술 환경에 맞게 문제 관리 프로세스와 도구를 정기적으로 검토하고 수정합니다.
 - 조직 전반에 걸쳐 인사이트와 모범 사례를 공유하여 보다 복원력 있고 효율적인 운영 환경을 구축합니다.
4. AWS Support 참여:
- AWS 지원 리소스(예: [AWS Trusted Advisor](#))를 사전 예방적 지침 및 최적화 권장 사항으로 활용합니다.
 - 엔터프라이즈 지원 고객은 [AWS 카운트다운](#) 과 같은 전문 프로그램을 통해 중요 이벤트 발생 시 지원을 받을 수 있습니다.
 -

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

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

관련 문서:

- [AWS 보안 인시던트 대응 가이드](#)
- [AWS 인시던트 탐지 및 대응](#)

- [AWS 클라우드 채택 프레임워크: 운영 관점 - 인시던트 및 문제 관리](#)
- [DevOps 및 SRE 시대의 인시던트 관리](#)
- [PagerDuty - PagerDuty - 인시던트 관리란 무엇인가요?](#)

관련 동영상:

- [AWS에서 제공하는 주요 인시던트 대응 팁](#)
- [AWS re:Invent 2022 - Amazon Builders' Library: 25년간 운영 분야에서 Amazon이 보여준 우수성](#)
- [AWS re:Invent 2022 - AWS 인시던트 감지 및 대응\(SUP201\)](#)
- [AWS Systems Manager의 Incident Manager 소개](#)

관련 예시:

- [AWS 사전 예방적 서비스 - 인시던트 관리 워크숍](#)
- [PagerDuty 및 AWS Systems Manager Incident Manager를 사용한 인시던트 대응 자동화 방법](#)
- [AWS Systems Manager Incident Manager에서 온콜 일정을 통해 인시던트 대응 담당자의 참여 유도](#)
- [AWS Systems Manager Incident Manager에서 인시던트 처리 중 가시성 및 협업 개선](#)
- [AMS의 인시던트 보고서 및 서비스 요청](#)

관련 서비스:

- [Amazon EventBridge](#)

OPS10-BP02 알림별 프로세스 마련

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 실행 가능한 알림만 발생시켜 알림 피로 감소
- 운영 문제의 평균 해결 시간(MTTR) 단축
- 평균 조사 시간(MTTI)이 단축되어 MTTR을 단축
- 운영 대응을 확장할 수 있는 기능 향상
- 운영 이벤트 처리의 일관성과 신뢰성 향상

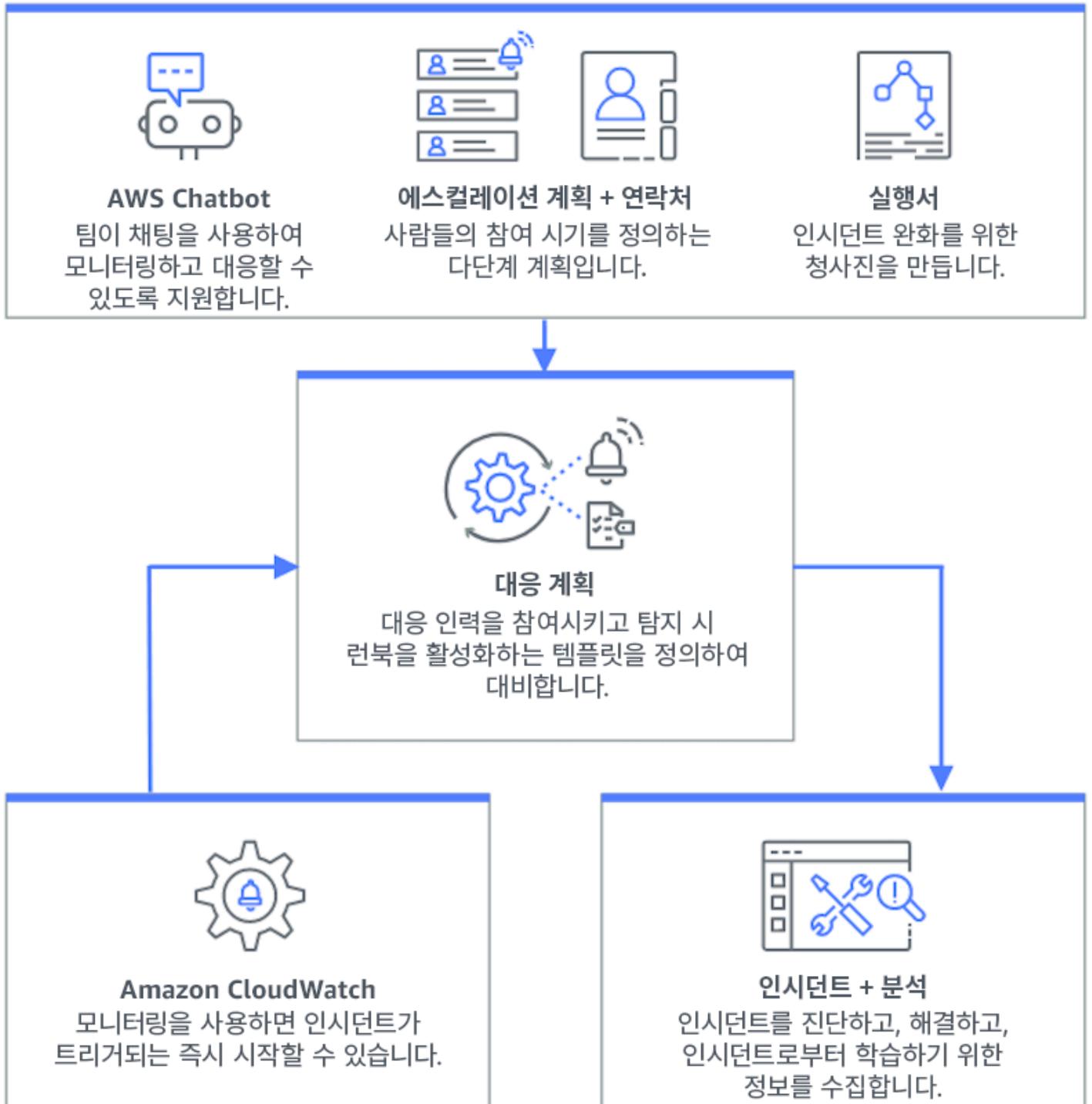
이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 가이드

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

구현 단계

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



1. 복합 경고 사용: CloudWatch에서 **복합 경고** 를 생성하여 경보를 그룹화하고 노이즈를 줄이며 보다 의미 있는 대응이 가능하게 합니다.
2. Amazon CloudWatch 경보를 Incident Manager에 통합합니다. 그리고 **AWS Systems Manager Incident Manager**에서 인시던트를 자동으로 생성하도록 CloudWatch 경보를 구성합니다.

3. Amazon EventBridge를 Incident Manager와 통합: 또한 [EventBridge 규칙](#) 을 만들어 정의된 대응 계획에 따라 이벤트에 대응하고 인시던트를 생성합니다.
4. Incident Manager에서 인시던트에 대비:
 - 알림 유형별 세부 [대응 계획](#) 을 Incident Manager에서 수립합니다.
 - Incident Manager의 대응 계획에 연결된 [AWS Chatbot](#) 를 통해 채팅 채널을 구축하여 Slack Microsoft Teams 및 Amazon Chime과 같은 플랫폼 전반에서 인시던트 발생 시 실시간 커뮤니케이션을 용이하게 합니다.
 - 그리고 [Systems Manager 자동화 런북](#) 을 통합하여(Incident Manager 내) 인시던트에 대한 자동 대응을 유도합니다.

리소스

관련 모범 사례:

- [OPS04-BP01 핵심 성과 지표 파악](#)
- [OPS08-BP04 실행 가능한 알림 생성](#)

관련 문서:

- [AWS 클라우드 채택 프레임워크: 운영 관점 - 인시던트 및 문제 관리](#)
- [Amazon CloudWatch 경보 사용](#)
- [AWS Systems Manager Incident Manager 설정](#)
- [Incident Manager에서 인시던트에 대비](#)

관련 동영상:

- [AWS에서 제공하는 주요 인시던트 대응 팁](#)

관련 예시:

- [AWS 워크숍 - AWS Systems Manager Incident Manager - 보안 이벤트에 대한 인시던트 대응 자동화](#)

OPS10-BP03 비즈니스 영향을 기반으로 운영 이벤트의 우선순위 지정

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 중요한 비즈니스 기능에 먼저 주의를 기울이도록 하여 잠재적 손상을 최소화
- 여러 동시 이벤트 발생 시 리소스 할당을 개선
- 조직의 신뢰 유지 및 규제 요구 사항 충족 능력을 향상

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

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

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

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

| 긴급성 수준 | 설명 |
|--------|--|
| 높음 | 피해가 기하급수적으로 증가하고, 시간에 민감한 업무가 영향을 받거나, 에스컬레이션이 임박하거나, VIP 사용자 또는 그룹이 영향을 받습니다. |
| 보통 | 피해가 시간이 경과함에 따라 증가하거나 단일 VIP 사용자 또는 그룹이 영향을 받습니다. |
| 낮음 | 시간이 지남에 따라 미미한 손상이 증가하거나 시간에 민감하지 않은 작업에 영향을 미칩니다. |

3. 우선순위 매트릭스 만들기:

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

| 긴급성 및 영향 | 높음 | 보통 | 낮음 |
|----------|----|----|----|
| 높음 | 심각 | 긴급 | 높음 |
| 보통 | 긴급 | 높음 | 일반 |
| 낮음 | 높음 | 일반 | 낮음 |

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

리소스

관련 모범 사례:

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

관련 문서:

- [Atlassian - 인시던트 심각도 수준의 이해](#)
- [IT 프로세스 맵 - 인시던트 우선 순위 체크리스트](#)

OPS10-BP04 에스컬레이션 경로 정의

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

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 사전 정의된 에스컬레이션 절차를 통해 인시던트 대응 간소화
- 사전 승인된 조치와 명확한 소유권을 통해 다운타임 감소
- 인시던트 심각도에 따라 리소스 할당 및 지원 수준 조정 개선
- 이해관계자 및 고객과의 커뮤니케이션 개선

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

1. 에스컬레이션 프롬프트 설정: 먼저 [CloudWatch 경보](#) 를 설정하여 [AWS Systems Manager Incident Manager](#)에서 인시던트를 생성합니다.
2. 온콜 일정 설정: 그리고 [온콜 스케줄](#) 을 Incident Manager에서 설정합니다(이는 에스컬레이션 경로와 일치). 대기 중인 직원에게 신속하게 조치를 취하는 데 필요한 권한과 도구를 제공합니다.
3. 상세 에스컬레이션 절차:

- 인시던트를 에스컬레이션해야 하는 구체적인 조건을 결정합니다.
 - 그리고 [에스컬레이션 계획](#) 을 생성합니다(Incident Manager에서).
 - 에스컬레이션 채널은 연락처 또는 온콜 일정으로 구성되어야 합니다.
 - 각 에스컬레이션 수준에서 팀의 역할과 책임을 정의합니다.
4. 완화 조치 사전 승인: 의사 결정권자와 협업하여 예상 시나리오에 대한 조치를 사전 승인합니다. 그리고 [Systems Manager 자동화 런북](#) (Incident Manager와 통합되어 인시던트 해결 속도 향상)을 사용합니다.
 5. 소유권 지정: 에스컬레이션 경로의 각 단계에서 내부 소유자를 명확하게 식별합니다.
 6. 타사 에스컬레이션에 대한 세부 정보:
 - 타사 서비스 수준 계약(SLA)을 문서화하고 내부 목표에 맞게 조정합니다.
 - 인시던트 발생 시 공급업체 커뮤니케이션을 위한 명확한 프로토콜을 설정합니다.
 - 공급업체 연락처를 인시던트 관리 도구에 통합하여 직접 액세스할 수 있습니다.
 - 타사 대응 시나리오가 포함된 정기적인 훈련을 실시합니다.
 - 공급업체 에스컬레이션 정보를 체계적으로 문서화하고 쉽게 액세스할 수 있도록 합니다.
 7. 에스컬레이션 계획 교육 및 연습: 에스컬레이션 프로세스에 대해 팀을 교육하고 정기적인 인시던트 대응 훈련 또는 게임 데이를 실시합니다. Enterprise Support 고객은 [인시던트 관리 워크숍](#)을 요청할 수 있습니다.
 8. 지속적인 개선: 에스컬레이션 경로의 효과를 정기적으로 검토합니다. 인시던트 사후 분석 및 지속적인 피드백을 통해 얻은 교훈을 기반으로 프로세스를 업데이트합니다.

구현 계획의 작업 수준: 보통

리소스

관련 모범 사례:

- [OPS08-BP04 실행 가능한 알림 생성](#)
- [OPS10-BP02 알림별 프로세스 마련](#)
- [OPS11-BP02 인시던트 사후 분석 수행](#)

관련 문서:

- [AWS Systems Manager Incident Manager 에스컬레이션 계획](#)
- [Incident Manager에서 온콜 일정 작업](#)

- [런북 생성 및 관리](#)
- [AWS IAM Identity Center을 통한 임시 승격 액세스 관리](#)
- [Atlassian - 효과적인 인시던트 관리를 위한 에스컬레이션 정책](#)

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

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

원하는 결과:

- 서비스에 영향을 미치는 이벤트 발생 시 고객과 이해관계자에게 효과적으로 정보를 제공하는 탄탄한 커뮤니케이션 계획.
- 커뮤니케이션의 투명성을 통해 신뢰를 구축하고 고객의 불안을 줄입니다.
- 서비스에 영향을 미치는 이벤트가 고객 경험 및 비즈니스 운영에 미치는 영향을 최소화합니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 적극적이고 명확한 커뮤니케이션을 통해 고객 신뢰와 만족도를 향상
- 고객 문제를 선제적으로 해결하여 지원 팀의 부담 완화
- 인시던트를 효과적으로 관리하고 복구하는 능력 향상

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

1. 역할과 책임 정의

- 주요 인시던트 관리자를 지정하여 인시던트 대응 활동을 감독합니다.
- 모든 외부 및 내부 커뮤니케이션을 조정할 책임이 있는 커뮤니케이션 관리자를 지정합니다.
- 지원 티켓을 통해 일관된 커뮤니케이션이 가능하도록 지원 관리자를 포함합니다.

2. 커뮤니케이션 채널 파악: 워크플레이스 채팅, 이메일, SMS, 소셜 미디어, 앱 내 알림, 상태 페이지와 같은 채널을 선택합니다. 이러한 채널은 복원력이 있어야 하며 서비스에 영향을 미치는 이벤트 발생 시 독립적으로 운영될 수 있어야 합니다.

3. 고객에게 빠르고 명확하게 정기적으로 커뮤니케이션하세요.

- 단순성과 필수 세부 정보를 강조하여 다양한 서비스 장애 시나리오에 대한 템플릿을 개발하고 템플릿에 서비스 장애, 예상 해결 시간 및 영향에 대한 정보를 포함하세요.
- Amazon Pinpoint를 사용하여 푸시 알림, 인앱 알림, 이메일, 문자 메시지, 음성 메시지 및 사용자 지정 채널을 통한 메시지를 사용하여 고객에게 알리고
- Amazon Simple Notification Service(Amazon SNS)를 사용하여 프로그래밍 방식으로 또는 이메일, 모바일 푸시 알림 및 문자 메시지를 통해 구독자에게 알리세요.
- Amazon CloudWatch 대시보드를 공개적으로 공유하여 대시보드를 통해 상태를 전달합니다.
- 소셜 미디어 참여 장려:
 - 소셜 미디어를 적극적으로 모니터링하여 고객의 분위기를 파악합니다.
 - 소셜 미디어 플랫폼에 게시하여 공개 업데이트 및 커뮤니티 참여를 확인합니다.
 - 일관되고 명확한 소셜 미디어 커뮤니케이션을 위한 템플릿을 준비합니다.

4. 내부 커뮤니케이션 조정: 팀 조정 및 커뮤니케이션을 위해 AWS Chatbot 같은 도구를 사용하여 내부 프로토콜을 구현합니다. CloudWatch 대시보드를 사용하여 상태를 알립니다.

5. 전용 도구 및 서비스를 사용하여 커뮤니케이션을 조율합니다.

- AWS Chatbot와 함께 AWS Systems Manager Incident Manager를 사용하여 인시던트 발생 시 실시간 내부 커뮤니케이션 및 조정을 위한 전용 채팅 채널을 설정합니다.
- AWS Systems Manager Incident Manager 런북을 사용하여 인시던트 발생 시 Amazon Pinpoint, Amazon SNS 또는 소셜 미디어 플랫폼과 같은 타사 도구를 통해 고객 알림을 자동화합니다.

- 런북에 승인 워크플로를 통합하여 선택적으로 전송 전에 모든 외부 커뮤니케이션을 검토하고 승인할 수 있습니다.

6. 연습 및 개선:

- 커뮤니케이션 도구 및 전략의 사용에 대한 교육을 실시합니다. 팀이 인시던트 발생 시 시의적절하게 결정을 내릴 수 있도록 지원합니다.
- 정기적인 훈련이나 게임데이를 통해 커뮤니케이션 계획을 테스트합니다. 이 테스트를 사용하여 메시지를 구체화하고 채널의 효과를 평가합니다.
- 피드백 메커니즘을 구현하여 인시던트 발생 시 커뮤니케이션 효과를 평가합니다. 피드백과 변화하는 요구 사항을 기반으로 커뮤니케이션 계획을 지속적으로 발전시킵니다.

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- [OPS07-BP03 런북을 사용한 절차 수행](#)
- [OPS10-BP06 대시보드를 통해 상태 전달](#)
- [OPS11-BP02 인시던트 사후 분석 수행](#)

관련 문서:

- [Atlassian - 인시던트 커뮤니케이션 모범 사례](#)
- [Atlassian - 효과적인 상태 업데이트를 작성하는 방법](#)
- [PagerDuty - 인시던트 커뮤니케이션 가이드](#)

관련 동영상:

- [Atlassian - 나만의 인시던트 커뮤니케이션 계획 만들기: 인시던트 템플릿](#)

관련 예시:

- [AWS Health 대시보드](#)
- [AWS 상태 업데이트 예시](#)

OPS10-BP06 대시보드를 통해 상태 전달

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 이해 관계자가 중요한 정보에 즉시 액세스할 수 있도록 하여 정보에 입각한 의사 결정을 촉진
- 수동 보고 및 빈번한 상태 조회를 최소화하여 운영 비효율성 완화
- 시스템 성능 및 비즈니스 지표에 대한 실시간 가시성을 통해 투명성과 신뢰도 향상

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

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

구현 단계

1. 이해관계자의 요구 사항 파악: 기술팀, 경영진, 고객 등 다양한 대상 그룹의 특정 정보 요구 사항을 결정합니다.

2. 적절한 도구 선택: 시스템 모니터링을 위한 [Amazon CloudWatch 대시보드](#) 및 대화형 비즈니스 인텔리전스를 위한 [Amazon QuickSight](#) 등의 적절한 도구를 선택합니다.
3. 효과적인 대시보드 설계:
 - 관련 지표와 KPI를 명확하게 제시하여 이해할 수 있고 실행 가능한 방식으로 대시보드를 설계합니다.
 - 필요에 따라 시스템 수준 및 비즈니스 수준 뷰를 통합합니다.
 - 상위 수준(광범위한 개요용) 및 하위 수준(세부 분석용) 대시보드를 모두 포함합니다.
 - 대시보드 내에 자동 경보를 통합하여 중요한 문제를 강조 표시합니다.
 - 대시보드에 중요한 지표 임계값 및 목표를 주석으로 추가하여 즉시 확인할 수 있습니다.
4. 데이터 소스 통합:
 - 먼저 [Amazon CloudWatch](#) 를 사용하여 다양한 AWS 서비스의 지표를 집계 및 표시하고 [다른 데이터 소스의 지표를 쿼리하여](#) 시스템의 상태 및 비즈니스 지표에 대한 통합된 뷰를 생성합니다.
 - 그리고 [CloudWatch Logs Insights](#) 와 같은 기능을 사용하여 다양한 애플리케이션 및 서비스의 로그 데이터를 쿼리하고 시각화합니다.
5. 데이터에 대한 셀프 서비스 액세스 제공:
 - 셀프 서비스 정보 액세스를 위해 관련 이해 관계자와 CloudWatch 대시보드를 공유합니다([대시보드 공유 기능](#) 사용).
 - 대시보드에 쉽게 액세스할 수 있도록 하고 실시간 최신 정보를 제공합니다.
6. 정기적으로 업데이트하고 수정합니다.
 - 진화하는 비즈니스 요구 사항 및 이해 관계자 피드백에 맞춰 대시보드를 지속적으로 업데이트하고 수정합니다.
 - 대시보드를 정기적으로 검토하여 필요한 정보를 전달하는 데 적합하고 효과적인지 확인합니다.

리소스

관련 모범 사례:

- [OPS08-BP05 대시보드 만들기](#)

관련 문서:

- [운영 가시성을 위한 대시보드 구축](#)
- [Amazon CloudWatch 대시보드 사용](#)

- [대시보드 변수를 사용하여 유연한 대시보드 만들기](#)
- [CloudWatch 대시보드 공유](#)
- [다른 데이터 소스의 지표 쿼리](#)
- [CloudWatch 대시보드에 사용자 지정 위젯 추가](#)

관련 예시:

- [단일 관찰성 워크숍 - 대시보드](#)

OPS10-BP07 이벤트 대응 자동화

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

원하는 결과:

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

일반적인 안티 패턴:

- 수동 이벤트 처리로 인해 지연과 오류가 발생합니다.
- 반복적이고 중요한 작업에서 자동화가 간과됩니다.
- 반복적인 수동 작업으로 인해 알림에 대한 피로감이 쌓이고 중요한 문제가 누락됩니다.

이 모범 사례 확립의 이점:

- 이벤트 대응 가속화를 통한 시스템 다운타임 감소.
- 자동화되고 일관된 이벤트 처리를 통한 안정적인 운영.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 보통

구현 가이드

자동화를 통합하여 효율적인 운영 워크플로를 만들고 수동 개입을 최소화합니다.

구현 단계

1. 자동화 기회 파악: 문제 해결, 티켓 강화, 용량 관리, 확장, 배포 및 테스트와 같은 자동화를 위한 반복 작업을 결정합니다.
2. 자동화 프롬프트 확인:
 - 이 단계에서는 [Amazon CloudWatch 경보 작업](#)을 사용하여 자동 응답을 게시하는 특정 조건이나 지표를 평가 및 정의합니다.
 - 또한 [Amazon EventBridge](#) 을 사용하여 AWS 서비스, 맞춤 워크로드, SaaS 애플리케이션의 이벤트에 응답합니다.
 - AWS 리소스에서 [특정 로그 항목](#), [성능 지표 임계값](#) 또는 [상태 변경](#) 등의 시작 이벤트를 고려해 보세요.
3. 이벤트 기반 자동화 구현:
 - AWS Systems Manager 자동화 런북을 사용하여 유지 관리, 배포 및 수정 작업을 간소화합니다.
 - [Incident Manager에서 인시던트를 생성하면](#) AWS 관련 리소스에 대한 세부 정보를 자동으로 수집하고 인시던트에 추가할 수 있습니다.
 - AWS를 위한 Quota Monitor를 사용하여 [할당량을 사전에 모니터링합니다](#).
 - 가용성과 성능을 유지하기 위해 [AWS Auto Scaling](#) 를 사용하여 용량을 자동으로 조정합니다.
 - 그리고 [Amazon CodeCatalyst](#)를 사용하여 개발 파이프라인을 자동화합니다.
 - 종합 모니터링을 사용하여 엔드포인트 및 API를 [스모크 테스트하거나 지속적으로 모니터링합니다](#).
4. 자동화를 통한 위험 완화 수행:
 - 위험을 신속하게 해결하기 위해 [자동화된 보안 대응](#) 을 구현합니다.
 - 그리고 [AWS Systems Manager State Manager](#) 를 사용하여 시간이 지나면서 발생하는 구성 편차를 줄입니다.
 - [AWS Config 규칙을 사용하여 규정 미준수 리소스를 수정합니다](#).

구현 계획의 작업 수준: 높음

리소스

관련 모범 사례:

- [OPS08-BP04 실행 가능한 알림 생성](#)
- [OPS10-BP02 알림별 프로세스 마련](#)

관련 문서:

- [인시던트 관리자와 함께 Systems Manager 자동화 런북 사용](#)
- [인시던트 관리자에서 인시던트 생성](#)
- [AWS Service Quotas](#)
- [리소스 사용량을 모니터링하고 할당량 도달 시 알림 전송](#)
- [AWS Auto Scaling](#)
- [Amazon CodeCatalyst란 무엇인가요?](#)
- [Amazon CloudWatch 경보 사용](#)
- [Amazon CloudWatch 경보 작업 사용](#)
- [AWS Config 규칙을 통해 규정 미준수 리소스 문제 해결](#)
- [필터를 사용하여 로그 이벤트에서 지표 생성](#)
- [AWS Systems Manager State Manager](#)

관련 동영상:

- [AWS Systems Manager를 사용하여 자동화 런북 생성](#)
- [AWS에서 IT 운영을 자동화하는 방법](#)
- [AWS Security Hub 자동화 규칙](#)
- [Amazon CodeCatalyst 청사진으로 소프트웨어 프로젝트를 빠르게 시작](#)

관련 예시:

- [Amazon CodeCatalyst 튜토리얼: 최신 3계층 웹 애플리케이션 청사진을 사용한 프로젝트 생성](#)
- [One Observability Workshop](#)
- [Incident Manager를 사용하여 인시던트에 대응](#)

개선

개선은 장기적으로 진행되는 지속적인 향상 주기입니다. 운영 활동에서 파악한 내용을 토대로 하여 소규모 증분 방식 변경을 자주 구현하고, 성공적으로 개선했는지 평가해야 합니다.

시간에 따라 운영을 개선하려면 다음을 수행할 수 있어야 합니다.

주제

- [학습, 공유 및 개선](#)

학습, 공유 및 개선

운영 활동 분석, 장애 분석, 실험 및 운영 방식 향상을 정기적으로 수행해야 합니다. 장애 발생 시에는 팀과 전체 엔지니어링 커뮤니티가 해당 장애에서 유용한 내용을 파악할 수 있어야 합니다. 그리고 장애를 분석하여 파악한 내용을 확인하고 운영 방식의 향상을 계획해야 합니다. 그리고 다른 팀이 파악한 내용도 정기적으로 검토하여 기존에 파악했던 내용을 다시 검증해야 합니다.

모범 사례

- [OPS11-BP01 지속적인 개선을 위한 프로세스 마련](#)
- [OPS11-BP02 인시던트 사후 분석 수행](#)
- [OPS11-BP03 피드백 루프 구현](#)
- [OPS11-BP04 지식 관리 수행](#)
- [OPS11-BP05 개선 추진 요인 정의](#)
- [OPS11-BP06 인사이트 검증](#)
- [OPS11-BP07 운영 지표 검토 수행](#)
- [OPS11-BP08 파악한 내용 문서화 및 공유](#)
- [OPS11-BP09 개선을 위한 시간 할애](#)

OPS11-BP01 지속적인 개선을 위한 프로세스 마련

내부 및 외부 아키텍처 모범 사례를 기준으로 워크로드를 평가하세요. 의도적인 워크로드 검토를 자주 실시하세요. 소프트웨어 개발 단계에서 개선 기회의 우선순위를 지정하세요.

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

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

구현 단계

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

구현 계획의 작업 수준: 낮음 AWS Well-Architected Framework를 사용하여 연간 아키텍처 검토를 수행할 수 있습니다.

리소스

관련 모범 사례:

- [OPS11-BP02 인시던트 사후 분석 수행](#)
- [OPS11-BP08 파악한 내용 문서화 및 공유](#)
- [OPS04 관찰성 구현](#)

관련 문서:

- [AWS Well-Architected Tool - Custom lenses](#)
- [AWS Well-Architected 백서 - 검토 프로세스](#)
- [Custom Lens 및 AWS Well-Architected Tool을 사용하여 Well-Architected 검토 사용자 지정](#)
- [조직에서 AWS Well-Architected Custom Lens 수명 주기 구현](#)

관련 동영상:

- [Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool](#)
- [AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization](#)

관련 예시:

- [AWS Well-Architected Tool](#)

OPS11-BP02 인시던트 사후 분석 수행

고객에게 영향을 주는 이벤트를 검토하고 기여 요인과 예방 조치를 식별합니다. 이 정보를 사용하여 재발을 제한하거나 방지하는 완화 기능을 개발합니다. 신속하고 효과적인 대응을 위한 절차를 개발합니다. 목표 대상에 맞게 적절히 발생 요인과 수정 조치를 전달합니다.

원하는 결과:

- 인시던트 사후 분석을 포함하는 인시던트 관리 프로세스를 수립했습니다.
- 이벤트에 대한 데이터를 수집하기 위한 관찰성 계획이 마련되어 있습니다.

- 이 데이터를 통해 인시던트 사후 분석 프로세스를 지원하는 지표를 이해하고 수집할 수 있습니다.
- 인시던트로부터 교훈을 얻어 미래의 결과를 개선합니다.

일반적인 안티 패턴:

- 애플리케이션 서버를 관리합니다. 약 23시간 55분마다 모든 활성 세션이 종료됩니다. 애플리케이션 서버에서 무엇이 잘못되었는지 파악하려고 했습니다. 네트워크 문제일 수도 있다고 생각하지만 네트워크 팀이 너무 바쁜 관계로 지원을 받을 수 없습니다. 지원을 받고 진행 상황을 파악하는 데 필요한 정보를 수집하기 위해 따라야 할 사전 정의된 프로세스가 없습니다.
- 워크로드 내에서 데이터가 손실되었습니다. 이런 일은 처음이며 그 원인이 명확하지 않습니다. 데이터를 다시 생성할 수 있으므로 대수롭지 않은 일로 생각합니다. 데이터 손실이 발생하면서 고객에게 영향을 미치는 빈도가 증가합니다. 또한 이로 인해 누락된 데이터를 복원할 때 운영 부담이 가중됩니다.

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

발생 요인을 확인하는 프로세스 사용 고객에게 영향을 미치는 모든 인시던트를 검토합니다. 재발을 제한하거나 방지하기 위한 완화책을 개발하고 빠르고 효과적인 대응을 위한 절차를 개발할 수 있도록 인시던트의 기여 요인을 식별하고 문서화하는 프로세스를 마련합니다. 인시던트의 근본 원인을 적절하게 전달하고 대상 고객에 맞게 커뮤니케이션을 조정합니다. 조직 내에서 학습한 내용을 공개적으로 공유합니다.

구현 단계

1. 배포 변경, 구성 변경, 인시던트 시작 시간, 경보 시간, 참여 시간, 완화 시작 시간, 인시던트 해결 시간과 같은 지표를 수집합니다.
2. 인시던트 발생 상황을 파악하기 위해 타임라인에 주요 시점을 표시합니다.
3. 다음과 같이 질문하세요.

- a. 탐지 시간을 단축할 수 있을까요?
 - b. 인시던트를 더 빨리 감지할 수 있는 지표 및 경보에 대한 업데이트가 있나요?
 - c. 진단 시간을 개선할 수 있나요?
 - d. 적절한 대응 담당자를 더 빨리 참여시킬 수 있는 대응 계획 또는 에스컬레이션 계획이 업데이트 되어 있나요?
 - e. 완화 시간을 단축할 수 있나요?
 - f. 추가하거나 개선할 수 있는 런북 또는 플레이북 단계가 있나요?
 - g. 향후 인시던트 발생을 방지할 수 있나요?
4. 체크리스트와 작업을 만듭니다. 모든 작업을 추적하고 전달합니다.

구현 계획의 작업 수준: 중간

리소스

관련 모범 사례:

- [OPS11-BP01 지속적인 개선을 위한 프로세스 마련](#)
- [OPS 4 - 관찰성 구현](#)

관련 문서:

- [인시던트 관리자에서 인시던트 사후 분석 수행](#)
- [운영 준비 상태 검토](#)

OPS11-BP03 피드백 루프 구현

피드백 루프는 의사 결정을 추진하는 실행 가능한 인사이트를 제공합니다. 절차와 워크로드에 피드백 루프를 구축하세요. 이를 통해 문제와 개선이 필요한 영역을 파악할 수 있습니다. 또한 개선에 대한 투자를 검증합니다. 이러한 피드백 루프는 워크로드를 지속적으로 향상하기 위한 기반입니다.

피드백 루프의 두 가지 카테고리: 즉각적 피드백 및 후행 분석. 즉각적 피드백은 운영 활동의 성과 및 결과를 검토하여 수집합니다. 이 피드백은 팀원, 고객 또는 자동화된 활동 출력으로부터 제공됩니다. A/B 테스트 및 새로운 기능 전달과 같은 사항을 통해 즉각적 피드백을 수신하며, 빠른 실패에 필수입니다.

시간 경과에 따른 운영 결과 및 지표 검토 결과의 피드백을 얻을 수 있도록 후행 분석이 정기적으로 수행됩니다. 이러한 후행 분석은 스프린트 후반, 정기적인 주기, 또는 주요 릴리스나 이벤트 이후 수행합

니다. 이러한 유형의 피드백 루프는 운영 또는 워크로드의 투자를 검증합니다. 이를 통해 성공 여부를 측정하고 결과를 검증할 수 있습니다.

원하는 결과: 즉각적 피드백 및 후행 분석을 사용하여 개선을 추진할 수 있습니다. 사용자 및 팀원의 피드백을 얻을 수 있는 메커니즘이 있습니다. 후행 분석은 개선을 추진하는 트렌드를 파악하는 데 사용됩니다.

일반적인 안티 패턴:

- 새로운 기능을 출시했지만 이에 대한 고객 피드백을 받을 수 있는 방법이 없습니다.
- 운영 개선에 투자한 후 이를 검증할만한 후행 분석을 수행하지 않습니다.
- 고객 피드백을 수집하지만 이를 정기적으로 검토하지 않습니다.
- 피드백 루프를 통해 제안된 조치 항목을 얻지만 소프트웨어 개발 프로세스에 포함되지 않습니다.
- 고객이 제안한 개선 사항에 대한 피드백을 받지 못합니다.

이 모범 사례 확립의 이점:

- 고객의 입장에서 시작한 역방향 작업을 통해 새로운 기능을 이끌어낼 수 있습니다.
- 조직 문화가 변화에 빠르게 반응할 수 있습니다.
- 트렌드를 사용하여 개선 기회를 파악할 수 있습니다.
- 후행 분석을 통해 워크로드 및 운영에 대한 투자를 검증할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험의 수준: 높음

구현 가이드

이 모범 사례를 구현하면 즉각적인 피드백과 후행 분석을 모두 사용하게 됩니다. 이러한 피드백 루프를 통해 개선을 추진할 수 있습니다. 설문 조사, 고객 투표, 피드백 양식 등 즉각적 피드백을 위한 다양한 메커니즘이 있습니다. 조직에서는 후행 분석도 사용하여 개선 기회를 파악하고 이니셔티브를 검증합니다.

고객 사례

AnyCompany Retail은 고객이 피드백을 제공하고 문제를 보고할 수 있는 웹 양식을 만들었습니다. 주간 스크럼 기간 동안 소프트웨어 개발 팀이 사용자 피드백을 평가합니다. 피드백은 플랫폼의 평가를 추진하는 데 정기적으로 사용됩니다. 각 스프린트의 후반에는 후행 분석을 수행하여 개선하고자 하는 항목을 파악합니다.

구현 단계

1. 즉각적 피드백

- 고객 및 팀원으로부터 피드백을 수신할 수 있는 메커니즘이 필요합니다. 또한 자동 피드백을 제공하도록 운영 활동을 구성할 수 있습니다.
- 조직은 이 피드백을 검토하고, 개선할 점을 결정하며 개선 일정을 지정하는 프로세스가 필요합니다.
- 피드백이 소프트웨어 개발 프로세스에 반드시 추가되어야 합니다.
- 개선 사항이 있을 때 피드백 제출자에게 후속 조치를 취합니다.
 - 이때 [AWS Systems Manager OpsCenter](#) 를 사용하여 이러한 개선 사항을 [OpsItems](#)로 생성하고 추적할 수 있습니다.

2. 후행 분석

- 개발 주기의 마지막, 정기적인 주기, 또는 주요 릴리스 이후에 후행 분석을 수행합니다.
- 후행 분석 회의를 위해 워크로드에 관련된 이해 관계자를 모읍니다.
- 화이트보드나 스프레드시트에 중지, 시작 및 유지라는 세 개의 열을 만듭니다.
 - 중지는 팀에서 수행을 중지하고자 하는 항목입니다.
 - 시작은 시작하고자 하는 아이디어입니다.
 - 유지는 계속하고자 하는 항목입니다.
- 회의실을 한 바퀴 돌며 이해 관계자들의 피드백을 수렴합니다.
- 피드백의 우선순위를 정합니다. 모든 시작 또는 유지 항목에 대한 활동 및 이해 관계자를 할당합니다.
- 소프트웨어 개발 프로세스에 해당 활동을 추가하고 개선 작업을 수행할 때 이해 관계자에게 상태 업데이트를 전달합니다.

구현 계획의 작업 수준: 보통. 이 모범 사례를 구현하려면 즉각적인 피드백을 수렴하고 이를 분석할 수 있는 방법이 필요합니다. 또한 후행 분석 프로세스를 확립해야 합니다.

리소스

관련 모범 사례:

- [OPS01-BP01 고객 요구 사항 평가](#): 피드백 루프는 외부 고객의 요구를 수집할 수 있는 메커니즘입니다.

- [OPS01-BP02 내부 고객 요구 평가](#): 내부 이해 관계자는 피드백 루프를 사용하여 필요 및 요구 사항을 논의합니다.
- [OPS11-BP02 인시던트 사후 분석 수행](#): 인시던트 사후 분석은 인시던트 후 수행하는 후행 분석의 중요한 양식입니다.
- [OPS11-BP07 운영 지표 검토 수행](#): 운영 지표 검토는 개선을 위한 트렌드와 영역을 파악합니다.

관련 문서:

- [CCOE 구축 시 피해야 할 7가지 위험](#)
- [Atlassian Team 플레이북 - 후행 분석](#)
- [이메일 정의: 피드백 루프](#)
- [AWS Well-Architected 프레임워크 검토를 기반으로 피드백 루프 확립](#)
- [IBM Garage 방법론 - 후행 분석 진행](#)
- [Investopedia - PDCS 주기](#)
- [Tim Cochran의 개발자 효율성 극대화](#)
- [ORR\(운영 준비 상태 검토\) 백서 - 반복](#)
- [TIL CSI - 지속적인 서비스 개선](#)
- [Toyota가 전자 상거래를 만났을 때: Amazon으로부터 배우기](#)

관련 동영상:

- [효과적인 고객 피드백 루프 구축](#)

관련 예시:

- [Astuto - 오픈 소스 고객 피드백 도구](#)
- [AWS 솔루션 - AWS의 QnABot](#)
- [Fider - 고객 피드백 구성을 위한 플랫폼](#)

관련 서비스:

- [AWS Systems Manager OpsCenter](#)

OPS11-BP04 지식 관리 수행

지식 관리란 팀원들이 업무 수행에 필요한 정보를 찾는 데 도움이 됩니다. 학습하는 조직에서는 개인에게 유용한 정보가 자유롭게 공유됩니다. 정보를 발견하거나 검색할 수 있습니다. 정확한 최신 정보입니다. 새로운 정보를 생성하고, 기존 정보를 업데이트하고, 오래된 정보를 보관하는 메커니즘이 있습니다. 지식 관리 플랫폼의 가장 일반적인 예로는 Wiki와 같은 콘텐츠 관리 시스템을 들 수 있습니다.

원하는 결과:

- 팀원이 적시에 정확한 정보에 액세스할 수 있습니다.
- 정보 검색이 가능합니다.
- 정보를 추가, 업데이트 및 보관하는 메커니즘이 있습니다.

일반적인 안티 패턴:

- 중앙 집중식 지식 스토리지가 없습니다. 팀 구성원은 로컬 컴퓨터에서 자신의 메모를 관리합니다.
- 셀프 호스팅된 Wiki가 있지만 정보를 관리하는 메커니즘이 없어 정보가 최신 상태가 아닙니다.
- 누군가 누락된 정보를 식별하지만 팀 Wiki에 추가하도록 요청할 프로세스가 없습니다. 이를 직접 추가하지만 중요한 단계를 놓쳐 중단으로 이어집니다.

이 모범 사례 확립의 이점:

- 정보가 자유롭게 공유되기 때문에 팀원의 역량이 강화됩니다.
- 문서가 최신 상태이고 검색 가능하기 때문에 새로운 팀 구성원이 더 빨리 온보딩됩니다.
- 정보는 시의적절하고 정확하며 실행 가능합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 높음

구현 가이드

지식 관리란 학습하는 조직의 중요한 측면입니다. 시작하려면 지식을 저장할 중앙 리포지토리가 필요합니다(일반적인 예: 셀프 호스팅된 Wiki). 지식을 추가, 업데이트 및 보관하는 프로세스를 마련해야 합니다. 문서화해야 하는 항목에 대한 표준을 개발하고 모든 사람이 기여하도록 합니다.

고객 사례

AnyCompany Retail은 모든 지식이 저장되는 내부 Wiki를 호스팅합니다. 팀원은 일상 업무를 수행하면서 지식 베이스에 추가하도록 권장됩니다. 다기능 팀은 분기별로 가장 적게 업데이트된 페이지를 평가하고 아카이브할지 또는 업데이트할지 결정합니다.

구현 단계

1. 먼저 지식이 저장될 콘텐츠 관리 시스템을 식별합니다. 조직 전체의 이해 관계자로부터 동의를 얻습니다.
 - a. 기존 콘텐츠 관리 시스템이 없는 경우 셀프 호스팅된 Wiki를 실행하거나 버전 관리 리포지토리에서 시작하는 것이 좋습니다.
2. 정보를 추가, 업데이트 및 보관하기 위한 런북을 개발합니다. 팀에 이러한 프로세스를 알려줍니다.
3. 콘텐츠 관리 시스템에 어떤 지식을 저장해야 하는지 식별합니다. 팀원이 수행하는 일상 업무(런북 및 플레이북)부터 시작합니다. 이해 관계자와 협력하여 추가되는 지식의 우선순위를 정합니다.
4. 주기적으로 이해 관계자와 협력하여 오래된 정보를 식별하여 아카이브하거나 최신 정보를 가져옵니다.

구현 계획의 작업 수준: 중간. 기존 콘텐츠 관리 시스템이 없는 경우 셀프 호스팅된 Wiki 또는 버전 관리 문서 리포지토리를 설정할 수 있습니다.

리소스

관련 모범 사례:

- [OPS11-BP08 파악한 내용 문서화 및 공유](#) - 지식 관리는 학습한 내용에 대한 정보 공유를 용이하게 합니다.

관련 문서:

- [Atlassian - 지식 관리](#)

관련 예시:

- [DokuWiki](#)
- [Gollum](#)
- [MediaWiki](#)
- [Wiki.js](#)

OPS11-BP05 개선 추진 요인 정의

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

- 개선 추진 요인 파악: 원하는 결과가 지원되는 경우에만 시스템을 변경해야 합니다.
 - 필요한 기능: 개선 기회를 평가할 때 필요한 기능을 평가합니다.
 - [AWS의 새로운 소식](#)

- 반드시 수정해야 할 문제: 개선 기회를 평가할 때 반드시 수정해야 할 문제, 버그 및 취약성을 평가합니다. 걱정 크기 조정 옵션을 추적하고 최적화 기회를 모색합니다.
 - [AWS 최신 보안 공지](#)
 - [AWS Trusted Advisor](#)
 - [Cloud Intelligence Dashboards](#)
- 규정 준수 요건: 개선 기회를 검토할 때 규정/정책 준수 상태를 유지하거나 타사의 지원을 계속 받으려는 데 필요한 업데이트와 변경 사항을 평가합니다.
 - [AWS 규정 준수](#)
 - [AWS 규정 준수 프로그램](#)
 - [AWS 규정 준수 관련 최신 소식](#)

리소스

관련 모범 사례:

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

관련 문서:

- [Amazon Athena](#)
- [Amazon QuickSight](#)
- [AWS 규정 준수](#)
- [AWS 규정 준수 관련 최신 소식](#)
- [AWS 규정 준수 프로그램](#)
- [AWS Glue](#)
- [AWS 최신 보안 공지](#)
- [AWS Trusted Advisor](#)
- [Export your log data to Amazon S3](#)

- [AWS의 새로운 소식](#)
- [고객 중심 혁신의 필요성](#)
- [Digital Transformation: Hype or a Strategic Necessity?](#)

관련 동영상

- [AWS re:Invent 2023 - Improve operational efficiency and resilience with AWS Support \(SUP310\)](#)

OPS11-BP06 인사이트 검증

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

원하는 결과:

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

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

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

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

- 인사이트 검증: 비즈니스 소유자 및 주제별 전문가와 협력하여 수집한 데이터의 의미에 대한 공통된 이해와 합의가 있는지 확인합니다. 추가 우려 사항, 잠재적 영향을 식별하고 조치 과정을 결정합니다.

리소스

관련 모범 사례:

- [OPS01-BP06 장점과 위험을 관리하면서 절충점 평가](#)
- [OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임](#)
- [OPS11-BP03 피드백 루프 구현](#)

관련 문서:

- [Designing a Cloud Center of Excellence \(CCOE\)](#)

관련 동영상:

- [Building observability to increase resiliency](#)

OPS11-BP07 운영 지표 검토 수행

다양한 실무 영역의 여러 팀 구성원들과 함께 운영 지표 후행 분석을 정기적으로 수행합니다. 이러한 검토에서는 개선 기회와 진행 가능한 조치 과정을 파악하고 배운 내용을 공유할 수 있습니다. 개발, 테스트, 프로덕션 등 모든 환경에서 항상 기회를 모색해야 합니다.

원하는 결과:

- 비즈니스에 영향을 미치는 지표를 자주 검토합니다.
- 관찰성 기능을 통해 이상 징후를 탐지하고 검토합니다.
- 데이터를 사용하여 비즈니스 성과 및 목표를 지원합니다.

일반적인 안티 패턴:

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

이 모범 사례 확립의 이점:

- 정기적으로 만나 운영 지표, 이벤트 및 인시던트를 검토하면 팀 간에 공통된 이해를 유지할 수 있습니다.
- 팀은 정기적으로 회의를 통해 지표와 인시던트를 검토하며, 이를 통해 위험에 대한 조치를 취하고 고객 SLA를 인식할 수 있습니다.
- 파악한 내용을 공유하여 비즈니스 성과에 대한 우선순위 지정 및 목표 개선을 위한 데이터를 제공합니다.

이 모범 사례를 따르지 않을 경우 노출 위험도: 중간

구현 가이드

- 다양한 실무 영역의 여러 팀 구성원들과 함께 운영 지표 후행 분석을 정기적으로 수행합니다.
- 실무 팀, 개발 팀, 운영 팀 등의 이해관계자와 함께 즉각적인 피드백 및 후행 분석에서 발견된 사항을 확인하고 파악한 내용을 공유합니다.
- 그리고 이러한 인사이트를 활용하여 개선 기회와 진행 가능한 조치 과정을 확인합니다.

리소스

관련 모범 사례:

- [OPS08-BP05 대시보드 만들기](#)
- [OPS09-BP03 운영 지표 검토 및 개선 우선 순위 지정](#)
- [OPS10-BP01 이벤트, 인시던트 및 문제 관리 프로세스 사용](#)

관련 문서:

- [Amazon CloudWatch](#)
- [Amazon CloudWatch 지표 및 차원 참조](#)
- [사용자 지정 지표 게시](#)
- [Amazon CloudWatch 지표 사용](#)
- [Dashboards and visualizations with CloudWatch](#)

OPS11-BP08 파악한 내용 문서화 및 공유

운영 활동 과정에서 파악한 내용을 문서화하고 공유하여 내부적으로, 그리고 여러 팀 간에 사용할 수 있도록 하십시오. 조직 전체에서 관련 이점을 더욱 효율적으로 활용하려면 팀에서 파악한 내용을 공유해야 합니다. 피할 수 있는 오류를 방지하고 개발 작업을 쉽게 수행하기 위해 정보와 리소스를 공유하고 원하는 기능을 제공하는 데 집중하세요.

AWS Identity and Access Management(IAM)를 사용하여 계정 내에서도 계정 간에 공유할 리소스 액세스를 제어할 수 있는 권한을 정의합니다.

원하는 결과:

- 버전 관리 리포지토리를 사용하여 애플리케이션 라이브러리, 스크립팅된 절차, 절차 설명서 및 기타 시스템 설명서를 공유합니다.
- 인프라 표준을 버전이 관리되는 AWS CloudFormation 템플릿으로 공유합니다.
- 팀 전체에서 파악한 내용을 검토합니다.

일반적인 안티 패턴:

- 조직에서 일반적으로 버그가 있는 라이브러리를 사용하기 때문에 운영 중단이 오래 지속되었습니다. 이후 신뢰할 수 있는 라이브러리로 마이그레이션했습니다. 조직의 다른 팀들은 그들이 위험에 처

해 있다는 것을 알지 못합니다. 아무도 이 라이브러리에 대한 경험을 문서화하고 공유하지 않으며 위험을 인식하지 못합니다.

- 내부적으로 공유된 마이크로서비스에서 세션 종단을 일으키는 옛지 사례를 발견했습니다. 이 옛지 사례를 방지하기 위해 서비스에 대한 호출을 업데이트했습니다. 조직의 다른 팀들은 위험에 처해 있다는 것을 알지 못합니다.
- 마이크로서비스 중 하나에 대한 CPU 사용률 요구 사항을 크게 줄일 수 있는 방법을 찾았습니다. 다른 팀에서 이 기술을 활용할 수 있는지 여부는 알 수 없습니다.

이 모범 사례 확립의 이점: 파악한 내용을 공유하여 개선을 지원하고 경험에서 얻은 이점을 극대화합니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

- 파악한 내용 문서화 및 공유: 운영 활동을 실행하면서 파악한 내용과 후행 분석 결과를 문서화하는 절차를 마련하여 다른 팀에서도 사용할 수 있도록 합니다.
- 파악한 내용 공유: 파악한 내용 및 관련 아티팩트를 팀 간에 공유할 수 있는 절차를 마련합니다. 예를 들어, 접속 가능한 Wiki를 통해 새로워진 절차, 지침, 거버넌스 및 모범 사례를 공유합니다. 스크립트, 코드 및 라이브러리는 공동 리포지토리를 통해 공유할 수 있습니다.
 - [Delegating access to your AWS environment](#)
 - [AWS CodeCommit 리포지토리 공유](#)

리소스

관련 모범 사례:

- [OPS02-BP06 미리 정의되었거나 협상된 팀 간 책임](#)
- [OPS05-BP01 버전 관리 사용](#)
- [OPS05-BP06 설계 표준 공유](#)
- [OPS11-BP03 피드백 루프 구현](#)
- [OPS11-BP07 운영 지표 검토 수행](#)

관련 문서:

- [Reduce project delays with a docs-as-code solution](#)

관련 동영상:

- [Delegating access to your AWS environment](#)
- [AWS Supports You | Exploring the Incident Management Tabletop Exercise](#)

OPS11-BP09 개선을 위한 시간 할애

프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행합니다.

원하는 결과:

- 실험과 테스트의 위험, 작업량 및 비용을 줄일 수 있도록 환경의 임시 복제본을 생성합니다.
- 이렇게 복제된 환경을 사용하여 분석의 결론을 테스트하고, 실험을 진행하고, 계획된 향상 내용을 개발/테스트할 수 있습니다.
- 게임데이를 운영하고 결함 주입을 통해 팀이 프로덕션과 유사한 환경에서 실험을 실행하는 데 필요한 제어 및 가드레일을 제공합니다.

일반적인 안티 패턴:

- 애플리케이션 서버에 알려진 성능 문제가 있습니다. 이는 계획된 모든 기능 구현 뒤의 백로그에 추가됩니다. 추가되는 계획된 기능의 비율이 일정하게 유지되는 경우 성능 문제는 해결되지 않습니다.
- 지속적인 개선 지원을 위해 관리자 및 개발자가 개선 사항을 선택하고 구현하는 데 여분의 시간을 모두 할애하는 것을 승인합니다. 개선이 완료되지 않습니다.
- 운영 승인이 완료되었으며 운영 사례를 다시 테스트하지 않습니다.

이 모범 사례 확립의 이점: 프로세스 내에서 전담 리소스와 시간을 할애하여 가능한 범위 내에서 점진적 개선을 지속적으로 수행할 수 있습니다.

이 모범 사례를 따르지 않을 경우 노출되는 위험 수준: 낮음

구현 가이드

- 개선을 위한 시간 할애: 프로세스 내에서 전담 리소스와 시간을 할애하여 점진적 개선을 지속적으로 수행합니다.
- 변경 사항을 적용하여 결과를 개선하고, 평가를 통하여 성공 여부를 확인합니다.
- 결과가 목표에 미치지 못하지만 여전히 개선을 우선해야 한다면 다른 대안을 찾아서 진행합니다.

- 게임데이 내내 프로덕션 워크로드를 시뮬레이션하고 이러한 시뮬레이션에서 파악한 내용을 활용하여 개선합니다.

리소스

관련 모범 사례:

- [OPS05-BP08 여러 환경 사용](#)

관련 동영상:

- [AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service](#)

결론

운영 우수성은 지속적으로 반복해서 수행해야 하는 작업입니다.

목표를 공유하여 조직이 성공할 수 있도록 지원합니다. 모든 사람이 비즈니스 성과를 달성하는 데 자신의 역할을 파악하고 어떻게 다른 사람의 역량에 영향을 미치는지 이해해야 합니다. 비즈니스 성과를 달성할 수 있도록 팀원을 지원합니다.

모든 운영 이벤트와 장애는 아키텍처 운영을 개선할 수 있는 기회로 간주해야 합니다. 워크로드의 요구 사항을 파악하고, 일상적인 활동을 위한 런북과 문제 해결 과정을 안내하는 플레이북을 미리 정의합니다. 그리고, 코드 기능을 사용하여 AWS를 운영하고, 상황을 지속적으로 인지하면 운영 준비 역량을 높이고 인시던트가 발생한 경우 더 효과적으로 대응할 수 있습니다.

요구 사항이 바뀌면 우선순위와 이벤트 대응 및 후행 분석에서 파악한 내용에 따라 증분식 개선을 집중적으로 수행합니다. 이러한 작업을 통해, 활동의 효율성을 높여 비즈니스 성공을 달성할 수 있습니다.

AWS는 응답성이 뛰어난 적응형 배포를 구축하는 동시에 효율성을 최대화하는 아키텍처를 구축하고 운영할 수 있도록 지원합니다. 워크로드의 운영 우수성을 향상시키려면 이 백서에서 설명한 모범 사례를 사용해야 합니다.

기여자

- Rich Boyd, Operational Excellence Pillar Lead, Well-Architected, Amazon Web Services
- Jon Steele, Solutions Architect Well-Architected, Amazon Web Services
- Ryan King, Technical Program Manager, Amazon Web Services
- Chris Kunselman, Advisory Consultant, Amazon Web Services
- Peter Mullen, Advisory Consultant, Amazon Web Services
- Brian Quinn, Advisory Consultant, Amazon Web Services
- David Stanley, Cloud Operating Model Lead, Amazon Web Services
- Chris Kozlowski, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Alex Livingstone, Principal Specialist Solutions Architect, Cloud Operations, Amazon Web Services
- Paul Moran, Principal Technologist, Enterprise Support, Amazon Web Services
- Peter Mullen, Advisory Consultant, Professional Services, Amazon Web Services
- Chris Pates, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Arvind Raghunathan, Principal Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Ben Mergen, Senior Cost Lead Solutions Architect, Amazon Web Services

추가 자료

자세한 지침은 다음 출처를 참조하십시오.

- [AWS Well-Architected Framework](#)
- [AWS 아키텍처 센터](#)

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

| 변경 사항 | 설명 | 날짜 |
|----------------------------------|--|-------------------|
| 백서 업데이트 | 모범 사례를 새로운 구현 가이드와 함께 업데이트했습니다. | June 27, 2024 |
| 주요 콘텐츠 업데이트 및 통합 | <p>콘텐츠는 여러 모범 사례 영역에서 업데이트 및 통합되었습니다. 두 가지 모범 사례 영역 (OPS 04와 OPS 08)이 새로운 내용과 초점을 맞춰 다시 작성되었습니다.</p> <p>다음 영역에서 모범 사례가 업데이트 및 통합되었습니다. 운영을 고려한 설계, 배포 위험 완화 및 운영 상태 파악. 모범 사례 영역 OPS 04가 다음과 같이 업데이트되었습니다. 관찰성 구현. 모범 사례 영역 OPS 08이 다음과 같이 업데이트되었습니다. 워크로드 관찰성 활용.</p> | October 3, 2023 |
| 새 프레임워크 관련 업데이트 | 모범 사례를 권장 가이드와 함께 업데이트하고 새로운 모범 사례를 추가했습니다. | April 10, 2023 |
| 백서 업데이트 | 모범 사례를 새로운 구현 가이드와 함께 업데이트했습니다. | December 15, 2022 |
| 백서 업데이트 | 모범 사례를 확대하고 개선 계획을 추가했습니다. | October 20, 2022 |
| 마이너 업데이트 | 편집상 소규모 업데이트가 있었습니다. | August 8, 2022 |

| | | |
|--|---|------------------|
| <u>백서 업데이트</u> | 새로운 AWS 서비스 및 기능과 최신 모범 사례를 반영하여 업데이트되었습니다. | February 2, 2022 |
| <u>마이너 업데이트</u> | 소개에 지속 가능성 원칙을 추가했습니다. | December 2, 2021 |
| <u>새 프레임워크 관련 업데이트</u> | 새로운 AWS 서비스 및 기능과 최신 모범 사례를 반영하여 업데이트되었습니다. | July 8, 2020 |
| <u>백서 업데이트</u> | 최신 AWS 서비스 및 기능과 새로워진 참조를 반영하여 업데이트했습니다. | July 1, 2018 |
| <u>최초 게시</u> | 운영 우수성 원칙 - AWS Well-Architected Framework를 게시했습니다. | November 1, 2017 |