



복원력 수명 주기 프레임워크

# AWS 권장 가이드



# AWS 권장 가이드: 복원력 수명 주기 프레임워크

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

# Table of Contents

소개 .....	1
용어 및 정의 .....	2
지속적 복원력 .....	3
1단계: 목표 설정 .....	4
중요한 애플리케이션 매핑 .....	4
사용자 스토리 매핑 .....	5
측정 정의 .....	5
추가 측정 생성 .....	6
2단계: 설계 및 구현 .....	7
AWS Well-Architected 프레임워크 .....	7
종속성 이해 .....	8
재해 복구 전략 .....	8
CI/CD 전략 정의 .....	9
ORR 수행 .....	10
AWS 장애 격리 경계 이해 .....	10
응답 선택 .....	10
복원력 모델링 .....	11
페일 세이프 .....	11
3단계: 평가 및 테스트 .....	12
배포 전 활동 .....	12
환경 설계 .....	12
통합 테스트하기 .....	12
자동화된 배포 파이프라인 .....	13
로드 테스트 .....	13
배포 후 활동 .....	14
복원력 평가 수행 .....	14
DR 테스트 .....	14
드리프트 감지 .....	14
가상 테스트 .....	15
카오스 엔지니어링 .....	15
4단계: 운영 .....	16
관찰성 .....	16
이벤트 관리 .....	16
지속적 복원력 .....	17

5단계: 대응 및 학습 .....	18
인시던트 분석 보고서 생성 .....	18
운영 검토 수행 .....	19
경보 성능 검토 .....	19
경보 정밀도 .....	20
거짓 긍정 .....	20
거짓 부정 .....	20
중복 알림 .....	20
지표 검토 수행 .....	20
교육 및 지원 제공 .....	21
인시던트 지식 기반 생성 .....	21
심층 복원력 구현 .....	22
결론 및 리소스 .....	23
기여자 .....	24
문서 기록 .....	25
용어집 .....	26
# .....	26
A .....	27
B .....	30
C .....	31
D .....	35
E .....	38
F .....	40
G .....	42
H .....	43
I .....	45
L .....	47
M .....	48
O .....	52
P .....	54
Q .....	57
R .....	57
S .....	60
T .....	64
U .....	65
V .....	66

---

W .....	66
Z .....	67
.....	lxix

# 복원력 수명 주기 프레임워크: 복원력 개선에 대한 지속적 접근 방식

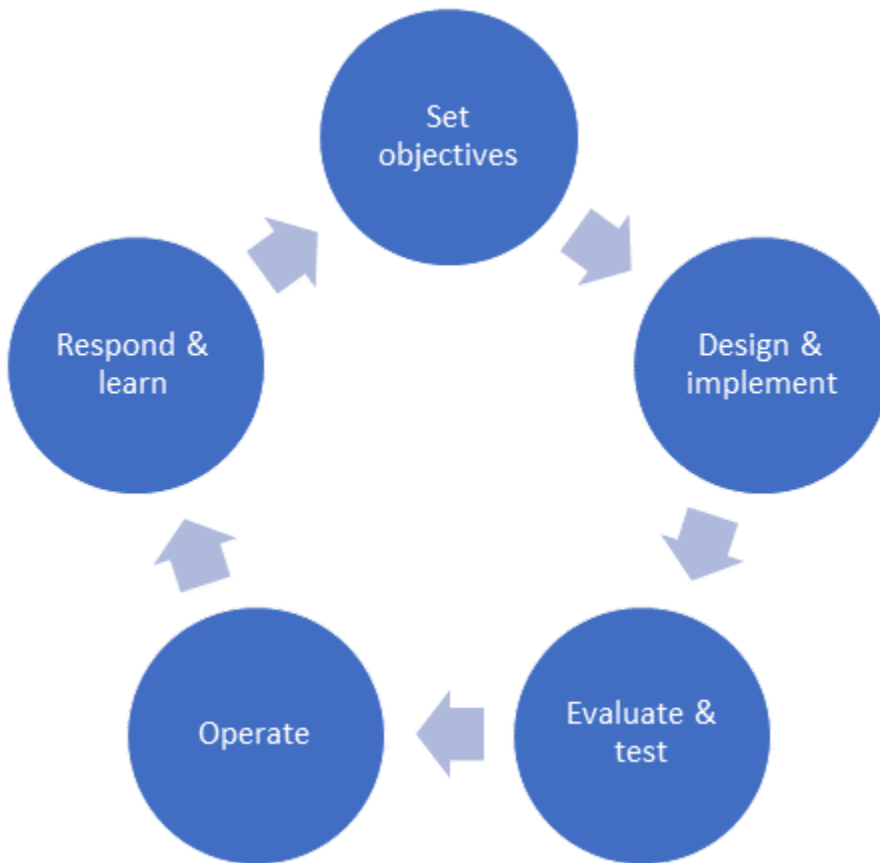
Amazon Web Services([기여자](#))

2023년 10월([문서 기록](#))

오늘날 현대 조직은 특히 고객의 기대치가 상시 가동 상시 가용 기반 사고방식으로 전환됨에 따라 지속적으로 증가하는 복원력 관련 문제에 직면하고 있습니다. 원격 팀과 복잡한 분산 애플리케이션의 경우 빈번한 릴리스의 필요성이 높아지고 있습니다. 따라서 조직과 애플리케이션은 그 어느 때보다 복원력이 뛰어나야 합니다.

AWS 는 복원력을 인프라, 종속 서비스, 잘못된 구성 및 일시적인 네트워크 문제와 관련된 중단에 저항하거나 복구하는 애플리케이션의 기능으로 정의합니다. ( AWS Well-Architected Framework 신뢰성 원칙 설명서 [의 복원력 및 신뢰성 구성](#) 요소를 참조하세요.) 그러나 원하는 수준의 복원력을 달성하려면 절충이 필요하곤 합니다. 그래서 운영 복잡성, 엔지니어링 복잡성 및 비용을 적절하게 평가하고 조정해야 합니다.

고객 및 내부 팀과 수년간 협력해 온 AWS 는 복원력 학습과 모범 사례를 캡처하는 복원력 수명 주기 프레임워크를 개발했습니다. 이 프레임워크는 다음 다이어그램에 설명된 5가지 주요 단계를 간략하게 설명합니다. 각 단계에서 전략, 서비스 및 메커니즘을 사용하여 복원력 태세를 개선할 수 있습니다.



이러한 단계는 이 안내서의 다음 섹션에서 다룹니다.

- [1단계: 목표 설정](#)
- [2단계: 설계 및 구현](#)
- [3단계: 평가 및 테스트](#)
- [4단계: 운영](#)
- [5단계: 대응 및 학습](#)

## 용어 및 정의

각 단계의 복원력 개념은 개별 구성 요소부터 전체 시스템에 이르기까지 다양한 수준에서 적용됩니다. 이러한 개념을 구현하려면 여러 용어를 명확하게 정의해야 합니다.

- 구성 요소는 기능을 수행하는 요소이며 소프트웨어 및 기술 리소스로 구성됩니다. 구성 요소의 예로, 코드 구성, 네트워킹 또는 서버, 데이터 저장소와 같은 인프라, 다중 인증(MFA) 디바이스와 같은 외부 종속성이 있습니다.

- 애플리케이션은 고객 대면 웹 스토어프론트 또는 기계 학습 모델을 개선하는 백엔드 프로세스와 같이 비즈니스 가치를 제공하는 구성 요소의 모음입니다. 애플리케이션은 단일 AWS 계정의 구성 요소 하위 세트로 구성되거나 여러 AWS 계정 및 리전에 분포된 여러 구성 요소의 모음일 수 있습니다.
- 시스템은 지정된 비즈니스 기능을 관리하는 데 필요한 애플리케이션, 사람 및 프로세스의 모음입니다. 여기에는 함수를 실행하는 데 필요한 애플리케이션, 지속적 통합 및 지속적 전송(CI/CD), 관찰성, 구성 관리, 인시던트 대응 및 재해 복구와 같은 운영 프로세스, 이러한 작업을 관리하는 운영자가 포함됩니다.
- 중단은 애플리케이션이 비즈니스 기능을 제대로 제공하지 못하게 하는 이벤트입니다.
- 장애는 중단이 완화되지 않는 경우 애플리케이션에 미치는 영향입니다. 일련의 중단이 발생하는 경우 애플리케이션 장애가 발생할 수 있습니다.

## 지속적 복원력

복원력 수명 주기는 지속적인 프로세스입니다. 동일한 조직 내에서도 애플리케이션 팀은 애플리케이션의 요구 사항에 따라 각 단계 내에서 다양한 수준의 완전성을 발휘할 수 있습니다. 그러나 각 단계가 완료에 가까워질수록 애플리케이션이 더 높은 수준의 복원력을 갖추게 됩니다.

복원력 수명 주기를 조직이 운영할 수 있는 표준 프로세스로 생각해야 합니다. AWS 는 애플리케이션을 개발하고 운영하는 동안 운영 프로세스 전반에 걸쳐 계획, 테스트 및 학습을 통합하는 것을 목표로 소프트웨어 개발 수명 주기(SDLC)와 유사하게 복원력 수명 주기를 의도적으로 모델링했습니다. 많은 애자일 개발 프로세스와 마찬가지로 개발 프로세스를 반복할 때마다 복원력 수명 주기를 반복할 수 있습니다. 수명 주기의 각 단계 내에서 시간이 지남에 따라 점진적으로 사례를 강화하는 것이 좋습니다.

## 1단계: 목표 설정

필요한 복원력 수준과 이를 측정하는 방법을 이해하는 것이 목표 설정 단계의 기반입니다. 목표가 없고 측정할 수 없다면 무언가를 개선하기 어렵습니다.

애플리케이션에 필요한 복원력 수준은 서로 다를 수 있습니다. 목표를 설정할 경우 올바른 투자와 장단점을 파악하려면 필요한 수준을 고려합니다. 자동차를 비유로 들어 설명할 수 있습니다. 타이어는 4개이지만 예비 타이어는 1개뿐입니다. 주행 중에 타이어에 고장이 날 가능성은 낮으며, 화물 공간 또는 연료 효율성과 같은 다른 기능에서 추가 예비 부품을 치울 수 있으므로 이는 합리적인 절충입니다.

목표를 정의한 후 이후 단계([2단계: 설계 및 구현](#), [4단계: 운영](#))에서 관찰성 제어를 구현하여 목표가 충족되고 있는지 파악합니다.

## 중요한 애플리케이션 매핑

복원력 목표를 정의할 때 기술적 대화로만 국한되어서는 안 됩니다. 대신 비즈니스 중심의 초점에서 시작하여 애플리케이션이 전달해야 하는 사항과 장애의 결과를 이해합니다. 그런 다음 비즈니스 목표에 대한 이러한 이해는 아키텍처, 엔지니어링 및 운영과 같은 영역으로 점차 확대됩니다. 사용자가 정의한 복원력 목표는 모든 애플리케이션에 적용될 수도 있지만, 목표 측정 방법은 애플리케이션의 기능에 따라 달라지는 경우가 많습니다. 비즈니스에 중요한 애플리케이션을 실행 중일 수 있으며, 이 애플리케이션이 손상되면 조직이 상당한 수익을 잃거나 평판에 해를 끼칠 수 있습니다. 또는 중요하지 않고 조직의 비즈니스 수행 능력에 부정적인 영향을 주지 않으면서 가동 중지 시간을 허용할 수 있는 다른 애플리케이션이 있을 수 있습니다.

예를 들어 소매 회사의 주문 관리 애플리케이션을 생각해 보세요. 주문 관리 애플리케이션의 구성 요소가 손상되어 제대로 실행되지 않으면 새 판매가 진행되지 않습니다. 이 소매 회사에는 건물 중 하나에 직원을 위한 커피숍도 있습니다. 커피숍에는 직원이 정적 웹 페이지에서 액세스할 수 있는 온라인 메뉴가 있습니다. 이 웹 페이지가 사용 불가능해지면 일부 직원이 불만 제기를 할 수 있지만 반드시 회사에 재정적 손해를 끼치는 것은 아닙니다. 이 예제를 기반으로 비즈니스는 주문 관리 애플리케이션에 대해 보다 적극적인 복원력 목표를 설정하기로 선택했지만 웹 애플리케이션의 복원력을 보장하기 위해 상당한 투자는 하지 않을 것입니다.

가장 중요한 애플리케이션, 가장 많은 노력을 기울일 부분, 절충할 부분을 파악하는 것은 프로덕션 환경에서 애플리케이션의 복원력을 측정할 수 있는 것만큼 중요합니다. 장애의 영향을 더 잘 이해하기 위해 [비즈니스 영향 분석\(BIA\)](#)을 수행할 수 있습니다. BIA는 중요한 비즈니스 애플리케이션을 식별 및 우선순위 지정하고, 잠재적 위험과 영향을 평가하며, 지원 종속성을 식별하는 체계적이고 체계적인 접근 방식을 제공합니다. BIA는 조직의 가장 중요한 애플리케이션에 대한 가동 중지 시간의 비용을 정량화하는 데 도움이 됩니다. 이 지표는 특정 애플리케이션에 장애가 발생하여 기능을 완료할 수 없는 경우

비용이 얼마나 드는지 설명하는 데 도움이 됩니다. 이전 예제에서 주문 관리 애플리케이션에 장애가 발생한 경우 리테일 비즈니스는 상당한 수익을 잃을 수 있습니다.

## 사용자 스토리 매핑

BIA 프로세스 중에 애플리케이션이 둘 이상의 비즈니스 기능을 담당하거나 비즈니스 기능에 여러 애플리케이션이 필요함을 발견할 수 있습니다. 이전 리테일 회사 예제를 사용하면 주문 관리 기능에 결제, 프로모션 및 요금을 위한 별도의 애플리케이션이 필요할 수 있습니다. 하나의 애플리케이션에서 장애가 발생하면 비즈니스 및 이 회사와 상호 작용하는 사용자가 영향을 받을 수 있습니다. 예를 들어 회사에서는 새 주문 추가, 프로모션 및 할인에 대한 액세스 제공 또는 제품 가격 업데이트와 같은 작업을 수행하지 못할 수 있습니다. 주문 관리 기능에 필요한 이러한 다양한 기능은 여러 애플리케이션에 의존할 수 있습니다. 이러한 기능에는 여러 외부 종속성이 있을 수 있으므로 순전히 구성 요소 중심의 복원력만 달성하려는 프로세스는 너무 복잡해질 수 있습니다. 이 시나리오를 처리하는 더 나은 방법은 사용자가 하나의 애플리케이션 또는 일련의 애플리케이션과 상호 작용할 때 기대하는 경험을 설명하는 [사용자 스토리](#)에 초점을 맞추는 것입니다.

사용자 스토리에 집중하면 고객 경험의 어떤 부분이 가장 중요한지 이해하는 데 도움이 되므로 특정 위협으로부터 보호하는 메커니즘을 빌드할 수 있습니다. 이전 예제에서 한 가지 사용자 스토리는 체크아웃일 수 있으며, 이는 체크아웃 애플리케이션과 관련이 있고 요금 애플리케이션에 종속되어 있습니다. 또 다른 사용자 스토리로, 홍보 애플리케이션과 관련된 홍보 표시가 있습니다. 가장 중요한 애플리케이션과 사용자 스토리를 매핑한 후 이러한 사용자 스토리의 복원력을 측정하는 데 사용할 지표를 정의할 수 있습니다. 이러한 지표는 전체 포트폴리오 또는 개별 사용자 스토리에 적용할 수 있습니다.

## 측정 정의

[목표 복구 시점\(RPO\)](#), [목표 복구 시간\(RTO\)](#) 및 [서비스 수준 목표\(SLO\)](#)는 특정 시스템의 복원력을 평가하는 데 사용되는 표준 업계 측정입니다. RPO는 장애 발생 시 비즈니스에서 허용할 수 있는 데이터 손실의 양을 나타내는 반면, RTO는 중단 후 애플리케이션을 다시 사용할 수 있어야 하는 속도를 측정한 것입니다. 이 두 지표는 초, 분, 시간의 단위로 측정됩니다. 또한 애플리케이션이 제대로 작동하는 시간을 측정할 수 있습니다. 즉, 애플리케이션이 설계된 대로 기능을 수행하고 해당 사용자에게 액세스할 수 있습니다. 이러한 SLO는 고객이 받게 될 예상 서비스 수준을 자세히 설명하고 1초 미만의 응답 시간 내에 오류 없이 지원되는 요청의 백분율(%)과 같은 지표로 측정됩니다(예: 요청의 99.99%가 매월 응답을 수신함). RPO 및 RTO는 백업 복원부터 사용자 트래픽 리디렉션에 이르기까지 애플리케이션 운영 및 복구 프로세스가 중단될 것이라고 가정할 때 재해 복구 전략과 관련이 있습니다. SLO는 고가용성 제어를 구현하여 처리됩니다. 이는 애플리케이션의 가동 중지 시간을 줄이는 경향이 있습니다.

SLO 지표는 일반적으로 서비스 공급자와 최종 사용자 간의 계약인 서비스 수준 계약(SLA)의 정의에 사용됩니다. SLA는 일반적으로 재무 약정과 함께 제공되며 이러한 계약이 충족되지 않는 경우 제공업

체에서 지불해야 하는 벌금을 간략하게 설명합니다. 그러나 SLA는 복원력 태세를 측정하는 것이 아니며, SLA를 늘리더라도 애플리케이션의 복원력이 향상되지 않습니다.

처음에 SLO, RPO 및 RTO를 기반으로 목표를 설정할 수 있습니다. 복원력 목표를 정의하고 RPO 및 RTO 목표를 명확하게 이해한 후 [AWS Resilience Hub](#)를 사용하여 아키텍처 평가를 실행해 잠재적 복원력 관련 약점을 발견할 수 있습니다. AWS Resilience Hub는 AWS Well-Architected Framework 모범 사례를 기준으로 애플리케이션 아키텍처를 평가하고 정의된 RTO 및 RPO 목표를 충족하기 위해 특별히 개선해야 할 사항의 맥락에서 문제 해결 지침을 공유합니다.

## 추가 측정 생성

RPO, RTO 및 SLO는 복원력의 좋은 지표이지만 비즈니스 관점에서 목표에 대해 생각하고 애플리케이션 기능에 대한 목표를 정의할 수도 있습니다. 예를 들어 목표는 다음과 같습니다. 프론트엔드와 백엔드 간의 지연 시간이 40% 증가하면 분당 성공적인 주문은 98% 이상으로 유지됩니다. 또는: 특정 구성 요소가 손실되더라도 초당 시작된 스트림은 평균에서 표준 편차 이내로 유지됩니다. 또한 목표를 생성하여 알려진 장애 유형에서 평균 복구 시간(MTTR)을 줄일 수 있습니다. 예를 들어 이러한 알려진 문제가 발생하면 복구 시간이 x% 단축됩니다. 비즈니스 요구 사항에 맞는 목표를 생성하면 애플리케이션이 허용해야 하는 장애 유형을 예측하는 데 도움이 됩니다. 또한 애플리케이션 장애 가능성을 줄이기 위한 접근 방식을 식별하는 데도 도움이 됩니다.

애플리케이션에 전원을 공급하는 인스턴스의 5%를 손실한 경우 계속 운영하는 것이 목표라고 생각하면 애플리케이션을 사전 확장해야 하거나 해당 이벤트 중에 발생하는 추가 트래픽을 지원할 수 있을 만큼 빠르게 확장할 수 있어야 한다고 판단할 수 있습니다. 또는 [2단계: 설계 및 구현](#) 섹션에 설명된 대로 여러 아키텍처 패턴을 활용해야 한다고 결정할 수 있습니다.

또한 특정 비즈니스 목표에 대한 관찰성 측정을 구현해야 합니다. 예를 들어 평균 주문률, 평균 주문 가격, 평균 구독 수 또는 애플리케이션의 동작에 따라 비즈니스 상태에 대한 인사이트를 제공할 수 있는 기타 지표를 추적할 수 있습니다. 애플리케이션에 대한 관찰성 기능을 구현하면 이러한 지표가 정의된 경계를 초과하는 경우 경보를 생성하고 조치를 취할 수 있습니다. 관찰성은 [4단계: 운영](#) 섹션에서 자세히 다룹니다.

## 2단계: 설계 및 구현

이전 단계에서는 복원력 목표를 설정합니다. 이제 설계 및 구현 단계에서는 이전 단계에서 설정한 목표에 따라 장애 모드를 예측하고 설계 선택을 식별하려고 시도합니다. 또한 변경 관리를 위한 전략을 정의하고 소프트웨어 코드 및 인프라 구성을 개발합니다. 다음 섹션에서는 비용, 복잡성 및 운영 오버헤드와 같은 장단점을 고려하면서 고려해야 할 AWS 모범 사례를 강조합니다.

### AWS Well-Architected 프레임워크

원하는 복원력 목표를 기반으로 애플리케이션을 설계할 때 여러 요인을 평가하고 가장 최적의 아키텍처에서 장단점을 확인해야 합니다. 복원력이 뛰어난 애플리케이션을 빌드하려면 설계, 빌드 및 배포, 보안 및 운영 측면을 고려해야 합니다. [AWS Well-Architected Framework](#)는 AWS에서 복원력이 뛰어난 애플리케이션을 설계하는 데 도움이 되는 일련의 모범 사례, 설계 원칙 및 아키텍처 패턴을 제공합니다. AWS Well-Architected Framework의 여섯 가지 원칙에서는 복원력, 보안, 효율성, 비용 효율성, 지속 가능성을 지원하는 시스템을 설계하고 운영하는 모범 사례를 제공합니다. 이 프레임워크는 모범 사례를 기준으로 아키텍처를 일관된 방식으로 측정하고 개선할 영역을 식별하는 방법을 제공합니다.

다음은 AWS Well-Architected Framework가 복원력 목표를 충족하는 애플리케이션을 설계하고 구현하는 데 도움이 되는 방법에 대한 예제입니다.

- 신뢰성 원칙: [신뢰성 원칙](#)에서는 장애 또는 중단 중에도 정확하고 일관되게 작동할 수 있는 애플리케이션을 빌드하는 것의 중요성을 강조합니다. 예를 들어 AWS Well-Architected Framework에서는 마이크로서비스 아키텍처를 사용하여 애플리케이션을 더 작고 간단하게 만들어 애플리케이션 내 여러 구성 요소의 가용성 요구 사항을 차별화할 수 있습니다. 또한 스로틀링, 지수 백오프를 사용한 재시도, 빠른 실패(로드 세이딩), 멍등성, 일정한 작업, 회로 차단기 및 정적 안정성을 사용하여 애플리케이션 빌드 모범 사례에 대한 자세한 설명을 찾을 수 있습니다.
- 포괄적인 검토: AWS Well-Architected Framework는 모범 사례 및 설계 원칙을 기준으로 아키텍처를 포괄적으로 검토할 것을 권장합니다. 그리고 아키텍처를 일관되게 측정하고 개선이 필요한 영역을 식별할 수 있는 방법을 제공합니다.
- 위험 관리: AWS Well-Architected Framework는 애플리케이션의 신뢰성에 영향을 미칠 수 있는 위험을 식별하고 관리하는 데 도움이 됩니다. 잠재적 장애 시나리오를 선제적으로 처리하면 장애의 가능성이나 이로 인한 장애를 줄일 수 있습니다.
- 지속적인 개선: 복원력은 지속적인 프로세스이며 AWS Well-Architected Framework는 지속적인 개선을 강조합니다. AWS Well-Architected Framework의 지침에 따라 아키텍처와 프로세스를 정기적으로 검토하고 세부 조정하면 진화하는 과제와 요구 사항에 직면하여 시스템이 복원력을 유지할 수 있습니다.

## 종속성 이해

시스템의 종속성을 이해하는 것이 복원력의 핵심입니다. 종속성에는 애플리케이션 내 구성 요소 간 연결과 서드 파티 API 및 비즈니스 소유 공유 서비스와 같은 애플리케이션 외부 구성 요소에 대한 연결이 포함됩니다. 이러한 연결을 이해하면 중단을 격리하고 관리하는 데 도움이 됩니다. 한 구성 요소의 장애가 다른 구성 요소에 영향을 미칠 수 있기 때문입니다. 이 지식은 엔지니어가 장애의 영향을 평가하고 적절히 계획하며 리소스가 효과적으로 사용되는지 확인하는 데 도움이 됩니다. 종속성을 이해하면 대체 전략을 생성하고 복구 프로세스를 조정하는 데 도움이 됩니다. 또한 하드 종속성을 소프트 종속성으로 대체할 수 있는 사례를 결정하는 데도 도움이 됩니다. 이를 통해 종속성 장애가 발생할 경우에도 애플리케이션이 비즈니스 기능을 계속 지원할 수 있습니다. 종속성은 로드 밸런싱 및 애플리케이션 조정에 대한 의사 결정에도 영향을 미칩니다. 종속성을 이해하면 잠재적인 위험과 영향을 결정하는 데 도움이 될 수 있으므로 애플리케이션을 변경할 때 매우 중요합니다. 이 지식은 안정적이고 복원력이 뛰어난 애플리케이션을 생성하여 장애 관리, 영향 평가, 장애 복구, 로드 밸런싱, 규모 조정 및 변경 관리를 지원하는 데 도움이 됩니다. 종속성을 수동으로 추적하거나 [AWS X-Ray](#)와 같은 도구 및 서비스를 사용하여 분산 애플리케이션의 종속성을 이해할 수 있습니다.

## 재해 복구 전략

재해 복구(DR) 전략은 주로 비즈니스 연속성을 보장하여 복원력이 뛰어난 애플리케이션을 빌드하고 운영하는 데 중요한 역할을 합니다. 이를 통해 치명적인 이벤트 중에도 중요한 비즈니스 운영이 최소한의 장애로 유지될 수 있으므로 가동 중지 시간과 잠재적 수익 손실을 최소화할 수 있습니다. DR 전략은 종종 여러 위치에 걸쳐 정기적인 데이터 백업 및 데이터 복제를 통합하므로 데이터 보호에 필수적입니다. 이를 통해 중요한 비즈니스 정보를 보호하고 재해 발생 시 전체 손실을 방지하는 데 도움이 됩니다. 또한 많은 산업이 민감한 데이터를 보호하고 재해 중에도 서비스를 계속 사용할 수 있도록 DR 전략을 마련해야 하는 정책의 규정이 적용됩니다. DR 전략은 서비스 장애를 최소화하여 고객의 신뢰와 만족도를 높입니다. 잘 구현되고 자주 연습된 DR 전략은 재해 발생 후 복구 시간을 줄이고 애플리케이션을 빠르게 온라인 상태로 되돌리는 데 도움이 됩니다. 더 나아가 재해는 가동 중지 시간으로 인한 수익 손실뿐만 아니라 애플리케이션 및 데이터 복원 비용에서도 상당한 비용을 발생시킬 수 있습니다. 잘 설계된 DR 전략은 이러한 재정적 손실을 방지하는 데 도움이 됩니다.

어떤 전략을 선택하는지는 애플리케이션의 특정 요구 사항, RTO 및 RPO, 예산에 따라 달라집니다. [AWS Elastic Disaster Recovery](#)는 온프레미스 및 클라우드 기반 애플리케이션 모두에서 DR 전략을 구현하는 데 사용할 수 있는 목적별 복원력 서비스입니다.

자세한 내용은 AWS 웹 사이트의 [Disaster Recovery of Workloads on AWS](#) 및 [AWS Multi-Region Fundamentals](#)를 참조하세요.

## CI/CD 전략 정의

애플리케이션 장애의 일반적인 원인 중 하나는 이전에 알려진 작동 상태에서 애플리케이션을 변경하는 코드 또는 기타 변경 사항입니다. 변경 관리를 신중하게 다루지 않으면 장애가 자주 발생할 수 있습니다. 변경 빈도는 영향을 미칠 가능성을 높입니다. 그러나 변경 빈도를 줄이면 변경 세트가 커져 복잡성이 높아지고 장애가 발생할 가능성이 훨씬 더 커집니다. 지속적 통합 및 지속적 전송(CI/CD) 사례는 빈번한 작은 변경 사항(생산성 향상)을 유지하는 동시에 자동화를 통해 각 변경 사항에 높은 수준의 검사를 적용하도록 설계되었습니다. 몇 가지 기본 전략은 다음과 같습니다.

- **전체 자동화:** CI/CD의 기본 개념은 빌드 및 배포 프로세스를 최대한 자동화하는 것입니다. 여기에는 빌드, 테스트, 배포 및 모니터링도 포함됩니다. 자동화된 파이프라인은 인적 오류 가능성을 줄이고, 일관성을 보장하며, 프로세스의 신뢰성과 효율성을 보다 개선하는 데 도움이 됩니다.
- **테스트 기반 개발(TDD):** 애플리케이션 코드를 작성하기 전에 테스트를 작성합니다. 이 사례에서는 모든 코드에 테스트가 연결되어 있으므로 코드의 신뢰성과 자동화된 검사의 품질이 향상됩니다. 이러한 테스트는 CI 파이프라인에서 실행되어 변경 사항을 검증합니다.
- **빈번한 커밋 및 통합:** 개발자가 자주 코드를 커밋하고 자주 통합을 수행하도록 장려합니다. 작지만 자주 변경하면 테스트 및 디버깅이 더 쉬워져 심각한 문제의 위험이 줄어듭니다. 자동화는 각 커밋 및 배포 비용을 줄여 빈번한 통합을 가능하게 합니다.
- **변경 불가능한 인프라:** 서버 및 변경 불가능한 정적 엔터티와 같은 기타 인프라 구성 요소를 처리합니다. 인프라를 최대한 수정하는 대신 대체하고, 테스트되고 파이프라인을 통해 배포된 코드를 통해 새 인프라를 빌드합니다.
- **롤백 메커니즘:** 문제가 발생할 경우 항상 쉽고 안정적이며 자주 테스트된 방법으로 변경 사항을 롤백할 수 있습니다. 배포 안전을 위해서는 이전에 알려진 정상 상태로 빠르게 돌아갈 수 있어야 합니다. 이전 상태로 되돌릴 수 있는 간단한 버튼과 같습니다. 아니면 경보에 의해 완전히 자동화되어 시작될 수 있습니다.
- **버전 제어:** 버전 제어 리포지토리에서 모든 애플리케이션 코드, 구성 및 코드형 인프라를 유지 관리합니다. 이 사례는 변경 사항을 쉽게 추적하고 필요한 경우 되돌리는 데 도움이 됩니다.
- **카나리 배포 및 블루/그린 배포:** 먼저 인프라의 하위 세트에 새 버전의 애플리케이션을 배포하거나 두 환경(블루/그린)을 유지 관리하면 프로덕션에서 변경 사항의 동작을 확인하고 필요한 경우 빠르게 롤백할 수 있습니다.

CI/CD는 도구뿐만 아니라 문화에도 적용됩니다. 자동화, 테스트 및 실패를 통한 학습을 중시하는 문화를 조성하는 것은 올바른 도구와 프로세스를 구현하는 것만큼 중요합니다. 롤백은 최소한의 영향으로 매우 빠르게 수행되는 경우 실패가 아니라 학습 경험으로 간주해야 합니다.

## ORR 수행

운영 준비 상태 검토(ORR)는 운영 및 절차상의 격차를 식별하는 데 도움이 됩니다. Amazon에서는 수십 년간의 대규모 서비스 운영에서 학습한 내용을 모범 사례 지침에 따라 선별된 질문으로 나누는 ORR을 만들었습니다. ORR은 이전에 학습한 내용을 캡처하고 새 팀이 애플리케이션에서 이 학습한 내용을 고려했는지 확인해야 합니다. ORR은 아래 복원력 모델링 섹션에 설명된 복원력 모델링 활동에 포함될 수 있는 장애 모드 또는 장애 원인 목록을 제공할 수 있습니다. 자세한 내용은 AWS Well-Architected Framework 웹 사이트의 [Operational Readiness Reviews \(ORRs\)](#)를 참조하세요.

## AWS 장애 격리 경계 이해

AWS에서는 복원력 목표를 달성하는 데 도움이 되는 여러 장애 격리 경계를 제공합니다. 이러한 경계를 사용하여 제공하는 예측 가능한 영향 억제 범위를 활용할 수 있습니다. 애플리케이션에 대해 선택한 종속성을 의도적으로 선택할 수 있도록 이러한 경계를 사용하여 AWS 서비스를 설계하는 방법을 숙지해야 합니다. 애플리케이션에서 경계를 사용하는 방법을 이해하려면 AWS 웹 사이트의 [AWS Fault Isolation Boundaries](#)를 참조하세요.

## 응답 선택

시스템은 경보에 다양한 방식으로 대응할 수 있습니다. 일부 경보는 운영 팀의 응답이 필요한 반면, 애플리케이션 내에서 자가 복구 메커니즘을 트리거하는 경보가 있을 수도 있습니다. 자동화 비용을 제어하거나 엔지니어링 제약 조건을 관리하기 위해 수동 작업으로 자동화될 수 있는 응답을 유지하기로 결정할 수도 있습니다. 경보에 대한 응답 유형은 응답 구현 비용, 경보의 예상 빈도, 경보의 정확도, 경보에 전혀 응답하지 않는 경우 잠재적 비즈니스 손실의 함수로 선택될 가능성이 큼니다.

예를 들어 서버 프로세스가 충돌하면 운영 체제에서 프로세스를 다시 시작하거나, 새 서버를 프로비저닝하고 이전 서버를 종료하거나, 운영자에게 서버에 원격으로 연결하고 다시 시작하도록 지시할 수 있습니다. 이러한 응답은 애플리케이션 서버 프로세스를 다시 시작하는 것과 결과는 동일하지만 구현 및 유지 관리 비용이 서로 다를 수 있습니다.

### Note

심층적인 복원력 접근 방식을 취하기 위해 여러 응답을 선택할 수 있습니다. 예를 들어 이전 시나리오에서 애플리케이션 팀은 각 응답 사이에 시간 지연을 두고 세 가지 응답을 모두 구현하도록 선택할 수 있습니다. 장애가 발생한 서버 프로세스 지표가 30초 후에도 여전히 경보 상태인 경우 팀은 운영 체제가 애플리케이션 서버를 다시 시작하지 못했다고 가정할 수 있습니다. 따라서 Auto Scaling 그룹을 생성하여 새 가상 서버를 생성하고 애플리케이션 서버 프로세스

를 복원할 수 있습니다. 300초 후에도 지표가 여전히 경보 상태인 경우 운영 담당자에게 알림이 전송되어 원래 서버에 연결하고 프로세스를 복원하려고 시도할 수 있습니다.

애플리케이션 팀과 비즈니스에서 선택하는 응답은 엔지니어링 시간에 대한 선결제 투자로 운영 오버헤드를 상쇄하는 비즈니스의 선호도를 반영해야 합니다. 각 응답 옵션의 제약 조건과 예상되는 유지 관리를 신중하게 고려하여 정적 안정성과 같은 아키텍처 패턴, 회로 차단기와 같은 소프트웨어 패턴 또는 운영 절차에 해당하는 응답을 선택해야 합니다. 애플리케이션 팀을 안내하기 위한 일부 표준 응답이 존재할 수 있으므로 중앙 집중식 아키텍처 함수에서 관리하는 라이브러리와 패턴을 이 고려 사항에 대한 입력으로 사용할 수 있습니다.

## 복원력 모델링

복원력 모델링은 애플리케이션이 예상되는 다양한 중단에 대응하는 방법을 문서화합니다. 팀은 중단을 예측하여 관찰성, 자동화된 제어 및 복구 프로세스를 구현해 중단에도 불구하고 장애를 완화하거나 방지할 수 있습니다. AWS는 [복원력 분석 프레임워크](#)를 사용하여 복원력 모델을 개발하기 위한 지침을 만들었습니다. 이 프레임워크는 중단과 이러한 중단이 애플리케이션에 미치는 영향을 예측하는 데 도움이 될 수 있습니다. 중단을 예측하면 복원력이 뛰어나고 신뢰할 수 있는 애플리케이션을 빌드하는데 필요한 완화 조치를 식별할 수 있습니다. 복원력 분석 프레임워크를 사용하여 애플리케이션 수명 주기가 반복될 때마다 복원력 모델을 업데이트하는 것이 좋습니다. 각 반복에서 이 프레임워크를 사용하면 설계 단계 중에 중단을 예상하고 프로덕션 배포 후 애플리케이션을 테스트하여 인시던트를 줄일 수 있습니다. 이 프레임워크를 사용하여 복원력 모델을 개발하면 복원력 목표를 충족할 수 있습니다.

## 페일 세이프

중단을 방지할 수 없는 경우 안전하게 실패하세요. 심각한 비즈니스 손실이 발생하지 않는 기본 페일 세이프 운영 모드로 애플리케이션을 생성하는 방법을 고려합니다. 데이터베이스의 페일 세이프 상태의 예로, 기본적으로 사용자가 데이터를 생성하거나 변경할 수 없는 읽기 전용 작업이 있습니다. 데이터의 민감도에 따라 애플리케이션이 종료 상태로 기본 설정되고 읽기 전용 쿼리를 수행하지 않도록 할 수도 있습니다. 애플리케이션의 페일 세이프 상태를 고려하고 극한 조건에서는 이 작동 모드를 기본값으로 설정합니다.

## 3단계: 평가 및 테스트

수명 주기의 평가 및 테스트 단계에서는 애플리케이션 또는 기존 애플리케이션의 변경 사항이 설계되었지만 아직 프로덕션으로 릴리스되지 않습니다. 이 단계에서는 이전 단계에서 수행된 사례를 테스트하고 결과를 평가하는 활동을 구현합니다. 애플리케이션은 여전히 활성 개발 상태이거나 기본 개발이 완료되어 프로덕션으로 릴리스되기 전에 테스트 중일 수 있습니다. 이 단계에서는 애플리케이션이 정의된 복원력 목표를 충족할 것이라는 기대치를 확인하거나 반박하는 테스트를 개발하고 실행하는 데 중점을 둡니다. 또한 시스템의 운영 절차를 개발하고 테스트합니다. [2단계: 설계 및 구현](#) 단계에서 개발한 배포 절차가 실행되고 해당 결과를 평가합니다. 이러한 테스트 및 평가 활동은 수명 주기의 이 부분에서 시작되지만 여기에서 끝나지 않습니다. [4단계: 운영](#) 단계로 진행해도 테스트 및 평가는 계속됩니다.

평가 및 테스트 단계는 [사전 배포 활동](#)과 [사후 배포 활동](#)의 두 단계로 나뉩니다. 사전 배포 활동은 소프트웨어의 새 버전 배포와 테스트 환경으로의 초기 배포를 포함하여 애플리케이션을 환경에 배포하기 전에 완료해야 하는 태스크로 구성됩니다. 사후 배포 활동은 소프트웨어가 테스트 또는 프로덕션 환경에 배포된 후 수행됩니다. 다음 섹션에서는 이러한 단계에 대해 자세히 설명합니다.

### 배포 전 활동

#### 환경 설계

애플리케이션을 테스트하고 평가하는 환경은 애플리케이션을 얼마나 철저하게 테스트할 수 있는지 그리고 이러한 결과가 프로덕션 환경에서 나타나는 상황을 정확하게 반영한다는 확신 수준에 영향을 줍니다. Amazon DynamoDB와 같은 서비스를 사용하여 개발자 시스템에서 일부 통합 테스트를 로컬로 수행할 수 있습니다(DynamoDB 설명서의 [DynamoDB 로컬 설정](#) 참조). 그러나 어느 시점에서 결과에 가장 높은 신뢰도를 얻으려면 프로덕션 환경을 복제하는 환경에서 테스트해야 합니다. 이 환경에는 비용이 발생하므로 프로덕션과 유사한 환경이 파이프라인의 뒷부분에 나타나는 환경에 대해 단계적 또는 파이프라인 접근 방식을 취하는 것이 좋습니다.

#### 통합 테스트하기

통합 테스트는 애플리케이션의 잘 정의된 구성 요소가 외부 종속성과 함께 작동할 때 해당 기능을 올바르게 수행하는지 테스트하는 프로세스입니다. 이러한 외부 종속성은 다른 사용자 지정 개발 구성 요소, 애플리케이션에 사용하는 AWS 서비스, 타사 종속성 및 온프레미스 종속성일 수 있습니다. 이 가이드에서는 애플리케이션의 복원력을 보여주는 통합 테스트에 중점을 둡니다. 소프트웨어의 기능 정확도를 보여주는 유닛 테스트 및 통합 테스트가 이미 있다고 가정합니다.

회로 차단기 패턴 또는 로드 분할과 같이 사용자가 구현한 복원력 패턴을 구체적으로 테스트하는 통합 테스트를 설계하는 것이 좋습니다([2단계: 설계 및 구현](#) 참조). 복원력 지향 통합 테스트에는 종종 애플

리케이션에 특정 로드를 적용하거나 [AWS Fault Injection Service \(AWS FIS\)](#)와 같은 기능을 사용하여 의도적으로 환경에 중단을 초래하는 과정이 포함됩니다. CI/CD 파이프라인의 일부로 모든 통합 테스트를 실행하고 코드가 커밋될 때마다 테스트를 실행하는 것이 가장 좋습니다. 이를 통해 복원력 목표를 위반하는 코드나 구성의 변경 사항을 신속하게 감지하고 이에 대응할 수 있습니다. 대규모 분산 애플리케이션은 복잡하며 사소한 변경도 애플리케이션의 관련 없는 것처럼 보이는 부분의 복원력에 큰 영향을 미칠 수 있습니다. 모든 커밋에서 테스트를 실행해 보세요. CI/CD 파이프라인 및 기타 DevOps 도구를 운영하기 위한 우수한 도구 세트를 AWS 제공합니다. 자세한 내용은 AWS 웹 사이트의 [에서 DevOps 소개 AWS](#)를 참조하세요.

## 자동화된 배포 파이프라인

사전 프로덕션 환경에 배포하고 여기서 테스트하는 작업은 자동화에 가장 적합한 반복적이고 복잡한 태스크입니다. 이 프로세스를 자동화하면 인적 자원이 확보되고 오류 가능성이 줄어듭니다. 이 프로세스를 자동화하는 메커니즘을 종종 파이프라인이라고 합니다. 파이프라인을 생성할 때 프로덕션 구성에 점점 더 가까워지는 일련의 테스트 환경을 설정하는 것이 좋습니다. 이 일련의 환경을 사용하여 애플리케이션을 반복적으로 테스트합니다. 첫 번째 환경에서는 프로덕션 환경보다 더 제한된 기능 세트를 제공하지만 발생하는 비용은 훨씬 더 저렴합니다. 후속 환경에서는 서비스를 추가하고 프로덕션 환경을 더 가깝게 미러링하도록 조정해야 합니다.

먼저 첫 번째 환경에서 테스트합니다. 배포가 첫 번째 테스트 환경에서 모든 테스트를 통과한 후 일정 시간 동안 일정량의 로드로 애플리케이션을 실행하여 시간이 지남에 따라 문제가 발생하는지 확인합니다. 발생하는 문제를 감지할 수 있도록 관찰성을 올바르게 구성했는지 확인합니다(이 가이드의 뒷 부분에 있는 경보 정밀도 참조). 이 관찰 기간이 성공적으로 완료되면 애플리케이션을 다음 테스트 환경에 배포하고 프로세스를 반복하여 환경에서 지원하는 추가 테스트 또는 로드를 추가합니다. 이러한 방식으로 애플리케이션을 충분히 테스트한 후에는 이전에 애플리케이션을 프로덕션 환경에 배포하도록 설정한 배포 방법을 사용할 수 있습니다(이 가이드의 앞부분에서 CI/CD 전략 정의 참조). Amazon Builders' Library의 [Automating safe, hands-off deployments](#) 문서는 Amazon이 코드 배포를 자동화하는 방법을 설명하는 훌륭한 리소스입니다. 프로덕션 배포 이전의 환경 수는 애플리케이션의 복잡성과 애플리케이션의 종속성 유형에 따라 달라집니다.

## 로드 테스트.

표면적으로 로드 테스트는 통합 테스트와 유사합니다. 애플리케이션의 개별 함수와 외부 종속성을 테스트하여 예상대로 작동하는지 확인합니다. 그런 다음 로드 테스트는 통합 테스트를 넘어 잘 정의된 로드에서 애플리케이션이 작동하는 방식에 중점을 둡니다. 로드 테스트를 수행하려면 올바른 기능을 확인해야 하므로 통합 테스트에 성공한 후에 수행해야 합니다. 예상되는 로드에서 애플리케이션이 얼마나 잘 응답하는지와 로드가 기대치를 초과할 때 어떻게 작동하는지 이해하는 것이 중요합니다. 이렇게 하면 애플리케이션이 과도한 로드에서도 복원력을 유지하는 데 필요한 메커니즘을 구현했는지 확인할

수 있습니다. 로드 테스트에 대한 포괄적인 가이드는 AWS 솔루션 라이브러리의 [에서 분산 로드 테스트를 AWS](#) 참조하세요.

## 배포 후 활동

복원력은 지속적인 프로세스이며 애플리케이션이 배포된 후에도 애플리케이션의 복원력 평가를 계속 해야 합니다. 지속적인 복원력 평가와 같은 사후 배포 활동의 결과에 따라 복원력 수명 주기 초기에 수행한 복원력 활동 중 일부를 재평가하고 업데이트해야 할 수 있습니다.

## 복원력 평가 수행

애플리케이션을 프로덕션에 배포한 후에도 복원력 평가는 중단되지 않습니다. 잘 정의되고 자동화된 배포 파이프라인이 있더라도 프로덕션 환경에서 직접 변경이 발생할 수 있습니다. 또한 사전 배포 복원력 확인에서 아직 고려하지 않은 요인이 있을 수 있습니다. [AWS Resilience Hub](#)에서 배포된 아키텍처가 정의된 RPO 및 RTO 요구 사항을 충족하는지 평가할 수 있는 중앙 위치를 제공합니다. 이 서비스를 사용하면 미를 사용하여 애플리케이션 복원력을 지속적으로 평가하는 AWS 블로그 게시물에 설명된 대로 애플리케이션의 복원력에 대한 온디맨드 평가를 실행하고, 평가를 자동화하고, CI/CD 도구에 통합할 수 있습니다. [AWS Resilience Hub](#) [AWS CodePipeline](#) 이러한 평가를 자동화하는 것은 프로덕션 환경에서 복원력을 지속적으로 평가하는 데 도움이 되므로 모범 사례입니다.

## DR 테스트

**2단계: 설계 및 구현**에서는 시스템의 일부로 재해 복구(DR) 전략을 개발했습니다. 4단계 중에 DR 절차를 테스트하여 팀이 인시던트에 완전히 대비하고 절차가 예상대로 작동하는지 확인해야 합니다. 장애 조치 및 장애 복구를 포함한 모든 DR 절차를 정기적으로 테스트하고 각 연습의 결과를 검토하여 가능한 최상의 결과를 얻기 위해 시스템 절차를 업데이트해야 하는지와 그 방법을 결정해야 합니다. DR 테스트를 처음 개발할 때 테스트를 미리 예약하고 전체 팀이 예상되는 사항, 결과를 측정하는 방법, 결과를 기반으로 절차를 업데이트하는 데 사용할 피드백 메커니즘을 이해하는지 확인합니다. 예약된 DR 테스트를 실행하는 데 능숙해지면 발표되지 않은 DR 테스트를 실행하는 것이 좋습니다. 실제 재해는 일정에 따라 발생하지 않으므로 언제든지 계획을 실행할 준비를 해야 합니다. 그러나 발표되지 않았다고 해서 계획되지 않았다는 의미는 아닙니다. 주요 이해관계자는 여전히 이벤트를 계획하여 적절한 모니터링이 이루어지고 고객과 중요한 애플리케이션이 부정적인 영향을 받지 않도록 해야 합니다.

## 드리프트 감지

자동화 및 잘 정의된 절차가 마련되어 있어도 프로덕션 애플리케이션의 구성에 예상치 못한 변경이 발생할 수 있습니다. 애플리케이션 구성의 변경 사항을 감지하려면 기존 구성과의 편차를 나타내는 드리프트를 감지하는 메커니즘이 있어야 합니다. AWS CloudFormation 스택의 드리프트를 감지하는 방법

을 알아보려면 AWS CloudFormation 설명서의 [스택 및 리소스에 대한 비관리형 구성 변경 감지](#)를 참조하세요. 애플리케이션 AWS 환경에서 드리프트를 감지하려면 AWS Control Tower 설명서의 [에서 드리프트 감지 및 해결을 AWS Control Tower](#) 참조하세요.

## 가상 테스트

[가상 테스트](#)는 예약 기반으로 프로덕션에서 실행되는 구성 가능한 소프트웨어를 생성하여 최종 사용자 경험을 시뮬레이션하는 방식으로 애플리케이션의 API를 테스트하는 프로세스입니다. 이러한 테스트는 카나리라고도 합니다. 이 단어는 석탄 채굴 과정에서 사용하던 방식에서 가져왔습니다. 가상 테스트는 장애가 부분적이거나 간헐적인 경우에도 애플리케이션이 중단될 때 조기 경고를 제공하는 경우가 많으며, [회색 장애](#)의 경우도 많습니다.

## 카오스 엔지니어링

카오스 엔지니어링은 위험 완화 방식으로 애플리케이션을 의도적으로 중단 이벤트에 노출시키고, 대응을 면밀히 모니터링하며, 필요한 개선 사항을 구현하는 체계적인 프로세스입니다. 그 목적은 이러한 중단을 처리하는 애플리케이션의 능력에 대한 가정을 검증하거나 이의를 제기하는 것입니다. 카오스 엔지니어링은 이러한 이벤트를 우연히 떠나는 대신 엔지니어가 일반적으로 트래픽이 적은 기간에, 효과적인 완화를 위해 즉시 사용할 수 있는 엔지니어링 지원을 기반으로 제어된 환경에서 실험을 오케스트레이션할 수 있도록 지원합니다.

카오스 엔지니어링은 고려 중인 애플리케이션의 정상 작동 조건, 즉 안정 상태를 이해하는 데서 시작됩니다. 여기에서 중단 시 애플리케이션의 성공적인 동작을 자세히 설명하는 가설을 세웁니다. 그리고 실험을 실행합니다. 여기에는 네트워크 지연 시간, 서버 장애, 하드 드라이브 오류 및 외부 종속성 장애를 포함하되 이에 국한되지 않는 중단의 의도적인 주입이 포함됩니다. 그런 다음 실험 결과를 분석하고 학습한 내용을 기반으로 애플리케이션의 복원력을 개선합니다. 실험은 성능을 포함하여 애플리케이션의 다양한 측면을 개선하는 데 유용한 도구이며, 그렇지 않으면 숨겨져 있을 수 있는 잠재적 문제도 찾아냅니다. 또한 카오스 엔지니어링은 관찰성 및 경보 도구의 결함을 찾아내고 이를 구체화하는 데 도움이 됩니다. 또한 복구 시간을 줄이고 운영 기술을 개선하는 데에도 기여합니다. 카오스 엔지니어링은 모범 사례 채택을 가속화하고 지속적인 개선이라는 사고방식을 구축합니다. 궁극적으로 팀은 정기적인 연습과 반복을 통해 운영 기술을 구축하고 연마할 수 있습니다.

AWS에서는 비프로덕션 환경에서 카오스 엔지니어링 작업을 시작할 것을 권장합니다. [AWS Fault Injection Service \(AWS FIS\)](#)를 사용하여 범용 장애와 AWS에 고유한 장애로 카오스 엔지니어링 실험을 실행할 수 있습니다. 이 완전 관리형 서비스에는 중지 조건 경보와 전체 권한 제어가 포함되어 있으므로 안심하고 안전하게 카오스 엔지니어링을 쉽게 채택할 수 있습니다.

## 4단계: 운영

**3단계: 평가 및 테스트**를 완료하면 애플리케이션을 프로덕션에 배포할 준비가 된 것입니다. 운영 단계에서는 애플리케이션을 프로덕션에 배포하고 고객 경험을 관리합니다. 애플리케이션의 설계 및 구현에 따라 많은 복원력 결과가 결정되지만, 이 단계에서는 시스템이 복원력을 유지하고 개선하는 데 사용하는 운영 사례에 중점을 둡니다. 운영 우수성 문화를 구축하면 이러한 사례에서 표준과 일관성을 만드는 데 도움이 됩니다.

### 관찰성

고객 경험을 이해하는 데 가장 중요한 부분은 모니터링과 경보를 사용하는 것입니다. 애플리케이션을 계측하여 상태를 이해해야 하며 다양한 관점이 필요합니다. 즉, 일반적으로 카나리를 사용하여 서버 측과 클라이언트 측 모두에서 측정해야 합니다. 지표에는 애플리케이션의 종속성 및 **장애 격리 경계에 맞는 차원**과의 상호 작용에 대한 데이터가 포함되어야 합니다. 또한 애플리케이션에서 수행하는 모든 작업 단위에 대한 추가 세부 정보를 제공하는 로그를 생성해야 합니다. **Amazon CloudWatch 임베디드 지표 형식**과 같은 솔루션을 사용하여 지표와 로그를 결합하는 방법을 고려할 수 있습니다. 항상 더 많은 관찰성을 원할 수 있으므로 원하는 수준의 계측을 구현하는 데 필요한 비용, 노력 및 복잡성의 절충 효과를 고려합니다.

다음 링크는 애플리케이션을 계측하고 경보를 생성하기 위한 모범 사례를 제공합니다.

- [Monitoring production services at Amazon](#)(AWS re:Invent 2020 프레젠테이션)
- [Amazon Builders' Library: Operational Excellence at Amazon](#)(AWS re:Invent 2021 프레젠테이션)
- [Observability best practices at Amazon](#)(AWS re:Invent 2022 프레젠테이션)
- [Instrumenting distributed systems for operational visibility](#)(Amazon Builders' Library 문서)
- [Building dashboards for operational visibility](#)(Amazon Builders' Library 문서)

### 이벤트 관리

경보나 고객 측(상황이 더 좋지 않음)에서 문제가 발생했음을 알리는 경우 장애를 처리하기 위한 이벤트 관리 프로세스가 마련되어 있어야 합니다. 이 프로세스에는 통화 중인 운영자 참여, 문제 에스컬레이션, 인적 오류를 제거하는 데 도움이 되는 문제 해결에 대한 일관된 접근 방식을 위한 런북 수립이 포함되어야 합니다. 그러나 장애는 일반적으로 단독으로 발생하지 않습니다. 단일 애플리케이션은 장애에 종속되는 다른 여러 애플리케이션에 영향을 미칠 수 있습니다. 영향을 받는 모든 애플리케이션을 이해하고 한 번의 컨퍼런스 통화로 여러 팀의 운영자를 한데 모아 문제를 신속하게 해결할 수 있습니다. 그러나 조직의 규모와 구조에 따라 이 프로세스에 중앙 집중식 운영 팀이 필요할 수 있습니다.

이벤트 관리 프로세스를 설정하는 것 외에도 대시보드를 통해 지표를 정기적으로 검토해야 합니다. 정기 검토는 애플리케이션 성능의 고객 경험과 장기 추세를 이해하는 데 도움이 됩니다. 이를 통해 프로덕션에 상당한 영향을 미치기 전에 문제와 병목 현상을 식별할 수 있습니다. 일관되고 표준화된 방식으로 지표를 검토하면 상당한 이점을 얻을 수 있지만 하향식 동의와 시간 투자가 필요합니다.

다음 링크는 대시보드 빌드 및 운영 지표 검토에 대한 모범 사례를 제공합니다.

- [Building dashboards for operational visibility](#)(Amazon Builders' Library 문서)
- [Amazon's approach to failing successfully](#)(AWS re:Invent 2019 프레젠테이션)

## 지속적 복원력

2단계: 설계 및 구현 및 3단계: 평가 및 테스트 중에 애플리케이션을 프로덕션에 배포하기 전에 검토 및 테스트 활동을 시작했습니다. 운영 단계에서는 프로덕션 환경에서 이러한 활동을 계속 반복해야 합니다. [AWS Well-Architected Framework 검토](#), [운영 준비 상태 검토\(ORR\)](#) 및 [복원력 분석 프레임워크](#)를 통해 애플리케이션의 복원력 상태를 주기적으로 검토해야 합니다. 이렇게 하면 애플리케이션이 설정된 기준 및 표준에서 드리프트되지 않도록 하고 새 지침 또는 업데이트된 지침을 최신 상태로 유지할 수 있습니다. 이러한 지속적인 복원력 활동은 이전에 예상치 못한 중단을 찾아내고 새로운 완화 조치를 마련하는 데 도움이 됩니다.

사전 프로덕션 환경에서 성공적으로 실행한 후 프로덕션 환경에서 [게임 데이](#) 및 [카오스 엔지니어링](#) 실험을 실행하는 방법도 고려해 볼 수 있습니다. 게임 데이는 완화를 위해 복원력 메커니즘을 빌드한 알려진 이벤트를 시뮬레이션합니다. 예를 들어 게임 데이는 AWS 리전 서비스 장애를 시뮬레이션하고 다중 리전 장애 조치를 구현할 수 있습니다. 이러한 활동을 구현하려면 상당한 노력이 필요할 수 있지만 두 가지 방법 모두 시스템이 견딜 수 있도록 설계한 장애 모드에 복원력이 있다는 확신을 쌓는 데 도움이 됩니다.

애플리케이션을 운영하고, 운영 이벤트에 직면하며, 지표를 검토하고, 애플리케이션을 테스트하면 대응하고 배울 수 있는 많은 기회가 생깁니다.

## 5단계: 대응 및 학습

애플리케이션이 중단 이벤트에 대응하는 방식은 신뢰성에 영향을 줍니다. 경험을 통해 학습하고 이전에 애플리케이션이 중단에 대응한 방식도 신뢰성을 개선하는 데 중요합니다.

대응 및 학습 단계는 애플리케이션의 중단 이벤트에 더 잘 대응하기 위해 구현할 수 있는 사례에 중점을 둡니다. 또한 운영 팀 및 엔지니어의 경험을 바탕으로 최대한 학습할 수 있도록 지원하는 사례도 포함되어 있습니다.

### 인시던트 분석 보고서 생성

인시던트가 발생하면 고객과 비즈니스에 미치는 추가 피해를 최대한 빨리 방지하는 것이 첫 번째 조치입니다. 애플리케이션이 복구된 후 다음 단계는 발생한 상황을 이해하고 재발을 방지하기 위한 단계를 식별하는 것입니다. 이 사후 인시던트 분석은 일반적으로 애플리케이션 장애로 이어지는 일련의 이벤트와 중단이 애플리케이션, 고객 및 비즈니스에 미치는 영향을 문서화하는 보고서로 캡처됩니다. 이러한 보고서는 귀중한 학습 아티팩트가 되며 비즈니스 전반에 걸쳐 널리 공유되어야 합니다.

#### Note

비난하지 않으면서 인시던트 분석을 수행하는 것이 중요합니다. 모든 운영자가 자신이 가지고 있는 정보를 기반으로 가장 적합하고 최선의 조치를 취했다고 가정합니다. 보고서에 연산자 또는 엔지니어의 이름을 사용하지 마세요. 인적 오류를 장애의 원인으로 언급하면 팀원이 스스로를 보호하기 위해 방어적 자세를 취할 수 있게 되고 이때 정확하지 않거나 불완전한 정보가 캡처될 수 있습니다.

[Amazon 오류 수정\(COE\) 프로세스](#)에 문서화된 것과 같은 바람직한 인시던트 분석 보고서는 표준화된 형식을 따르며 애플리케이션 장애로 이어지는 조건을 최대한 자세히 캡처하려고 합니다. 이 보고서는 타임스탬프가 지정된 일련의 이벤트를 자세히 설명하고 타임라인 동안 애플리케이션의 측정 가능한 상태를 설명하는 정량적 데이터(종종 모니터링 대시보드의 지표 및 스크린샷)를 캡처합니다. 보고서는 조치를 취한 운영자와 엔지니어의 사고 프로세스와 이들을 결론으로 이끈 정보를 캡처해야 합니다. 또한 보고서에는 발생한 경보, 해당 경보가 애플리케이션 상태를 정확하게 반영했는지, 이벤트와 결과 경보 간의 지연 시간, 인시던트 해결 시간 등 다양한 지표의 성능도 자세히 설명되어 있어야 합니다. 또한 타임라인은 시작된 런북 또는 자동화와 애플리케이션이 유용한 상태를 다시 확보하는 데 어떻게 도움이 되었는지도 캡처합니다. 타임라인의 이러한 요소는 팀이 문제를 얼마나 빨리 해결했는지, 중단 완화에 얼마나 효과적인지를 포함하여 자동화된 응답과 운영자 응답의 효과를 이해하는 데 도움이 됩니다.

과거 이벤트의 이 세부 그림은 강력한 교육 도구입니다. 팀은 다른 사람이 이벤트를 검토하고 이로부터 학습할 수 있도록 전체 비즈니스에서 사용할 수 있는 중앙 리포지토리에 이러한 보고서를 저장해야 합니다. 이렇게 하면 프로덕션에서 발생할 수 있는 문제에 대한 팀의 직관이 향상될 수 있습니다.

상세한 인시던트 보고서의 리포지토리도 운영자를 위한 교육 자료의 출처가 될 수 있습니다. 팀은 인시던트 보고서를 사용하여 테이블탑 또는 라이브 게임 데이에 영감을 줄 수 있습니다. 여기서 팀에는 보고서에 캡처된 타임라인을 재생하는 정보가 제공됩니다. 운영자는 타임라인의 부분 정보를 사용하여 시나리오를 살펴보고 어떤 조치를 취할지 설명할 수 있습니다. 그러면 게임 데이의 진행자는 운영자의 작업에 따라 애플리케이션의 응답 방식에 대한 지침을 제공할 수 있습니다. 그러면 운영자의 문제 해결 기술을 개발하여 문제를 보다 쉽게 예측하고 해결할 수 있습니다.

애플리케이션 신뢰성을 담당하는 중앙 집중식 팀은 전체 조직이 액세스할 수 있는 중앙 집중식 라이브러리에서 이러한 보고서를 유지 관리해야 합니다. 또한 이 팀은 보고서 템플릿의 유지 관리와 인시던트 분석 보고서를 작성하는 방법에 대한 팀 교육도 책임져야 합니다. 신뢰성 팀은 정기적으로 보고서를 검토하여 소프트웨어 라이브러리, 아키텍처 패턴 또는 팀 프로세스 변경을 통해 해결할 수 있는 비즈니스 전반의 추세를 감지해야 합니다.

## 운영 검토 수행

**4단계: 운영**에서 설명한 대로 운영 검토는 최신 기능 릴리스, 인시던트 및 운영 지표를 검토할 수 있는 기회입니다. 또한 운영 검토는 기능 릴리스 및 인시던트로부터 학습한 내용을 조직의 더 광범위한 엔지니어링 커뮤니티와 공유할 수 있는 기회이기도 합니다. 운영 검토 중에 팀은 롤백된 기능 배포, 발생한 인시던트 및 해당 인시던트 처리 방법을 검토합니다. 이를 통해 조직 전체에서 엔지니어가 다른 사람의 경험을 통해 배우고 질문할 수 있습니다.

운영 검토를 회사의 엔지니어링 커뮤니티로 공개하여 비즈니스를 운영하는 IT 애플리케이션과 발생할 수 있는 문제 유형에 대해 자세히 알아볼 수 있습니다. 비즈니스에 대한 다른 애플리케이션을 설계, 구현 및 배포할 때 이러한 지식을 함께 전달합니다.

## 경보 성능 검토

운영 단계에서 설명한 대로 경보로 인해 대시보드 알림, 티켓 생성, 이메일 전송 또는 운영자 페이지가 발생할 수 있습니다. 애플리케이션에는 작업의 다양한 측면을 모니터링하도록 구성된 많은 경보가 있습니다. 시간이 지남에 따라 이러한 경보의 정확성과 효과를 검토하여 경보 정밀도를 높이고 오탐을 줄이며 중복 알림을 통합해야 합니다.

## 경보 정밀도

경보는 경보를 유발한 특정 종단을 해석하거나 진단하는 데 소요되는 시간을 줄이기 위해 최대한 구체적이어야 합니다. 애플리케이션 장애에 대응하여 경보가 발생하면 경보를 수신하고 이에 대응하는 운영자는 먼저 경보가 전달하는 정보를 해석해야 합니다. 이 정보는 복구 절차와 같은 일련의 작업에 매핑되는 간단한 오류 코드이거나 여기에 경보가 발생한 이유를 이해하기 위해 검토해야 하는 애플리케이션 로그의 줄이 포함될 수 있습니다. 팀이 애플리케이션을 보다 효과적으로 운영하는 방법을 학습하면 이러한 경보를 최대한 명확하고 간결하게 세부 조정해야 합니다.

애플리케이션에 발생할 수 있는 모든 종단을 예상할 수는 없으므로 운영자가 분석하고 진단해야 하는 일반 경보가 항상 있습니다. 팀은 응답 시간을 개선하고 평균 복구 시간(MTTR)을 줄이기 위해 일반 경보 수를 줄여야 합니다. 이상적으로는 경보와 자동화된 응답 또는 사람이 수행하는 응답 간에 일대일 관계가 있어야 합니다.

## 거짓 긍정

운영자의 조치가 필요하지 않지만 이메일, 페이지 또는 티켓으로 알림을 생성하는 경보는 시간이 지남에 따라 운영자가 무시합니다. 주기적으로 또는 인시던트 분석의 일부로 경보를 검토하여 종종 무시되거나 운영자의 조치가 필요하지 않은 경보(거짓 긍정)를 식별합니다. 경보를 제거하거나 운영자에게 실행 가능한 알림을 실행하도록 경보를 개선해야 합니다.

## 거짓 부정

인시던트 중에는 예기치 않은 방식으로 애플리케이션에 영향을 미치는 이벤트로 인해 인시던트 중에 경고하도록 구성된 경보가 실패할 수 있습니다. 인시던트 분석의 일환으로 발생했어야 하지만 발생하지 않은 경보를 검토해야 합니다. 이러한 경보를 개선하여 이벤트에서 경보가 발생할 수 있는 조건을 더 잘 반영하도록 해야 합니다. 또는 동일한 종단에 매핑되지만 종단의 다른 증상으로 인해 발생하는 추가 경보를 생성해야 할 수 있습니다.

## 중복 알림

애플리케이션에 장애를 일으키는 종단은 여러 증상을 유발할 수 있으며 여러 경보를 초래할 수 있습니다. 정기적으로 또는 인시던트 분석의 일부로 실행된 경보와 알림을 검토해야 합니다. 운영자가 중복 알림을 받은 경우 집계 경보를 생성하여 단일 알림 메시지로 통합합니다.

## 지표 검토 수행

팀은 매월 심각도별 인시던트 수, 인시던트 감지 시간, 원인 식별 시간, 문제 해결 시간, 생성된 티켓 수, 전송된 알림, 발생한 페이지 등 애플리케이션에 대한 운영 지표를 수집해야 합니다. 최소 한 달에 한 번

이러한 지표를 검토하여 운영 직원의 부담, 운영 직원이 처리하는 신호 대 잡음 비율(예: 정보 제공 및 실행 가능한 알림), 팀이 제어하는 애플리케이션을 운영하는 능력을 개선하고 있는지를 파악합니다. 이 검토를 사용하여 운영 팀의 측정 가능한 측면의 추세를 파악합니다. 이러한 지표를 개선하는 방법에 대해 팀에 아이디어를 구합니다.

## 교육 및 지원 제공

인시던트 또는 예상치 못한 동작으로 이어지는 애플리케이션 및 해당 환경에 대한 자세한 설명을 캡처하기 어렵습니다. 또한 이러한 시나리오를 예측하기 위해 애플리케이션의 복원력을 모델링하는 작업이 항상 간단한 것은 아닙니다. 조직은 운영 팀과 개발자가 복원력 모델링, 인시던트 분석, 게임 데이, 카오스 엔지니어링 실험과 같은 활동에 참여할 수 있도록 교육 및 지원 자료에 투자해야 합니다. 그러면 팀이 생성하는 보고서의 충실도와 팀이 캡처하는 지식이 향상됩니다. 또한 팀은 예정된 검토를 통해 통찰력을 얻어야 하는 더 작고 경험이 풍부한 엔지니어 그룹에 의존하지 않고도 장애를 예상할 수 있는 더 나은 역량을 갖추게 됩니다.

## 인시던트 지식 기반 생성

인시던트 보고서는 인시던트 분석의 표준 출력입니다. 애플리케이션에 장애가 발생하지 않았더라도 이상 애플리케이션 동작을 감지한 시나리오를 문서화하려면 동일하거나 유사한 보고서를 사용해야 합니다. 동일한 표준화된 보고서 구조를 사용하여 카오스 실험 및 게임 데이의 결과를 캡처합니다. 보고서는 인시던트 또는 예상치 못한 동작으로 이어지는 애플리케이션 및 환경의 스냅샷을 나타냅니다. 이러한 표준화된 보고서는 비즈니스 내 모든 엔지니어가 액세스할 수 있는 중앙 리포지토리에 저장해야 합니다.

그런 다음 운영 팀과 개발자는 이 지식 기반을 검색하여 과거에 애플리케이션을 중단시킨 요소, 중단을 일으킬 수 있는 시나리오 유형, 애플리케이션 장애를 방해한 요소를 파악할 수 있습니다. 이 지식 기반은 운영 팀과 개발자의 기술을 개선하기 위한 액셀러레이터가 되고, 이를 통해 지식과 경험을 공유할 수 있습니다. 또한 보고서를 게임 데이 또는 카오스 실험을 위한 훈련 자료 또는 시나리오로 사용하여 운영 팀의 직관과 중단 문제를 해결하는 능력을 개선할 수 있습니다.

### Note

또한 표준화된 보고서 형식은 독자에게 친숙함을 제공하고 독자가 찾고 있는 정보를 더 빠르게 찾는 데 도움이 됩니다.

## 심층 복원력 구현

앞서 설명한 대로 진보한 조직은 경보에 대해 여러 응답을 구현합니다. 응답이 효과적일 것이라는 보장은 없으므로 응답을 계층화하면 애플리케이션이 정상적으로 실패할 수 있습니다. 개별 응답이 DR 시나리오로 이어질 수 있는 단일 장애점이 되지 않도록 각 지표에 대해 두 개 이상의 응답을 구현하는 것이 좋습니다. 이러한 계층은 이전 응답이 효과적이지 않은 경우에만 연속 응답이 수행되도록 순차적으로 생성해야 합니다. 단일 경보에 대해 계층화된 여러 응답을 실행해서는 안 됩니다. 대신 응답이 실패했는지 여부를 나타내는 경보를 사용하고 실패한 경우 다음 계층화된 응답을 시작합니다.

## 결론 및 리소스

이 가이드에서는 목표 설정, 설계 및 구현, 평가 및 테스트, 운영, 대응 및 학습과 같은 5단계로 모범 사례를 구현하여 애플리케이션의 복원력을 지속적으로 개선하는 데 도움이 되는 수명 주기를 제공합니다.

이 가이드에서 설명하는 서비스 및 개념에 대한 자세한 내용은 다음 리소스를 참조하세요.

AWS 서비스:

- [AWS Backup](#)
- [AWS Elastic Disaster Recovery](#)
- [AWS Fault Injection Service \(AWS FIS\)](#)
- [AWS Resilience Hub](#)
- [Amazon Application Recovery Controller\(ARC\)](#)
- [AWS X-Ray](#)

블로그 게시물 및 기사:

- [Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS](#)
- [AWS Fault Isolation Boundaries](#)
- [AWS Multi-Region Fundamentals](#)
- [Chaos Engineering in the cloud](#)
- [Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline](#)
- [Disaster Recovery of On-Premises Applications to AWS](#)
- [신뢰성 원칙 - AWS Well-Architected Framework](#)
- [Resilience analysis framework](#)

## 기여자

이 가이드의 기여자는 다음과 같습니다.

- Bruno Emer, Principal Solutions Architect, AWS
- Clark Richey, Principal Solutions Architect, AWS
- Elaine Harvey, General Manager, Reliability Services, AWS
- Jason Barto, Principal Solutions Architect, AWS
- John Formento, Principal Solutions Architect, AWS
- Lisi Lewis, Sr. Product Marketing Manager, AWS
- Michael Haken, Principal Solutions Architect, AWS
- Neeraj Kumar, Principal Solutions Architect, AWS
- Wangechi Doble, Principal Solutions Architect, AWS

## 문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 [RSS 피드](#)를 구독하십시오.

변경 사항	설명	날짜
<a href="#">최초 게시</a>	—	2023년 10월 6일

# AWS 권장 가이드 용어집

다음은 AWS 권장 가이드에서 제공하는 전략, 가이드 및 패턴에서 일반적으로 사용되는 용어입니다. 용어집 항목을 제안하려면 용어집 끝에 있는 피드백 제공 링크를 사용하십시오.

## 숫자

### 7가지 전략

애플리케이션을 클라우드로 이전하기 위한 7가지 일반적인 마이그레이션 전략 이러한 전략은 Gartner가 2011년에 파악한 5가지 전략을 기반으로 하며 다음으로 구성됩니다.

- 리팩터링/리아키텍트 - 클라우드 네이티브 기능을 최대한 활용하여 애플리케이션을 이동하고 해당 아키텍처를 수정함으로써 민첩성, 성능 및 확장성을 개선합니다. 여기에는 일반적으로 운영 체제와 데이터베이스 이식이 포함됩니다. 예: 온프레미스 Oracle 데이터베이스를 Amazon Aurora PostgreSQL 호환 에디션으로 마이그레이션합니다.
- 리플랫폼(리프트 앤드 리세이프) - 애플리케이션을 클라우드로 이동하고 일정 수준의 최적화를 도입하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드의 Amazon Relational Database Service(Amazon RDS) for Oracle로 마이그레이션합니다.
- 재구매(드롭 앤드 쇼프) - 일반적으로 기존 라이선스에서 SaaS 모델로 전환하여 다른 제품으로 전환합니다. 예: 고객 관계 관리(CRM) 시스템을 Salesforce.com으로 마이그레이션합니다.
- 리호스팅(리프트 앤드 시프트) - 애플리케이션을 변경하지 않고 클라우드로 이동하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드클라우드의 EC2 인스턴스에 있는 Oracle로 마이그레이션합니다.
- 재배포(하이퍼바이저 수준의 리프트 앤 시프트) - 새 하드웨어를 구매하거나, 애플리케이션을 다시 작성하거나, 기존 운영을 수정하지 않고도 인프라를 클라우드로 이동합니다. 온프레미스 플랫폼에서 동일한 플랫폼의 클라우드 서비스로 서버를 마이그레이션합니다. 예: Microsoft Hyper-V 애플리케이션을 로 마이그레이션합니다 AWS.
- 유지(보관) - 소스 환경에 애플리케이션을 유지합니다. 대규모 리팩터링이 필요하고 해당 작업을 나중에 연기하려는 애플리케이션과 비즈니스 차원에서 마이그레이션할 이유가 없어 유지하려는 레거시 애플리케이션이 여기에 포함될 수 있습니다.
- 사용 중지 - 소스 환경에서 더 이상 필요하지 않은 애플리케이션을 폐기하거나 제거합니다.

# A

## A2A(Agent-to-Agent)

작업 위임 및 상태 전송 agent-to-agent 공동 작업을 위한 상태 저장 프로토콜입니다.

## ABAC

[속성 기반 액세스 제어](#)를 참조하세요.

## 추상화된 서비스

[관리형 서비스](#)를 참조하세요.

## ACID

[원자성, 일관성, 격리성, 내구성](#)을 참조하세요.

## 능동-능동 마이그레이션

양방향 복제 도구 또는 이중 쓰기 작업을 사용하여 소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되고, 두 데이터베이스 모두 마이그레이션 중 연결 애플리케이션의 트랜잭션을 처리하는 데이터베이스 마이그레이션 방법입니다. 이 방법은 일회성 전환이 필요한 대신 소규모의 제어된 배치로 마이그레이션을 지원합니다. 더 유연하지만 [액티브 패시브 마이그레이션](#)보다 더 많은 작업이 필요합니다.

## 능동-수동 마이그레이션

소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되지만 소스 데이터베이스만 연결 애플리케이션의 트랜잭션을 처리하고 데이터는 대상 데이터베이스로 복제되는 데이터베이스 마이그레이션 방법입니다. 대상 데이터베이스는 마이그레이션 중 어떤 트랜잭션도 허용하지 않습니다.

## 에이전트

목표를 달성하기 위한 도구를 사용하여 자율적으로 추론, 계획 및 조치를 취할 수 있는 AI 시스템입니다.

## 에이전트 운영

대규모 프로덕션 환경에서 AI 에이전트를 구축, 테스트, 배포 및 실행하기 위한 운영 사례입니다.

## 집계 함수

행 그룹에서 작동하고 그룹에 대한 단일 반환 값을 계산하는 SQL 함수입니다. 집계 함수의 예로 SUM 및 MAX가 있습니다.

## AI

[인공 지능](#)을 참조하세요.

### AIOps

[인공 지능 운영](#)을 참조하세요.

### 익명화

데이터세트에서 개인 정보를 영구적으로 삭제하는 프로세스입니다. 익명화는 개인 정보 보호에 도움이 될 수 있습니다. 익명화된 데이터는 더 이상 개인 데이터로 간주되지 않습니다.

### 안티 패턴

솔루션이 다른 솔루션보다 비생산적이거나 비효율적이거나 덜 효과적이어서 반복되는 문제에 자주 사용되는 솔루션입니다.

### 애플리케이션 제어

맬웨어로부터 시스템을 보호하기 위해 승인된 애플리케이션만 사용하도록 허용하는 보안 접근 방식입니다.

### 애플리케이션 포트폴리오

애플리케이션 구축 및 유지 관리 비용과 애플리케이션의 비즈니스 가치를 비롯하여 조직에서 사용하는 각 애플리케이션에 대한 세부 정보 모음입니다. 이 정보는 [포트폴리오 탐색 및 분석 프로세스](#)의 핵심이며 마이그레이션, 현대화 및 최적화할 애플리케이션을 식별하고 우선순위를 정하는 데 도움이 됩니다.

### 인공 지능

컴퓨터 기술을 사용하여 학습, 문제 해결, 패턴 인식 등 일반적으로 인간과 관련된 인지 기능을 수행하는 것을 전문으로 하는 컴퓨터 과학 분야입니다. 자세한 내용은 [What is Artificial Intelligence?](#)를 참조하십시오.

### 인공 지능 운영(AIOps)

기계 학습 기법을 사용하여 운영 문제를 해결하고, 운영 인시던트 및 사용자 개입을 줄이고, 서비스 품질을 높이는 프로세스입니다. AWS 마이그레이션 전략에서 AIOps가 사용되는 방법에 대한 자세한 내용은 [운영 통합 가이드](#)를 참조하십시오.

### 비대칭 암호화

한 쌍의 키, 즉 암호화를 위한 퍼블릭 키와 복호화를 위한 프라이빗 키를 사용하는 암호화 알고리즘입니다. 퍼블릭 키는 복호화에 사용되지 않으므로 공유할 수 있지만 프라이빗 키에 대한 액세스는 엄격히 제한되어야 합니다.

## 원자성, 일관성, 격리성, 내구성(ACID)

오류, 정전 또는 기타 문제가 발생한 경우에도 데이터베이스의 데이터 유효성과 운영 신뢰성을 보장하는 소프트웨어 속성 세트입니다.

## ABAC(속성 기반 액세스 제어)

부서, 직무, 팀 이름 등의 사용자 속성을 기반으로 세분화된 권한을 생성하는 방식입니다. 자세한 내용은 AWS Identity and Access Management (IAM) 설명서의 [용 ABAC AWS](#)를 참조하세요.

## 신뢰할 수 있는 데이터 소스

가장 신뢰할 수 있는 정보 소스로 간주되는 기본 버전의 데이터를 저장하는 위치입니다. 익명화, 편집 또는 가명화와 같은 데이터 처리 또는 수정의 목적으로 신뢰할 수 있는 데이터 소스의 데이터를 다른 위치로 복사할 수 있습니다.

## 가용 영역

다른 가용 영역의 장애로부터 격리 AWS 리전 되고 동일한 리전의 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하는 내의 고유한 위치입니다.

## AWS 클라우드 채택 프레임워크(AWS CAF)

조직이 클라우드로 성공적으로 전환하기 위한 효율적이고 효과적인 계획을 개발하는 AWS 데 도움이 되는 지침 및 모범 사례 프레임워크입니다. AWS CAF는 지침을 비즈니스, 사람, 거버넌스, 플랫폼, 보안 및 운영이라는 6가지 중점 영역으로 구성합니다. 비즈니스, 사람 및 거버넌스 관점은 비즈니스 기술과 프로세스에 초점을 맞추고, 플랫폼, 보안 및 운영 관점은 전문 기술과 프로세스에 중점을 둡니다. 예를 들어, 사람 관점은 인사(HR), 직원 배치 기능 및 인력 관리를 담당하는 이해관계자를 대상으로 합니다. 이러한 관점에서 AWS CAF는 성공적인 클라우드 채택을 위해 조직을 준비하는 데 도움이 되는 인력 개발, 교육 및 커뮤니케이션에 대한 지침을 제공합니다. 자세한 내용은 [AWS CAF 웹사이트](#)와 [AWS CAF 백서](#)를 참조하세요.

## AWS 워크로드 검증 프레임워크(AWS WQF)

데이터베이스 마이그레이션 워크로드를 평가하고, 마이그레이션 전략을 권장하고, 작업 견적을 제공하는 도구입니다. AWS WQF는 AWS Schema Conversion Tool (AWS SCT)에 포함되어 있습니다. 데이터베이스 스키마 및 코드 객체, 애플리케이션 코드, 종속성 및 성능 특성을 분석하고 평가 보고서를 제공합니다.

## B

### 악성 봇

개인 또는 조직을 방해하거나 해를 입히기 위한 [봇](#)입니다.

### BCP

[비즈니스 연속성 계획](#)을 참조하세요.

### 동작 그래프

리소스 동작과 시간 경과에 따른 상호 작용에 대한 통합된 대화형 뷰입니다. Amazon Detective에서 동작 그래프를 사용하여 실패한 로그인 시도, 의심스러운 API 직접 호출 및 유사한 작업을 검사할 수 있습니다. 자세한 내용은 Detective 설명서의 [Data in a behavior graph](#)를 참조하십시오.

### 빅 엔디안 시스템

가장 중요한 바이트를 먼저 저장하는 시스템입니다. [엔디안](#)도 참조하세요.

### 바이너리 분류

바이너리 결과(가능한 두 클래스 중 하나)를 예측하는 프로세스입니다. 예를 들어, ML 모델이 “이 이메일이 스팸인가요, 스팸이 아닌가요?”, ‘이 제품은 책임가요, 자동차인가요?’ 등의 문제를 예측해야 할 수 있습니다.

### 블룸 필터

요소가 세트의 멤버인지 여부를 테스트하는 데 사용되는 메모리 효율성이 높은 확률론적 데이터 구조입니다.

### 블루/그린(Blue/Green) 배포

동일하지만 별개의 두 환경을 생성하는 배포 전략입니다. 하나의 환경(파란색)에서 현재 애플리케이션 버전을 실행하고 새 애플리케이션 버전은 다른 환경(녹색)에서 실행합니다. 이 전략을 사용하면 영향을 최소화하면서 신속하게 롤백할 수 있습니다.

### bot

인터넷을 통해 자동화된 태스크를 실행하고 인적 활동이나 상호 작용을 시뮬레이션하는 소프트웨어 애플리케이션입니다. 인터넷에서 정보를 인덱싱하는 웹 크롤러와 같이 유용하거나 이로운 봇도 있습니다. 악성 봇이라고 하는 다른 일부 봇은 개인 또는 조직을 방해하거나 해를 입히기 위한 봇입니다.

## 봇넷

[맬웨어](#)에 감염되고 봇 허더 또는 봇 운영자와 같은 단일 당사자가 제어하는 [봇](#) 네트워크입니다. 봇넷은 봇의 규모와 봇의 영향 범위를 확대하는 가장 잘 알려진 메커니즘입니다.

## 브랜치

코드 리포지토리의 포함된 영역입니다. 리포지토리에 생성되는 첫 번째 브랜치가 기본 브랜치입니다. 기존 브랜치에서 새 브랜치를 생성한 다음 새 브랜치에서 기능을 개발하거나 버그를 수정할 수 있습니다. 기능을 구축하기 위해 생성하는 브랜치를 일반적으로 기능 브랜치라고 합니다. 기능을 출시할 준비가 되면 기능 브랜치를 기본 브랜치에 다시 병합합니다. 자세한 내용은 [About branches](#)(GitHub 설명서)를 참조하십시오.

## 긴급 액세스 권한

예외적인 상황에서 승인된 프로세스를 통해 사용자가 일반적으로 액세스할 권한이 없는데 액세스할 수 있는 빠른 방법입니다. 자세한 내용은 AWS Well-Architected 지침의 [Implement break-glass procedures](#) 지표를 참조하세요.

## 브라운필드 전략

사용자 환경의 기존 인프라 시스템 아키텍처에 브라운필드 전략을 채택할 때는 현재 시스템 및 인프라의 제약 조건을 중심으로 아키텍처를 설계합니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 [그린필드](#) 전략을 혼합할 수 있습니다.

## 버퍼 캐시

가장 자주 액세스하는 데이터가 저장되는 메모리 영역입니다.

## 사업 역량

기업이 가치를 창출하기 위해 하는 일(예: 영업, 고객 서비스 또는 마케팅)입니다. 마이크로서비스 아키텍처 및 개발 결정은 비즈니스 역량에 따라 이루어질 수 있습니다. 자세한 내용은 백서의 [AWS에서 컨테이너화된 마이크로서비스 실행의 비즈니스 역량 중심의 구성화](#) 섹션을 참조하십시오.

## 비즈니스 연속성 계획(BCP)

대규모 마이그레이션과 같은 중단 이벤트가 운영에 미치는 잠재적 영향을 해결하고 비즈니스가 신속하게 운영을 재개할 수 있도록 지원하는 계획입니다.

# C

## CAF

[AWS Cloud Adoption Framework](#)를 참조하세요.

## 카나리 배포

최종 사용자에게 제공하는 느린 증분 릴리스 버전입니다. 확신이 들면 새 버전을 배포하고 현재 버전을 완전히 교체합니다.

## CCoE

[클라우드 혁신 센터](#)를 참조하세요.

## CDC

[데이터 캡처 변경](#)을 참조하세요.

## 변경 데이터 캡처(CDC)

데이터베이스 테이블과 같은 데이터 소스의 변경 내용을 추적하고 변경 사항에 대한 메타데이터를 기록하는 프로세스입니다. 대상 시스템의 변경 내용을 감사하거나 복제하여 동기화를 유지하는 등의 다양한 용도로 CDC를 사용할 수 있습니다.

## 카오스 엔지니어링

시스템의 복원력을 테스트하기 위해 의도적으로 장애나 중단 이벤트를 도입합니다. [AWS Fault Injection Service \(AWS FIS\)](#)를 사용하여 AWS 워크로드에 스트레스를 주고 응답을 평가하는 실험을 수행할 수 있습니다.

## CI/CD

[지속적 통합 및 지속적 전송](#)을 참조하세요.

## 분류

예측을 생성하는 데 도움이 되는 분류 프로세스입니다. 분류 문제에 대한 ML 모델은 이산 값을 예측합니다. 이산 값은 항상 서로 다릅니다. 예를 들어, 모델이 이미지에 자동차가 있는지 여부를 평가해야 할 수 있습니다.

## 시민 개발자

전문 기술 없이 노코드/로우코드 플랫폼을 사용하여 AI 애플리케이션을 생성하는 비즈니스 사용자입니다.

## 클라이언트측 암호화

대상이 데이터를 AWS 서비스 수신하기 전에 로컬에서 데이터를 암호화합니다.

## 클라우드 혁신 센터(CCoE)

클라우드 모범 사례 개발, 리소스 동원, 마이그레이션 타임라인 설정, 대규모 혁신을 통한 조직 선도 등 조직 전체에서 클라우드 채택 노력을 추진하는 다분야 팀입니다. 자세한 내용은 AWS 클라우드 엔터프라이즈 전략 블로그의 [CCoE 게시물](#)을 참조하세요.

## 클라우드 컴퓨팅

원격 데이터 스토리지와 IoT 디바이스 관리에 일반적으로 사용되는 클라우드 기술 클라우드 컴퓨팅은 일반적으로 [엣지 컴퓨팅](#) 기술에 연결되어 있습니다.

## 클라우드 운영 모델

IT 조직에서 하나 이상의 클라우드 환경을 구축, 성숙화 및 최적화하는 데 사용되는 운영 모델입니다. 자세한 내용은 [클라우드 운영 모델 구축](#)을 참조하십시오.

## 클라우드 채택 단계

조직이 AWS 클라우드로 마이그레이션할 때 일반적으로 거치는 4단계는 다음과 같습니다.

- 프로젝트 - 개념 증명 및 학습 목적으로 몇 가지 클라우드 관련 프로젝트 실행
- 기반 - 클라우드 채택 확장을 위한 기초 투자(예: 랜딩 존 생성, CCoE 정의, 운영 모델 구축)
- 마이그레이션 - 개별 애플리케이션 마이그레이션
- Re-invention - 제품 및 서비스 최적화와 클라우드 혁신

이러한 단계는 Stephen Orban이 블로그 게시물 [The Journey Toward Cloud-First and the Stages of Adoption](#) on the AWS 클라우드 Enterprise Strategy 블로그에서 정의했습니다. AWS 마이그레이션 전략과 어떤 관련이 있는지에 대한 자세한 내용은 [마이그레이션 준비 가이드](#)를 참조하세요.

## CMDB

[구성 관리 데이터베이스](#)를 참조하세요.

## 코드 리포지토리

소스 코드와 설명서, 샘플, 스크립트 등의 기타 자산이 버전 관리 프로세스를 통해 저장되고 업데이트되는 위치입니다. 일반적인 클라우드 리포지토리로 GitHub 또는 Bitbucket Cloud가 포함됩니다. 코드의 각 버전을 브랜치라고 합니다. 마이크로서비스 구조에서 각 리포지토리는 단일 기능 전용입니다. 단일 CI/CD 파이프라인은 여러 리포지토리를 사용할 수 있습니다.

## 콜드 캐시

비어 있거나, 제대로 채워지지 않았거나, 오래되었거나 관련 없는 데이터를 포함하는 버퍼 캐시입니다. 주 메모리나 디스크에서 데이터베이스 인스턴스를 읽어야 하기 때문에 성능에 영향을 미치며, 이는 버퍼 캐시에서 읽는 것보다 느립니다.

## 콜드 데이터

거의 액세스되지 않고 일반적으로 과거 데이터인 데이터. 이런 종류의 데이터를 쿼리할 때는 일반적으로 느린 쿼리가 허용됩니다. 이 데이터를 성능이 낮고 비용이 저렴한 스토리지 계층 또는 클래스로 옮기면 비용을 절감할 수 있습니다.

## 컴퓨터 비전(CV)

기계 학습을 사용하여 디지털 이미지 및 비디오와 같은 시각적 형식에서 정보를 분석하고 추출하는 [AI](#) 필드입니다. 예를 들어 Amazon SageMaker AI는 CV에 대한 이미지 처리 알고리즘을 제공합니다.

## 구성 드리프트

워크로드의 경우 구성이 예상되는 상태에서 변경됩니다. 이로 인해 워크로드가 규정을 준수하지 않을 수 있으며, 이는 일반적으로 점진적이고 의도되지 않은 작업입니다.

## 구성 관리 데이터베이스(CMDB)

하드웨어 및 소프트웨어 구성 요소와 해당 구성을 포함하여 데이터베이스와 해당 IT 환경에 대한 정보를 저장하고 관리하는 리포지토리입니다. 일반적으로 마이그레이션의 포트폴리오 탐색 및 분석 단계에서 CMDB의 데이터를 사용합니다.

## 규정 준수 팩

규정 준수 및 보안 검사를 사용자 지정하기 위해 조합할 수 있는 AWS Config 규칙 및 문제 해결 작업의 모음입니다. YAML 템플릿을 사용하여 적합성 팩을 AWS 계정 및 리전 또는 조직 전체에 단일 엔터티로 배포할 수 있습니다. 자세한 내용은 AWS Config 설명서의 [적합성 팩](#)을 참조하세요.

## 지속적 통합 및 지속적 전달(CI/CD)

소프트웨어 릴리스 프로세스의 소스, 빌드, 테스트, 스테이징 및 프로덕션 단계를 자동화하는 프로세스입니다. CI/CD는 일반적으로 파이프라인으로 설명됩니다. CI/CD를 통해 프로세스를 자동화하고, 생산성을 높이고, 코드 품질을 개선하고, 더 빠르게 제공할 수 있습니다. 자세한 내용은 [지속적 전달의 이점](#)을 참조하십시오. CD는 지속적 배포를 의미하기도 합니다. 자세한 내용은 [지속적 전달 \(Continuous Delivery\)과 지속적인 개발](#)을 참조하십시오.

## CV

[컴퓨터 비전](#)을 참조하세요.

## D

### 저장 데이터

스토리지에 있는 데이터와 같이 네트워크에 고정되어 있는 데이터입니다.

### 데이터 분류

중요도와 민감도를 기준으로 네트워크의 데이터를 식별하고 분류하는 프로세스입니다. 이 프로세스는 데이터에 대한 적절한 보호 및 보존 제어를 결정하는 데 도움이 되므로 사이버 보안 위험 관리 전략의 중요한 구성 요소입니다. 데이터 분류는 AWS Well-Architected Framework의 보안 원칙 구성 요소입니다. 자세한 내용은 [데이터 분류](#)를 참조하십시오.

### 데이터 드리프트

프로덕션 데이터와 ML 모델 학습에 사용된 데이터 간의 상당한 차이 또는 시간 경과에 따른 입력 데이터의 의미 있는 변화. 데이터 드리프트는 ML 모델 예측의 전반적인 품질, 정확성 및 공정성을 저하시킬 수 있습니다.

### 전송 중 데이터

네트워크를 통과하고 있는 데이터입니다. 네트워크 리소스 사이를 이동 중인 데이터를 예로 들 수 있습니다.

### 데이터 메시

중앙 집중식 관리 및 거버넌스를 통해 분산되고 탈중앙화된 데이터 소유권을 제공하는 아키텍처 프레임워크입니다.

### 데이터 최소화

꼭 필요한 데이터만 수집하고 처리하는 원칙입니다. 에서 데이터를 최소화하면 개인 정보 보호 위험, 비용 및 분석 탄소 발자국을 줄일 AWS 클라우드 수 있습니다.

### 데이터 경계

신뢰할 수 있는 자격 증명만 예상 네트워크에서 신뢰할 수 있는 리소스에 액세스하도록 하는 데 도움이 되는 AWS 환경의 예방 가드레일 세트입니다. 자세한 내용은 [데이터 경계 구축을 참조하세요 AWS](#).

### 데이터 사전 처리

원시 데이터를 ML 모델이 쉽게 구문 분석할 수 있는 형식으로 변환하는 것입니다. 데이터를 사전 처리한다는 것은 특정 열이나 행을 제거하고 누락된 값, 일관성이 없는 값 또는 중복 값을 처리함을 의미할 수 있습니다.

## 데이터 출처

라이프사이클 전반에 걸쳐 데이터의 출처와 기록을 추적하는 프로세스(예: 데이터 생성, 전송, 저장 방법).

## 데이터 주체

데이터를 수집 및 처리하는 개인입니다.

## 데이터 웨어하우스

분석과 같은 비즈니스 인텔리전스를 지원하는 데이터 관리 시스템입니다. 데이터 웨어하우스에는 보통 많은 양의 기록 데이터가 포함되며 일반적으로 쿼리 및 분석에 사용됩니다.

## 데이터 정의 언어(DDL)

데이터베이스에서 테이블 및 객체의 구조를 만들거나 수정하기 위한 명령문 또는 명령입니다.

## 데이터베이스 조작 언어(DML)

데이터베이스에서 정보를 수정(삽입, 업데이트 및 삭제)하기 위한 명령문 또는 명령입니다.

## DDL

[데이터 정의 언어](#)를 참조하세요.

## 딥 앙상블

예측을 위해 여러 딥 러닝 모델을 결합하는 것입니다. 딥 앙상블을 사용하여 더 정확한 예측을 얻거나 예측의 불확실성을 추정할 수 있습니다.

## 딥 러닝

여러 계층의 인공 신경망을 사용하여 입력 데이터와 관심 대상 변수 간의 매핑을 식별하는 ML 하위 분야입니다.

## 심층 방어

네트워크와 그 안의 데이터 기밀성, 무결성 및 가용성을 보호하기 위해 컴퓨터 네트워크 전체에 일련의 보안 메커니즘과 제어를 신중하게 계층화하는 정보 보안 접근 방식입니다. 이 전략을 채택하면 AWS Organizations 구조의 여러 계층에 여러 컨트롤을 AWS 추가하여 리소스를 보호할 수 있습니다. 예를 들어, 심층 방어 접근 방식은 다단계 인증, 네트워크 세분화 및 암호화를 결합할 수 있습니다.

## 위임된 관리자

에서 AWS Organizations 호환되는 서비스는 AWS 멤버 계정을 등록하여 조직의 계정을 관리하고 해당 서비스에 대한 권한을 관리할 수 있습니다. 이러한 계정을 해당 서비스의 위임된 관리자라고

합니다. 자세한 내용과 호환되는 서비스 목록은 AWS Organizations 설명서의 [AWS Organizations](#)와 함께 사용할 수 있는 AWS 서비스를 참조하십시오.

## 배포

대상 환경에서 애플리케이션, 새 기능 또는 코드 수정 사항을 사용할 수 있도록 하는 프로세스입니다. 배포에는 코드 베이스의 변경 사항을 구현한 다음 애플리케이션 환경에서 해당 코드베이스를 구축하고 실행하는 작업이 포함됩니다.

## 개발 환경

[환경](#)을 참조하세요.

## 탐지 제어

이벤트 발생 후 탐지, 기록 및 알림을 수행하도록 설계된 보안 제어입니다. 이러한 제어는 기존의 예방적 제어를 우회한 보안 이벤트를 알리는 2차 방어선입니다. 자세한 내용은 AWS에서 보안 제어 구현의 [탐지 제어](#)를 참조하세요.

## 개발 가치 흐름 매핑 (DVSM)

소프트웨어 개발 라이프사이클에서 속도와 품질에 부정적인 영향을 미치는 제약 조건을 식별하고 우선 순위를 지정하는 데 사용되는 프로세스입니다. DVSM은 원래 린 제조 방식을 위해 설계된 가치 흐름 매핑 프로세스를 확장합니다. 소프트웨어 개발 프로세스를 통해 가치를 창출하고 이동하는 데 필요한 단계와 팀에 중점을 둡니다.

## 디지털 트윈

건물, 공장, 산업 장비 또는 생산 라인과 같은 실제 시스템을 가상으로 표현한 것입니다. 디지털 트윈은 예측 유지 보수, 원격 모니터링, 생산 최적화를 지원합니다.

## 차원 테이블

[스타 스키마](#)에서 팩트 테이블의 정량적 데이터에 대한 데이터 속성을 포함하는 더 작은 테이블을 말합니다. 차원 테이블 속성은 일반적으로 텍스트 필드나 텍스트처럼 동작하는 개별 숫자입니다. 이러한 속성은 보통 쿼리 제약, 필터링 및 결과 세트 레이블 지정에 사용됩니다.

## 재해

워크로드 또는 시스템이 기본 배포 위치에서 비즈니스 목표를 달성하지 못하게 방해하는 이벤트입니다. 이러한 이벤트는 자연재해, 기술적 오류, 의도하지 않은 구성 오류 또는 멀웨어 공격과 같은 사람의 행동으로 인한 결과일 수 있습니다.

## 재해 복구(DR)

[재해](#)로 인한 가동 중지 시간 및 데이터 손실을 최소화하기 위해 사용하는 전략 및 프로세스입니다. 자세한 내용은 AWS Well-Architected Framework의 [Disaster Recovery of Workloads on AWS: Recovery in the Cloud](#)를 참조하세요.

## DML

[데이터베이스 조작 언어](#)를 참조하세요.

## 도메인 기반 설계

구성 요소를 각 구성 요소가 제공하는 진화하는 도메인 또는 핵심 비즈니스 목표에 연결하여 복잡한 소프트웨어 시스템을 개발하는 접근 방식입니다. 이 개념은 에릭 에반스에 의해 그의 저서인 도메인 기반 디자인: 소프트웨어 중심의 복잡성 해결(Boston: Addison-Wesley Professional, 2003)에서 소개되었습니다. Strangler Fig 패턴과 함께 도메인 기반 설계를 사용하는 방법에 대한 자세한 내용은 [컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET\(ASMX\) 웹 서비스를 점진적으로 현대화하는 방법](#)을 참조하십시오.

## DR

[재해 복구](#)를 참조하세요.

## 드리프트 감지

기준이 되는 구성과의 편차 추적을 말합니다. 예를 들어 AWS CloudFormation 를 사용하여 [시스템 리소스의 드리프트를 감지](#)하거나 사용하여 AWS Control Tower 거버넌스 요구 사항 준수에 영향을 미칠 수 있는 [랜딩 존의 변경 사항을 감지](#)할 수 있습니다.

## DVSM

[개발 가치 흐름 매핑](#)을 참조하세요.

## E

### EDA

[탐색 데이터 분석](#)을 참조하세요.

### EDI

[전자 데이터 교환](#)을 참조하세요.

## 엣지 컴퓨팅

IoT 네트워크의 엣지에서 스마트 디바이스의 컴퓨팅 성능을 개선하는 기술 엣지 컴퓨팅은 [클라우드 컴퓨팅](#)에 비해 보다 통신 지연 시간을 줄이고 응답 시간을 개선할 수 있습니다.

## 전자 데이터 교환(EDI)

조직 간 비즈니스 문서의 자동화된 교환을 나타냅니다. 자세한 내용은 [전자 데이터 교환\(EDI\)이란 무엇인가요?](#)를 참조하세요.

## 암호화

사람이 읽을 수 있는 일반 텍스트 데이터를 사이퍼텍스트로 변환하는 컴퓨팅 프로세스입니다.

## 암호화 키

암호화 알고리즘에 의해 생성되는 무작위 비트의 암호화 문자열입니다. 키의 길이는 다양할 수 있으며 각 키는 예측할 수 없고 고유하게 설계되었습니다.

## 엔디안

컴퓨터 메모리에 바이트가 저장되는 순서입니다. 빅 엔디안 시스템은 가장 중요한 바이트를 먼저 저장합니다. 리틀 엔디안 시스템은 가장 덜 중요한 바이트를 먼저 저장합니다.

## 엔드포인트

[서비스 엔드포인트](#)를 참조하세요.

## 엔드포인트 서비스

Virtual Private Cloud(VPC)에서 호스팅하여 다른 사용자와 공유할 수 있는 서비스입니다. 를 사용하여 엔드포인트 서비스를 생성하고 다른 AWS 계정 또는 AWS Identity and Access Management (IAM) 보안 주체에 권한을 AWS PrivateLink 부여할 수 있습니다. 이러한 계정 또는 보안 주체는 인터페이스 VPC 엔드포인트를 생성하여 엔드포인트 서비스에 비공개로 연결할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud(VPC) 설명서의 [엔드포인트 서비스 생성](#)을 참조하십시오.

## 엔터프라이즈 리소스 계획(ERP)

엔터프라이즈의 주요 비즈니스 프로세스(예: 회계, [MES](#), 프로젝트 관리)를 자동화하고 관리하는 시스템입니다.

## 봉투 암호화

암호화 키를 다른 암호화 키로 암호화하는 프로세스입니다. 자세한 내용은 AWS Key Management Service (AWS KMS) 설명서의 [봉투 암호화](#)를 참조하세요.

## 환경

실행 중인 애플리케이션의 인스턴스입니다. 다음은 클라우드 컴퓨팅의 일반적인 환경 유형입니다.

- 개발 환경 - 애플리케이션 유지 관리를 담당하는 핵심 팀만 사용할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. 개발 환경은 변경 사항을 상위 환경으로 승격하기 전에 테스트하는 데 사용됩니다. 이러한 유형의 환경을 테스트 환경이라고도 합니다.
- 하위 환경 - 초기 빌드 및 테스트에 사용되는 환경을 비롯한 애플리케이션의 모든 개발 환경입니다.
- 프로덕션 환경 - 최종 사용자가 액세스할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. CI/CD 파이프라인에서 프로덕션 환경이 마지막 배포 환경입니다.
- 상위 환경 - 핵심 개발 팀 이외의 사용자가 액세스할 수 있는 모든 환경입니다. 프로덕션 환경, 프로덕션 이전 환경 및 사용자 수용 테스트를 위한 환경이 여기에 포함될 수 있습니다.

## 에픽

애자일 방법론에서 작업을 구성하고 우선순위를 정하는 데 도움이 되는 기능적 범주입니다. 에픽은 요구 사항 및 구현 작업에 대한 개괄적인 설명을 제공합니다. 예를 들어, AWS CAF 보안 에픽에는 ID 및 액세스 관리, 탐지 제어, 인프라 보안, 데이터 보호 및 인시던트 대응이 포함됩니다. AWS 마 이그레이션 전략의 에픽에 대한 자세한 내용은 [프로그램 구현 가이드](#)를 참조하십시오.

## ERP

[엔터프라이즈 리소스 계획](#)을 참조하세요.

### 탐색 데이터 분석(EDA)

데이터 세트를 분석하여 주요 특성을 파악하는 프로세스입니다. 데이터를 수집 또는 집계한 다음 초기 조사를 수행하여 패턴을 찾고, 이상을 탐지하고, 가정을 확인합니다. EDA는 요약 통계를 계산하고 데이터 시각화를 생성하여 수행됩니다.

## F

### 팩트 테이블

[스타 스키마](#)의 중앙 테이블입니다. 비즈니스 운영에 대한 정량적 데이터를 저장합니다. 일반적으로 팩트 테이블은 측정값이 있는 열 및 차원 테이블에 대한 외래 키가 있는 열과 같이 두 가지 열 유형을 포함합니다.

## 빠른 실패

개발 수명 주기를 줄이기 위해 빈번한 증분 테스트를 사용하는 철학입니다. 애자일 접근 방식의 핵심입니다.

## 장애 격리 경계

에서 장애의 영향을 제한하고 워크로드의 복원력을 개선하는 데 도움이 되는 가용 영역, AWS 리전 컨트롤 플레인 또는 데이터 플레인과 같은 AWS 클라우드경계입니다. 자세한 내용은 [AWS 장애 격리 경계](#)를 참조하세요.

## 기능 브랜치

[브랜치](#)를 참조하세요.

## 기능

예측에 사용하는 입력 데이터입니다. 예를 들어, 제조 환경에서 기능은 제조 라인에서 주기적으로 캡처되는 이미지일 수 있습니다.

## 기능 중요도

모델의 예측에 특성이 얼마나 중요한지를 나타냅니다. 이는 일반적으로 SHAP(Shapley Additive Descriptions) 및 통합 그래디언트와 같은 다양한 기법을 통해 계산할 수 있는 수치 점수로 표현됩니다. 자세한 내용은 [기계 학습 모델 해석 가능성을 참조하세요 AWS](#).

## 기능 변환

추가 소스로 데이터를 보강하거나, 값을 조정하거나, 단일 데이터 필드에서 여러 정보 세트를 추출하는 등 ML 프로세스를 위해 데이터를 최적화하는 것입니다. 이를 통해 ML 모델이 데이터를 활용할 수 있습니다. 예를 들어, 날짜 '2021-05-27 00:15:37'을 '2021년', '5월', '목', '15일'로 분류하면 학습 알고리즘이 다양한 데이터 구성 요소와 관련된 미묘한 패턴을 학습하는 데 도움이 됩니다.

## 퓨샷 프롬프팅

유사한 태스크를 수행하도록 요청하기 전에 [LLM](#)에 태스크와 원하는 출력을 보여주는 몇 가지 예제를 제공합니다. 이 기법은 모델이 프롬프트에 포함된 예제(샷)에서 학습하는 컨텍스트 내 학습을 적용합니다. 퓨샷 프롬프팅은 특정 형식 지정, 추론 또는 분야별 지식이 필요한 태스크에 효과적일 수 있습니다. [제로샷 프롬프팅](#)도 참조하세요.

## FGAC

[세분화된 액세스 제어](#)를 참조하세요.

## 세분화된 액세스 제어(FGAC)

여러 조건을 사용하여 액세스 요청을 허용하거나 거부합니다.

## 플래시컷 마이그레이션

단계적 접근 방식을 사용하는 대신 [변경 데이터 캡처](#)를 통해 지속적 데이터 복제를 사용하여 최단 시간에 데이터를 마이그레이션하는 데이터베이스 마이그레이션 방법입니다. 목표는 가동 중지 시간을 최소화하는 것입니다.

### FM

[파운데이션 모델](#)을 참조하세요.

#### 파운데이션 모델(FM)

일반화되고 레이블이 지정되지 않은 데이터의 대규모 데이터세트에서 훈련된 대규모 딥 러닝 신경망입니다. FM은 언어 이해, 텍스트 및 이미지 생성, 자연어 대화와 같은 다양한 일반 태스크를 수행할 수 있습니다. 자세한 내용은 [파운데이션 모델이란?](#)을 참조하세요.

#### FM 게이트웨이

[파운데이션 모델에](#) 대한 액세스를 제어하고 정규화하는 중앙 집중식 중개자입니다. LLM 게이트웨이이라고도 합니다.

## G

### 생성형 AI

대량의 데이터에서 훈련되었으며 간단한 텍스트 프롬프트를 사용하여 이미지, 비디오, 텍스트, 오디오와 같은 새 콘텐츠와 아티팩트를 생성할 수 있는 [AI](#) 모델의 하위 세트입니다. 자세한 내용은 [생성형 AI란 무엇인가요?](#)를 참조하세요.

#### 지리적 차단

[지리적 제한](#)을 참조하세요.

#### 지리적 제한(지리적 차단)

Amazon CloudFront에서 특정 국가의 사용자가 콘텐츠 배포에 액세스하지 못하도록 하는 옵션입니다. 허용 목록 또는 차단 목록을 사용하여 승인된 국가와 차단된 국가를 지정할 수 있습니다. 자세한 내용은 CloudFront 설명서의 [콘텐츠의 지리적 배포 제한](#)을 참조하십시오.

#### Gitflow 워크플로

하위 환경과 상위 환경이 소스 코드 리포지토리의 서로 다른 브랜치를 사용하는 방식입니다. Gitflow 워크플로는 레거시로 간주되며 [트렁크 기반 워크플로](#)는 선호되는 현대적 접근 방식입니다.

## 골든 이미지

시스템 또는 소프트웨어의 새 인스턴스를 배포하기 위한 템플릿으로 사용되는 해당 시스템 또는 소프트웨어의 스냅샷입니다. 예를 들어 제조 분야에서는 골든 이미지를 사용하여 여러 디바이스에서 소프트웨어를 프로비저닝할 수 있으며 이를 통해 디이스 제조 작업의 속도, 확장성 및 생산성을 개선할 수 있습니다.

## 브라운필드 전략

새로운 환경에서 기존 인프라의 부재 시스템 아키텍처에 대한 그린필드 전략을 채택할 때 [브라운필드](#)라고도 하는 기존 인프라와의 호환성 제한 없이 모든 새로운 기술을 선택할 수 있습니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 그린필드 전략을 혼합할 수 있습니다.

## 가드레일

조직 단위(OU) 전체에서 리소스, 정책 및 규정 준수를 관리하는 데 도움이 되는 중요 규칙입니다. 예방 가드레일은 규정 준수 표준에 부합하도록 정책을 시행하며, 서비스 제어 정책과 IAM 권한 경계를 사용하여 구현됩니다. 탐지 가드레일은 정책 위반 및 규정 준수 문제를 감지하고 해결을 위한 알림을 생성하며, 이는 AWS Config Amazon GuardDuty AWS Security Hub CSPM, , AWS Trusted Advisor Amazon Inspector 및 사용자 지정 AWS Lambda 검사를 사용하여 구현됩니다.

## 가드레일(AI)

책임감 있고 안전한 AI 동작을 보장하기 위해 [에이전트](#) 입력 및 출력을 필터링, 검증 및 제약하는 안전 메커니즘입니다.

# H

## HA

[고가용성](#)을 참조하세요.

## 이기종 데이터베이스 마이그레이션

다른 데이터베이스 엔진을 사용하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Oracle에서 Amazon Aurora로) 이기종 마이그레이션은 일반적으로 리아키텍트 작업의 일부이며 스키마를 변환하는 것은 복잡한 작업일 수 있습니다. AWS 는 스키마 변환에 도움이 되는 [AWS SCT](#)를 [제공](#)합니다.

## 높은 가용성(HA)

문제나 재해 발생 시 개입 없이 지속적으로 운영할 수 있는 워크로드의 능력. HA 시스템은 자동으로 장애 조치되고, 지속적으로 고품질 성능을 제공하고, 성능에 미치는 영향을 최소화하면서 다양한 부하와 장애를 처리하도록 설계되었습니다.

## 히스토리언 현대화

제조 산업의 요구 사항을 더 잘 충족하도록 운영 기술(OT) 시스템을 현대화하고 업그레이드하는 데 사용되는 접근 방식입니다. 히스토리언은 공장의 다양한 출처에서 데이터를 수집하고 저장하는 데 사용되는 일종의 데이터베이스입니다.

## 홀드아웃 데이터

[기계 학습](#) 모델을 훈련하는 데 사용되는 데이터세트에서 보류되는 레이블이 지정된 기록 데이터의 일부입니다. 홀드아웃 데이터를 사용하여 모델 예측을 홀드아웃 데이터와 비교해 모델 성능을 평가할 수 있습니다.

## human-in-the-loop(HitL)

중요한 결정 시점에서 인적 검토 및 승인을 위해 [에이전트](#) 실행이 일시 중지되는 워크플로 패턴입니다.

## 동종 데이터베이스 마이그레이션

동일한 데이터베이스 엔진을 공유하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Microsoft SQL Server에서 Amazon RDS for SQL Server로) 동종 마이그레이션은 일반적으로 리호스팅 또는 리플랫폼 작업의 일부입니다. 네이티브 데이터베이스 유틸리티를 사용하여 스키마를 마이그레이션할 수 있습니다.

## 핫 데이터

자주 액세스하는 데이터(예: 실시간 데이터 또는 최근 번역 데이터). 일반적으로 이 데이터에는 빠른 쿼리 응답을 제공하기 위한 고성능 스토리지 계층 또는 클래스가 필요합니다.

## 핫픽스

프로덕션 환경의 중요한 문제를 해결하기 위한 긴급 수정입니다. 핫픽스는 긴급하기 때문에 일반적인 DevOps 릴리스 워크플로 외부에서 실행됩니다.

## 하이퍼케어 기간

전환 직후 마이그레이션 팀이 문제를 해결하기 위해 클라우드에서 마이그레이션된 애플리케이션을 관리하고 모니터링하는 기간입니다. 일반적으로 이 기간은 1~4일입니다. 하이퍼케어 기간이 끝나면 마이그레이션 팀은 일반적으로 애플리케이션에 대한 책임을 클라우드 운영 팀에 넘깁니다.

## I

## IaC

[코드형 인프라](#)를 참조하세요.

## 자격 증명 기반 정책

AWS 클라우드 환경 내에서 권한을 정의하는 하나 이상의 IAM 보안 주체에 연결된 정책입니다.

## 유휴 애플리케이션

90일 동안 평균 CPU 및 메모리 사용량이 5~20%인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하거나 온프레미스에 유지하는 것이 일반적입니다.

## IIoT

[산업용 사물 인터넷](#)을 참조하세요.

## 변경 불가능한 인프라

기존 인프라를 업데이트, 패치 또는 수정하는 대신 프로덕션 워크로드에 대한 새 인프라를 배포하는 모델입니다. 변경 불가능한 인프라는 [변경 가능한 인프라](#)보다 본질적으로 더 일관되고 안정적이며 예측 가능합니다. 자세한 내용은 AWS Well-Architected Framework의 [변경 불가능한 인프라를 사용하여 배포](#) 모범 사례를 참조하세요.

## 인바운드(수신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 외부에서 네트워크 연결을 수락, 검사 및 라우팅하는 VPC입니다. [AWS Security Reference Architecture](#)에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

## 증분 마이그레이션

한 번에 전체 전환을 수행하는 대신 애플리케이션을 조금씩 마이그레이션하는 전환 전략입니다. 예를 들어, 처음에는 소수의 마이크로서비스나 사용자만 새 시스템으로 이동할 수 있습니다. 모든 것이 제대로 작동하는지 확인한 후에는 레거시 시스템을 폐기할 수 있을 때까지 추가 마이크로서비스 또는 사용자를 점진적으로 이동할 수 있습니다. 이 전략을 사용하면 대규모 마이그레이션과 관련된 위험을 줄일 수 있습니다.

## Industry 4.0

연결성, 실시간 데이터, 자동화, 분석 및 AI/ML의 발전을 통해 제조 프로세스의 현대화를 나타내기 위해 2016년에 [Klaus Schwab](#)에서 도입한 용어입니다.

## 인프라

애플리케이션의 환경 내에 포함된 모든 리소스와 자산입니다.

### 코드형 인프라(IaC)

구성 파일 세트를 통해 애플리케이션의 인프라를 프로비저닝하고 관리하는 프로세스입니다. IaC는 새로운 환경의 반복 가능성, 신뢰성 및 일관성을 위해 인프라 관리를 중앙 집중화하고, 리소스를 표준화하고, 빠르게 확장할 수 있도록 설계되었습니다.

### 산업용 사물 인터넷(IIoT)

제조, 에너지, 자동차, 의료, 생명과학, 농업 등의 산업 부문에서 인터넷에 연결된 센서 및 디바이스의 사용 자세한 내용은 [산업용 사물 인터넷\(IoT\) 디지털 트랜스포메이션 전략 구축](#)을 참조하십시오.

### 검사 VPC

AWS 다중 계정 아키텍처에서는 VPC(동일하거나 다른 AWS 리전), 인터넷 및 온프레미스 네트워크 간의 네트워크 트래픽 검사를 관리하는 중앙 집중식 VPCs입니다. [AWS Security Reference Architecture](#)에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

### 사물 인터넷(IoT)

인터넷이나 로컬 통신 네트워크를 통해 다른 디바이스 및 시스템과 통신하는 센서 또는 프로세서가 내장된 연결된 물리적 객체의 네트워크 자세한 내용은 [IoT란?](#)을 참조하십시오.

### 해석력

모델의 예측이 입력에 따라 어떻게 달라지는지를 사람이 이해할 수 있는 정도를 설명하는 기계 학습 모델의 특성입니다. 자세한 내용은 [기계 학습 모델 해석 가능성을 참조하세요 AWS](#).

### IoT

[사물 인터넷](#)을 참조하세요.

### IT 정보 라이브러리(ITIL)

IT 서비스를 제공하고 이러한 서비스를 비즈니스 요구 사항에 맞게 조정하기 위한 일련의 모범 사례 ITIL은 ITSM의 기반을 제공합니다.

### IT 서비스 관리(ITSM)

조직의 IT 서비스 설계, 구현, 관리 및 지원과 관련된 활동 클라우드 운영을 ITSM 도구와 통합하는 방법에 대한 자세한 내용은 [운영 통합 가이드](#)를 참조하십시오.

## ITIL

[IT 정보 라이브러리](#)를 참조하세요.

## ITSM

[IT 서비스 관리](#)를 참조하세요.

## L

### 레이블 기반 액세스 제어(LBAC)

사용자 및 데이터 자체에 각각 보안 레이블 값을 명시적으로 할당하는 필수 액세스 제어(MAC)를 구현한 것입니다. 사용자 보안 레이블과 데이터 보안 레이블 간의 교차 부분에 따라 사용자가 볼 수 있는 행과 열이 결정됩니다.

### 랜딩 존

랜딩 존은 확장 가능하고 안전한 잘 설계된 다중 계정 AWS 환경입니다. 조직은 여기에서부터 보안 및 인프라 환경에 대한 확신을 가지고 워크로드와 애플리케이션을 신속하게 시작하고 배포할 수 있습니다. 랜딩 존에 대한 자세한 내용은 [안전하고 확장 가능한 다중 계정 AWS 환경 설정](#)을 참조하십시오.

### 대규모 언어 모델(LLM)

방대한 양의 데이터에서 사전 훈련된 딥 러닝 [AI](#) 모델입니다. LLM은 질문에 대한 답변, 문서 요약, 텍스트를 다른 언어로 번역, 문장 완성과 같은 여러 태스크를 수행할 수 있습니다. 자세한 내용은 [대규모 언어 모델\(LLM\)이란 무엇인가요?](#)를 참조하세요.

### 대규모 마이그레이션

300대 이상의 서버 마이그레이션입니다.

### LBAC

[레이블 기반 액세스 제어](#)를 참조하세요.

### 최소 권한

작업을 수행하는 데 필요한 최소 권한을 부여하는 보안 모범 사례입니다. 자세한 내용은 IAM 설명서의 [최소 권한 적용](#)을 참조하십시오.

### 리프트 앤드 시프트

[7R](#)을 참조하세요.

## 리틀 엔디안 시스템

가장 덜 중요한 바이트를 먼저 저장하는 시스템입니다. [엔디안](#)도 참조하세요.

## LLM

[대규모 언어 모델](#)을 참조하세요.

## 하위 환경

[환경](#)을 참조하세요.

## M

### 기계 학습(ML)

패턴 인식 및 학습에 알고리즘과 기법을 사용하는 인공지능의 한 유형입니다. ML은 사물 인터넷 (IoT) 데이터와 같은 기록된 데이터를 분석하고 학습하여 패턴을 기반으로 통계 모델을 생성합니다. 자세한 내용은 [기계 학습](#)을 참조하십시오.

### 기본 브랜치

[브랜치](#)를 참조하세요.

### 맬웨어

컴퓨터 보안 또는 프라이버시를 위협하도록 설계된 소프트웨어입니다. 맬웨어는 컴퓨터 시스템을 방해하거나 민감한 정보를 유출하거나 무단 액세스 권한을 확보할 수 있습니다. 맬웨어의 예로 바이러스, 웜, 랜섬웨어, 트로이 목마, 스파이웨어, 키로거 등이 있습니다.

### 관리형 서비스

AWS 서비스는 인프라 계층, 운영 체제 및 플랫폼을 AWS 운영하며 사용자는 엔드포인트에 액세스하여 데이터를 저장하고 검색합니다. 관리형 서비스의 예로 Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB가 있습니다. 이를 추상화된 서비스라고도 합니다.

### 제조 실행 시스템(MES)

원자재를 생산 현장에서 완제품으로 변환하는 생산 프로세스를 추적, 모니터링, 문서화 및 제어하기 위한 소프트웨어 시스템입니다.

## MAP

[Migration Acceleration Program](#)을 참조하세요.

## MCP

[모델 컨텍스트 프로토콜](#)을 참조하세요.

### Model Context Protocol(MCP)

[에이전트 간??? 통신](#)을 위한 상태 비저장 프로토콜입니다.

### MCP 서버

[모델 컨텍스트 프로토콜](#)을 통해 하나 이상의 [도구](#)를 노출하는 서비스입니다.

### 메커니즘

도구를 생성하고 도구 채택을 유도한 다음 조정을 위해 결과를 검사하는 전체 프로세스입니다. 메커니즘은 작동 시 자체적으로 강화하고 개선하는 주기입니다. 자세한 내용은 AWS Well-Architected Framework의 [메커니즘 구축](#)을 참조하세요.

### 멤버 계정

조직의 일부인 관리 계정을 AWS 계정 제외한 모든 계정. AWS Organizations 하나의 계정은 한 번에 하나의 조직 멤버만 될 수 있습니다.

## MES

[제조 실행 시스템](#)을 참조하세요.

### 메시지 큐 원격 분석 전송(MQTT)

리소스 제약이 있는 [IoT](#) 디바이스에 대한 [게시 및 구독](#) 패턴을 기반으로 하는 경량 Machine-to-Machine(M2M) 통신 프로토콜입니다.

### 마이크로서비스

잘 정의된 API를 통해 통신하고 일반적으로 소규모 자체 팀이 소유하는 소규모 독립 서비스입니다. 예를 들어, 보험 시스템에는 영업, 마케팅 등의 비즈니스 역량이나 구매, 청구, 분석 등의 하위 영역에 매핑되는 마이크로 서비스가 포함될 수 있습니다. 마이크로서비스의 이점으로 민첩성, 유연한 확장, 손쉬운 배포, 재사용 가능한 코드, 복원력 등이 있습니다. 자세한 내용은 [AWS 서버리스 서비스를 사용하여 마이크로서비스 통합](#)을 참조하세요.

### 마이크로서비스 아키텍처

각 애플리케이션 프로세스를 마이크로서비스로 실행하는 독립 구성 요소를 사용하여 애플리케이션을 구축하는 접근 방식입니다. 이러한 마이크로서비스는 경량 API를 사용하여 잘 정의된 인터페이스를 통해 통신합니다. 애플리케이션의 특정 기능에 대한 수요에 맞게 이 아키텍처의 각 마이크로서비스를 업데이트, 배포 및 조정할 수 있습니다. 자세한 내용은 [에서 마이크로서비스 구현을 참조하세요 AWS](#).

## Migration Acceleration Program(MAP)

조직이 클라우드로 전환하기 위한 강력한 운영 기반을 구축하고 초기 마이그레이션 비용을 상쇄하는 데 도움이 되는 컨설팅 지원, 교육 및 서비스를 제공하는 AWS 프로그램입니다. MAP에는 레거시 마이그레이션을 체계적인 방식으로 실행하기 위한 마이그레이션 방법론과 일반적인 마이그레이션 시나리오를 자동화하고 가속화하는 도구 세트가 포함되어 있습니다.

### 대규모 마이그레이션

애플리케이션 포트폴리오의 대다수를 웨이브를 통해 클라우드로 이동하는 프로세스로, 각 웨이브에서 더 많은 애플리케이션이 더 빠른 속도로 이동합니다. 이 단계에서는 이전 단계에서 배운 모범 사례와 교훈을 사용하여 팀, 도구 및 프로세스의 마이그레이션 팩토리를 구현하여 자동화 및 민첩한 제공을 통해 워크로드 마이그레이션을 간소화합니다. 이것은 [AWS 마이그레이션 전략](#)의 세 번째 단계입니다.

### 마이그레이션 팩토리

자동화되고 민첩한 접근 방식을 통해 워크로드 마이그레이션을 간소화하는 다기능 팀입니다. 마이그레이션 팩토리 팀에는 일반적으로 스프린트에서 일하는 운영, 비즈니스 분석가 및 소유자, 마이그레이션 엔지니어, 개발자, DevOps 전문가가 포함됩니다. 엔터프라이즈 애플리케이션 포트폴리오의 20~50%는 공장 접근 방식으로 최적화할 수 있는 반복되는 패턴으로 구성되어 있습니다. 자세한 내용은 이 콘텐츠 세트의 [클라우드 마이그레이션 팩토리 가이드](#)와 [마이그레이션 팩토리에 대한 설명](#)을 참조하십시오.

### 마이그레이션 메타데이터

마이그레이션을 완료하는 데 필요한 애플리케이션 및 서버에 대한 정보 각 마이그레이션 패턴에는 서로 다른 마이그레이션 메타데이터 세트가 필요합니다. 마이그레이션 메타데이터의 예로는 대상 서브넷, 보안 그룹 및 AWS 계정이 있습니다.

### 마이그레이션 패턴

사용되는 마이그레이션 전략, 마이그레이션 대상, 마이그레이션 애플리케이션 또는 서비스를 자세히 설명하는 반복 가능한 마이그레이션 작업입니다. 예: AWS Application Migration Service를 사용하여 Amazon EC2로 마이그레이션을 리호스팅합니다.

## Migration Portfolio Assessment(MPA)

AWS 클라우드로 마이그레이션하는 비즈니스 사례를 검증하기 위한 정보를 제공하는 온라인 도구입니다. MPA는 상세한 포트폴리오 평가(서버 적정 규모 조정, 가격 책정, TCO 비교, 마이그레이션 비용 분석)와 마이그레이션 계획(애플리케이션 데이터 분석 및 데이터 수집, 애플리케이션 그룹화, 마이그레이션 우선순위 지정, 웨이브 계획)을 제공합니다. [MPA 도구](#)(로그인 필요)는 모든 AWS 컨설턴트와 APN 파트너 컨설턴트가 무료로 사용할 수 있습니다.

## 마이그레이션 준비 상태 평가(MRA)

AWS CAF를 사용하여 조직의 클라우드 준비 상태에 대한 인사이트를 얻고, 강점과 약점을 식별하고, 식별된 격차를 해소하기 위한 행동 계획을 수립하는 프로세스입니다. 자세한 내용은 [마이그레이션 준비 가이드](#)를 참조하십시오. MRA는 [AWS 마이그레이션 전략](#)의 첫 번째 단계입니다.

### 마이그레이션 전략

워크로드를 AWS 클라우드로 마이그레이션하는 데 사용되는 접근 방식입니다. 자세한 내용은 [이 용어집의 7R 항목과 조직을 동원하여 대규모 마이그레이션 가속화](#)를 참조하세요.

### ML

[기계 학습](#)을 참조하세요.

### 현대화

비용을 절감하고 효율성을 높이고 혁신을 활용하기 위해 구식(레거시 또는 모놀리식) 애플리케이션과 해당 인프라를 클라우드의 민첩하고 탄력적이고 가용성이 높은 시스템으로 전환하는 것입니다. 자세한 내용은 [AWS 클라우드에서 애플리케이션을 현대화하기 위한 전략](#)을 참조하세요.

### 현대화 준비 상태 평가

조직 애플리케이션의 현대화 준비 상태를 파악하고, 이점, 위험 및 종속성을 식별하고, 조직이 해당 애플리케이션의 향후 상태를 얼마나 잘 지원할 수 있는지를 확인하는 데 도움이 되는 평가입니다. 평가 결과는 대상 아키텍처의 청사진, 현대화 프로세스의 개발 단계와 마일스톤을 자세히 설명하는 로드맵 및 파악된 격차를 해소하기 위한 실행 계획입니다. 자세한 내용은 [AWS 클라우드에서 애플리케이션의 현대화 준비 상태 평가](#)를 참조하세요.

### 모놀리식 애플리케이션(모놀리식 유형)

긴밀하게 연결된 프로세스를 사용하여 단일 서비스로 실행되는 애플리케이션입니다. 모놀리식 애플리케이션에는 몇 가지 단점이 있습니다. 한 애플리케이션 기능에 대한 수요가 급증하면 전체 아키텍처 규모를 조정해야 합니다. 코드 베이스가 커지면 모놀리식 애플리케이션의 기능을 추가하거나 개선하는 것도 더 복잡해집니다. 이러한 문제를 해결하기 위해 마이크로서비스 아키텍처를 사용할 수 있습니다. 자세한 내용은 [마이크로서비스로 모놀리식 유형 분해](#)를 참조하십시오.

### MPA

[Migration Portfolio Assessment](#)를 참조하세요.

### MQTT

[메시지 큐 원격 분석 전송](#)을 참조하세요.

## 멀티클래스 분류

여러 클래스에 대한 예측(2개 이상의 결과 중 하나 예측)을 생성하는 데 도움이 되는 프로세스입니다. 예를 들어, ML 모델이 '이 제품은 책인가요, 자동차인가요, 휴대폰인가요?' 또는 '이 고객이 가장 관심을 갖는 제품 범주는 무엇인가요?'라고 물을 수 있습니다.

## 변경 가능한 인프라

프로덕션 워크로드에 대한 기존 인프라를 업데이트하고 수정하는 모델입니다. 일관성, 신뢰성 및 예측 가능성을 높이기 위해 AWS Well-Architected Framework는 [변경 불가능한 인프라](#)를 모범 사례로 사용할 것을 권장합니다.

## O

### OAC

[오리진 액세스 제어](#)를 참조하세요.

### OAI

[오리진 액세스 ID](#)를 참조하세요.

### OCM

[조직 변경 관리](#)를 참조하세요.

## 오프라인 마이그레이션

마이그레이션 프로세스 중 소스 워크로드가 중단되는 마이그레이션 방법입니다. 이 방법은 가동 중지 증가를 수반하며 일반적으로 작고 중요하지 않은 워크로드에 사용됩니다.

## OI

[운영 통합](#)을 참조하세요.

### OLA

[운영 수준 계약](#)을 참조하세요.

## 온라인 마이그레이션

소스 워크로드를 오프라인 상태로 전환하지 않고 대상 시스템에 복사하는 마이그레이션 방법입니다. 워크로드에 연결된 애플리케이션은 마이그레이션 중에도 계속 작동할 수 있습니다. 이 방법은 가동 중지 차단 또는 최소화를 수반하며 일반적으로 중요한 프로덕션 워크로드에 사용됩니다.

## OPC-UA

[Open Process Communications - Unified Architecture\(OPC-UA\)](#)를 참조하세요.

### Open Process Communications - Unified Architecture(OPC-UA)

산업 자동화를 위한 Machine-to-Machine(M2M) 통신 프로토콜입니다. OPC-UA는 데이터 암호화, 인증 및 권한 부여 체계에 관한 상호 운용성 표준을 제공합니다.

### 운영 수준 협약(OLA)

서비스 수준에 관한 계약(SLA)을 지원하기 위해 직무 IT 그룹이 서로에게 제공하기로 약속한 내용을 명확히 하는 계약입니다.

### 운영 준비 상태 검토(ORR)

인시던트 및 잠재적 장애의 범위를 이해, 평가 또는 예방하거나 줄이는 데 도움이 되는 질문 체크리스트 및 관련 모범 사례입니다. 자세한 내용은 AWS Well-Architected Framework의 [운영 준비 상태 검토\(ORR\)](#)를 참조하세요.

### 운영 기술(OT)

물리적 환경에서 작동하여 산업 운영, 장비 및 인프라를 제어하는 하드웨어 및 소프트웨어 시스템입니다. 제조 분야에서 OT 및 정보 기술(IT) 시스템의 통합은 [Industry 4.0](#) 트랜스포메이션의 주요 중점 사항입니다.

### 운영 통합(OI)

클라우드에서 운영을 현대화하는 프로세스로 준비 계획, 자동화 및 통합을 수반합니다. 자세한 내용은 [운영 통합 가이드](#)를 참조하십시오.

### 조직 트레일

조직 AWS 계정 내 모든에 대한 모든 이벤트를 로깅 AWS CloudTrail 하는에서 생성된 추적입니다 AWS Organizations. 이 트레일은 조직에 속한 각 AWS 계정에 생성되고 각 계정의 활동을 추적합니다. 자세한 내용은 CloudTrail 설명서의 [Creating a trail for an organization](#)을 참조하십시오.

### 조직 변경 관리(OCM)

사람, 문화 및 리더십 관점에서 중대하고 파괴적인 비즈니스 혁신을 관리하기 위한 프레임워크입니다. OCM은 변화 채택을 가속화하고, 과도기적 문제를 해결하고, 문화 및 조직적 변화를 주도함으로써 조직이 새로운 시스템 및 전략을 준비하고 전환할 수 있도록 지원합니다. AWS 마이그레이션 전략에서는 클라우드 채택 프로젝트에 필요한 변경 속도 때문에이 프레임워크를 인력 가속화라고 합니다. 자세한 내용은 [사용 가이드](#)를 참조하십시오.

## 오리진 액세스 제어(OAC)

CloudFront에서 Amazon Simple Storage Service(S3) 콘텐츠를 보호하기 위해 액세스를 제한하는 고급 옵션입니다. OAC는 AWS KMS (SSE-KMS)를 사용한 모든 서버 측 암호화 AWS 리전와 S3 버킷에 대한 동적 PUT 및 DELETE 요청에서 모든 S3 버킷을 지원합니다.

## 오리진 액세스 ID(OAI)

CloudFront에서 Amazon S3 콘텐츠를 보호하기 위해 액세스를 제한하는 옵션입니다. OAI를 사용하면 CloudFront는 Amazon S3가 인증할 수 있는 보안 주체를 생성합니다. 인증된 보안 주체는 특정 CloudFront 배포를 통해서만 S3 버킷의 콘텐츠에 액세스할 수 있습니다. 더 세분화되고 향상된 액세스 제어를 제공하는 [OAC](#)도 참조하십시오.

## ORR

[운영 준비 상태 검토](#)를 참조하세요.

## OT

[운영 기술](#)을 참조하세요.

## 아웃바운드(송신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 내에서 시작된 네트워크 연결을 처리하는 VPC입니다. [AWS Security Reference Architecture](#)에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

## P

### 권한 경계

사용자나 역할이 가질 수 있는 최대 권한을 설정하기 위해 IAM 보안 주체에 연결되는 IAM 관리 정책입니다. 자세한 내용은 IAM 설명서의 [권한 경계](#)를 참조하십시오.

### 개인 식별 정보(PII)

직접 보거나 다른 관련 데이터와 함께 짝을 지을 때 개인의 신원을 합리적으로 추론하는 데 사용할 수 있는 정보입니다. PII의 예로는 이름, 주소, 연락처 정보 등이 있습니다.

## PII

[개인 식별 정보](#)를 참조하세요.

## 플레이북

클라우드에서 핵심 운영 기능을 제공하는 등 마이그레이션과 관련된 작업을 캡처하는 일련의 사전 정의된 단계입니다. 플레이북은 스크립트, 자동화된 런북 또는 현대화된 환경을 운영하는 데 필요한 프로세스나 단계 요약의 형태를 취할 수 있습니다.

## PLC

[프로그래밍 가능 로직 컨트롤러](#)를 참조하세요.

## PLM

[제품 수명 주기 관리](#)를 참조하세요.

## 정책

권한 정의([ID 기반 정책](#) 참조), 액세스 조건 지정([리소스 기반 정책](#) 참조), AWS Organizations 내 조직의 모든 계정에 대한 최대 권한 정의([서비스 제어 정책](#) 참조)와 같은 작업을 수행할 수 있는 객체입니다.

## 다국어 지속성

데이터 액세스 패턴 및 기타 요구 사항을 기반으로 독립적으로 마이크로서비스의 데이터 스토리지 기술 선택. 마이크로서비스가 동일한 데이터 스토리지 기술을 사용하는 경우 구현 문제가 발생하거나 성능이 저하될 수 있습니다. 요구 사항에 가장 적합한 데이터 저장소를 사용하면 마이크로서비스를 더 쉽게 구현하고 성능과 확장성을 높일 수 있습니다.

## 포트폴리오 평가

마이그레이션을 계획하기 위해 애플리케이션 포트폴리오를 검색 및 분석하고 우선순위를 정하는 프로세스입니다. 자세한 내용은 [마이그레이션 준비 상태 평가](#)를 참조하십시오.

## 조건자

보통 WHERE 절에 있는 true 또는 false를 반환하는 쿼리 조건입니다.

## 푸시다운 조건자

전송 전에 쿼리의 데이터를 필터링하는 데이터베이스 쿼리 최적화 기법입니다. 이렇게 하면 관계형 데이터베이스에서 검색하고 처리해야 하는 데이터의 양이 줄고 쿼리 성능이 향상됩니다.

## 예방적 제어

이벤트 발생을 방지하도록 설계된 보안 제어입니다. 이 제어는 네트워크에 대한 무단 액세스나 원치 않는 변경을 방지하는 데 도움이 되는 1차 방어선입니다. 자세한 내용은 Implementing security controls on AWS의 [Preventative controls](#)를 참조하십시오.

## 보안 주체

작업을 수행하고 리소스에 액세스할 수 있는 AWS IAM 엔티티입니다. 이 엔티티는 일반적으로 , AWS 계정 IAM 역할 또는 사용자의 루트 사용자입니다. 자세한 내용은 IAM 설명서의 [역할 용어 및 개념](#)의 보안 주체를 참조하십시오.

## 개인 정보 보호 중심 설계

전체 개발 프로세스에서 개인 정보를 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

## 프라이빗 호스팅 영역

Amazon Route 53에서 하나 이상의 VPC 내 도메인과 하위 도메인에 대한 DNS 쿼리에 응답하는 방법에 대한 정보가 담긴 컨테이너입니다. 자세한 내용은 Route 53 설명서의 [프라이빗 호스팅 영역 작업](#)을 참조하십시오.

## 선제적 제어

규정 미준수 리소스의 배포를 방지하도록 설계된 [보안 제어](#)입니다. 이러한 제어는 리소스를 프로비저닝하기 전에 리소스를 스캔합니다. 리소스가 제어를 준수하지 않으면 프로비저닝되지 않습니다. 자세한 내용은 AWS Control Tower 설명서의 [제어 참조 가이드](#)를 참조하고 보안 [제어 구현의 사전 예방적 제어](#)를 참조하세요. AWS

## 제품 수명 주기 관리(PLM)

설계, 개발 및 출시부터 성장 및 성숙도를 거쳐 거부 및 제거에 이르기까지 전체 수명 주기 동안 제품의 데이터 및 프로세스 관리를 나타냅니다.

## 프로덕션 환경

[환경](#)을 참조하세요.

## 프로그래밍 가능 로직 컨트롤러(PLC)

제조 분야에서 기계를 모니터링하고 제조 프로세스를 자동화하는 매우 안정적이고 적응력이 뛰어난 컴퓨터입니다.

## 프롬프트 체이닝

한 [LLM](#) 프롬프트의 출력을 다음 프롬프트의 입력으로 사용하여 더 나은 응답을 생성합니다. 이 기법은 복잡한 태스크를 하위 태스크로 나누거나 예비 응답을 반복적으로 세부 조정하거나 확장하는데 사용됩니다. 이를 통해 모델 응답의 정확성과 관련성을 개선하고 보다 세분화되고 개인화된 결과를 얻을 수 있습니다.

## 가명화

데이터세트의 개인 식별자를 자리 표시자 값으로 바꾸는 프로세스입니다. 가명화는 개인 정보를 보호하는 데 도움이 될 수 있습니다. 가명화된 데이터는 여전히 개인 데이터로 간주됩니다.

## 게시/구독(pub/sub)

여러 마이크로서비스에서 비동기 통신을 지원하여 확장성과 응답성을 개선하는 패턴입니다. 예를 들어 마이크로서비스 기반 [MES](#)에서 마이크로서비스는 다른 마이크로서비스가 구독할 수 있는 채널에 이벤트 메시지를 게시할 수 있습니다. 시스템은 게시 서비스를 변경하지 않고도 새 마이크로 서비스를 추가할 수 있습니다.

## Q

### 쿼리 계획

SQL 관계형 데이터베이스 시스템의 데이터에 액세스하는 데 사용되는 명령어와 같은 일련의 단계입니다.

### 쿼리 계획 회귀

데이터베이스 서비스 최적화 프로그램이 데이터베이스 환경을 변경하기 전보다 덜 최적의 계획을 선택하는 경우입니다. 통계, 제한 사항, 환경 설정, 쿼리 파라미터 바인딩 및 데이터베이스 엔진 업데이트의 변경으로 인해 발생할 수 있습니다.

## R

### RACI 매트릭스

[Responsible, Accountable, Consulted, Informed\(RACI\)](#)를 참조하세요.

### RAG

[검색 증강 생성](#)을 참조하세요.

### 랜섬웨어

결제 완료될 때까지 컴퓨터 시스템이나 데이터에 대한 액세스를 차단하도록 설계된 악성 소프트웨어입니다.

### RASCI 매트릭스

[Responsible, Accountable, Consulted, Informed\(RACI\)](#)를 참조하세요.

## RCAC

[행 및 열 액세스 제어](#)를 참조하세요.

### 읽기 전용 복제본

읽기 전용 용도로 사용되는 데이터베이스의 사본입니다. 쿼리를 읽기 전용 복제본으로 라우팅하여 기본 데이터베이스의 로드를 줄일 수 있습니다.

### 리아키텍팅

[7R](#)을 참조하세요.

### Recovery Point Objective(RPO)

마지막 데이터 복구 시점 이후 허용되는 최대 시간입니다. 이에 따라 마지막 복구 시점과 서비스 중단 사이에 허용되는 데이터 손실로 간주되는 범위가 결정됩니다.

### Recovery Time Objective(RTO)

서비스 중단과 서비스 복원 사이의 허용 가능한 지연 시간입니다.

### 리팩터링

[7R](#)을 참조하세요.

### 리전

지리적 영역의 AWS 리소스 모음입니다. 각 AWS 리전은 내결함성, 안정성 및 복원력을 제공하기 위해 서로 격리되고 독립적입니다. 자세한 내용은 [계정에서 사용할 수 있는 AWS 리전 지정](#)을 참조하세요.

### 회귀

숫자 값을 예측하는 ML 기법입니다. 예를 들어, '이 집은 얼마에 팔릴까?'라는 문제를 풀기 위해 ML 모델은 선형 회귀 모델을 사용하여 주택에 대해 알려진 사실(예: 면적)을 기반으로 주택의 매매 가격을 예측할 수 있습니다.

### 리호스팅

[7R](#)을 참조하세요.

### 릴리스

배포 프로세스에서 변경 사항을 프로덕션 환경으로 승격시키는 행위입니다.

## 재배치

[7R](#)을 참조하세요.

## 리플랫폼

[7R](#)을 참조하세요.

## 재구매

[7R](#)을 참조하세요.

## 복원력

중단에 저항하거나 중단을 복구할 수 있는 애플리케이션의 기능입니다. [고가용성 및 재해 복구](#)는 AWS 클라우드에서 복원력을 계획할 때 일반적인 고려 사항입니다. 자세한 내용은 [AWS 클라우드 복원력](#)을 참조하세요.

## 리소스 기반 정책

Amazon S3 버킷, 엔드포인트, 암호화 키 등의 리소스에 연결된 정책입니다. 이 유형의 정책은 액세스가 허용된 보안 주체, 지원되는 작업 및 충족해야 하는 기타 조건을 지정합니다.

## RACI(Responsible, Accountable, Consulted, Informed) 매트릭스

마이그레이션 활동 및 클라우드 운영에 참여하는 모든 당사자의 역할과 책임을 정의하는 매트릭스입니다. 매트릭스 이름은 매트릭스에 정의된 책임 유형에서 파생됩니다. 실무 담당자 (R), 의사 결정권자 (A), 업무 수행 조언자 (C), 결과 통보 대상자 (I). 지원자는 (S) 선택사항입니다. 지원자를 포함하면 매트릭스를 RASCI 매트릭스라고 하고, 지원자를 제외하면 RACI 매트릭스라고 합니다.

## 대응 제어

보안 기준에서 벗어나거나 부정적인 이벤트를 해결하도록 설계된 보안 제어입니다. 자세한 내용은 AWS에서 보안 제어 구현의 [대응 제어](#)를 참조하세요.

## retain

[7R](#)을 참조하세요.

## 사용 중지

[7R](#)을 참조하세요.

## 검색 증강 세대(RAG)

응답을 생성하기 전에 [LLM](#)이 훈련 데이터 소스 외부에 있는 신뢰할 수 있는 데이터 소스를 참조하는 [생성형 AI](#) 기술입니다. 예를 들어 RAG 모델은 조직의 지식 기반 또는 사용자 지정 데이터에 대

한 시맨틱 검색을 수행할 수 있습니다. 자세한 내용은 [검색 증강 생성\(RAG\)이란 무엇인가요?](#)를 참조하세요.

## 교체

공격자가 자격 증명에 액세스하는 것을 더욱 어렵게 만들기 위해 [보안 암호](#)를 주기적으로 업데이트하는 프로세스입니다.

## 행 및 열 액세스 제어(RCAC)

액세스 규칙이 정의된 기본적이고 유연한 SQL 표현식을 사용합니다. RCAC는 행 권한과 열 마스크로 구성됩니다.

## RPO

[목표 복구 시점\(RPO\)](#)을 참조하세요.

## RTO

[목표 복구 시간\(RTO\)](#)을 참조하세요.

## 런북

특정 작업을 수행하는 데 필요한 일련의 수동 또는 자동 절차입니다. 일반적으로 오류율이 높은 반복 작업이나 절차를 간소화하기 위해 런북을 만듭니다.

# S

## SAML 2.0

많은 ID 제공업체(idP)에서 사용하는 개방형 표준입니다. 이 기능을 사용하면 연동 SSO(Single Sign-On)를 AWS Management Console 사용할 수 있으므로 사용자는 조직의 모든 사용자에게 대해 IAM에서 사용자를 생성하지 않고도 로그인하거나 AWS API 작업을 호출할 수 있습니다. SAML 2.0 기반 페더레이션에 대한 자세한 내용은 IAM 설명서의 [SAML 2.0 기반 페더레이션 정보](#)를 참조하십시오.

## SCADA

[감독 제어 및 데이터 획득](#)을 참조하세요.

## SCP

[서비스 제어 정책](#)을 참조하세요.

## 보안 암호

에는 암호 또는 사용자 자격 증명과 같이 암호화된 형식으로 저장하는 AWS Secrets Manager 기밀 또는 제한된 정보가 있습니다. 보안 암호 값과 메타데이터로 구성됩니다. 보안 암호 값은 바이너리, 단일 문자열 또는 여러 문자열일 수 있습니다. 자세한 내용은 AWS Secrets Manager 설명서의 [Secrets Manager 보안 암호란 무엇인가요?](#)를 참조하세요.

## 보안 중심 설계

전체 개발 프로세스에서 보안을 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

## 보안 제어

위협 행위자가 보안 취약성을 악용하는 능력을 방지, 탐지 또는 감소시키는 기술적 또는 관리적 가드레일입니다. 보안 제어는 [예방](#), [감지](#), [대응](#), [선제적](#)과 같은 기본적인 네 가지 보안 제어 유형으로 구분됩니다.

## 보안 강화

공격 표면을 줄여 공격에 대한 저항력을 높이는 프로세스입니다. 더 이상 필요하지 않은 리소스 제거, 최소 권한 부여의 보안 모범 사례 구현, 구성 파일의 불필요한 기능 비활성화 등의 작업이 여기에 포함될 수 있습니다.

## 보안 정보 및 이벤트 관리(SIEM) 시스템

보안 정보 관리(SIM)와 보안 이벤트 관리(SEM) 시스템을 결합하는 도구 및 서비스입니다. SIEM 시스템은 서버, 네트워크, 디바이스 및 기타 소스에서 데이터를 수집, 모니터링 및 분석하여 위협과 보안 침해를 탐지하고 알림을 생성합니다.

## 보안 응답 자동화

보안 이벤트에 자동으로 응답하거나 이를 해결하도록 설계된 사전 정의되고 프로그래밍된 작업입니다. 이러한 자동화는 보안 모범 사례를 구현하는 데 도움이 되는 [탐지](#) 또는 [대응](#) AWS 보안 제어 역할을 합니다. 자동화된 응답 작업의 예로 VPC 보안 그룹 수정, Amazon EC2 인스턴스 패치 적용 또는 자격 증명 교체 등이 있습니다.

## 서버 측 암호화

데이터를 AWS 서비스 수신하는가 대상에서 데이터를 암호화합니다.

## 서비스 제어 정책(SCP)

AWS Organizations에 속한 조직의 모든 계정에 대한 권한을 중앙 집중식으로 제어하는 정책입니다. SCP는 관리자가 사용자 또는 역할에 위임할 수 있는 작업에 대해 제한을 설정하거나 가드레일을 정의합니다. SCP를 허용 목록 또는 거부 목록으로 사용하여 허용하거나 금지할 서비스 또는 작

업을 지정할 수 있습니다. 자세한 내용은 AWS Organizations 설명서의 [서비스 제어 정책을](#) 참조하세요.

## 서비스 엔드포인트

에 대한 진입점의 URL입니다 AWS 서비스. 엔드포인트를 사용하여 대상 서비스에 프로그래밍 방식으로 연결할 수 있습니다. 자세한 내용은 AWS 일반 참조의 [AWS 서비스 엔드포인트](#)를 참조하십시오.

## 서비스 수준에 관한 계약(SLA)

IT 팀이 고객에게 제공하기로 약속한 내용(예: 서비스 가동 시간 및 성능)을 명시한 계약입니다.

## 서비스 수준 지표(SLI)

오류 발생률, 가용성 또는 처리량과 같은 서비스의 성능 측면에 대한 측정값입니다.

## 서비스 수준 목표(SLO)

[서비스 수준 지표](#)로 측정되는 서비스의 상태를 나타내는 목표 지표입니다.

## 공동 책임 모델

클라우드 보안 및 규정 준수를 AWS 위해와 공유하는 책임을 설명하는 모델입니다. AWS 는 클라우드의 보안을 책임지고, 사용자는 클라우드의 보안을 책임집니다. 자세한 내용은 [공동 책임 모델](#)을 참조하십시오.

## 새도우 AI

조직 내 관리형 채널 외부에서 구축되거나 사용되는 승인되지 않은 [AI](#) 애플리케이션입니다.

## SIEM

[보안 정보 및 이벤트 관리 시스템](#)을 참조하세요.

## 단일 장애점(SPOF)

애플리케이션을 중단시킬 수 있는 애플리케이션의 중요한 단일 구성 요소에서 발생하는 장애입니다.

## SLA

[서비스 수준 계약](#)을 참조하세요.

## SLI

[서비스 수준 지표](#)를 참조하세요.

## SLO

[서비스 수준 목표](#)를 참조하세요.

### 분할 앤 시드 모델

현대화 프로젝트를 확장하고 가속화하기 위한 패턴입니다. 새로운 기능과 제품 릴리스가 정의되면 핵심 팀이 분할되어 새로운 제품 팀이 만들어집니다. 이를 통해 조직의 역량과 서비스 규모를 조정하고, 개발자 생산성을 개선하고, 신속한 혁신을 지원할 수 있습니다. 자세한 내용은 [AWS 클라우드에서 애플리케이션을 현대화하기 위한 단계별 접근 방식](#)을 참조하세요.

## SPOF

[단일 장애점](#)을 참조하세요.

### 스타 스키마

하나의 큰 팩트 테이블을 사용하여 트랜잭션 또는 측정된 데이터를 저장하고 하나 이상의 더 작은 차원 테이블을 사용하여 데이터 속성을 저장하는 데이터베이스 조직 구조입니다. 이 구조는 [데이터 웨어하우스](#)에서 또는 비즈니스 인텔리전스 목적으로 사용하도록 설계되었습니다.

### Strangler Fig 패턴

레거시 시스템을 폐기할 수 있을 때까지 시스템 기능을 점진적으로 다시 작성하고 교체하여 모놀리식 시스템을 현대화하기 위한 접근 방식. 이 패턴은 무화과 덩굴이 나무로 자라 결국 숙주를 압도하고 대체하는 것과 비슷합니다. [Martin Fowler](#)가 모놀리식 시스템을 다시 작성할 때 위험을 관리하는 방법으로 이 패턴을 도입했습니다. 이 패턴을 적용하는 방법의 예는 [컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET\(ASMX\) 웹 서비스를 점진적으로 현대화하는 방법](#)을 참조하십시오.

### 서브넷

VPC의 IP 주소 범위입니다. 서브넷은 단일 가용 영역에 상주해야 합니다.

### 감독 제어 및 데이터 획득(SCADA)

제조 분야에서 하드웨어와 소프트웨어를 사용하여 물리적 자산과 프로덕션 작업을 모니터링하는 시스템입니다.

### 대칭 암호화

동일한 키를 사용하여 데이터를 암호화하고 복호화하는 암호화 알고리즘입니다.

## 합성 테스트

사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 감지하거나 성능을 모니터링하는 방식으로 진행되는 시스템 테스트입니다. [Amazon CloudWatch Synthetics](#)를 사용하여 이러한 테스트를 생성할 수 있습니다.

## 시스템 프롬프트

[LLM](#)에 컨텍스트, 명령 또는 지침을 제공하여 동작을 지시하는 기법입니다. 시스템 프롬프트는 컨텍스트를 설정하고 사용자와의 상호 작용을 위한 규칙을 설정하는 데 도움이 됩니다.

# T

## tags

AWS 리소스를 구성하기 위한 메타데이터 역할을 하는 키-값 페어입니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색, 필터링할 수 있습니다. 자세한 내용은 [AWS 리소스에 태그 지정](#)을 참조하십시오.

## 대상 변수

지도 ML에서 예측하려는 값으로, 결과 변수라고도 합니다. 예를 들어, 제조 설정에서 대상 변수는 제품 결함일 수 있습니다.

## 작업 목록

런북을 통해 진행 상황을 추적하는 데 사용되는 도구입니다. 작업 목록에는 런북의 개요와 완료해야 할 일반 작업 목록이 포함되어 있습니다. 각 일반 작업에 대한 예상 소요 시간, 소유자 및 진행 상황이 작업 목록에 포함됩니다.

## 테스트 환경

[환경](#)을 참조하세요.

## 훈련

ML 모델이 학습할 수 있는 데이터를 제공하는 것입니다. 훈련 데이터에는 정답이 포함되어야 합니다. 학습 알고리즘은 훈련 데이터에서 대상(예측하려는 답)에 입력 데이터 속성을 매핑하는 패턴을 찾고, 이러한 패턴을 캡처하는 ML 모델을 출력합니다. 그런 다음 ML 모델을 사용하여 대상을 모르는 새 데이터에 대한 예측을 할 수 있습니다.

## tool

[에이전트](#)가 외부 시스템에서 작업을 수행하기 위해 호출할 수 있는 함수 또는 API입니다.

## Transit Gateway

VPC와 온프레미스 네트워크를 상호 연결하는 데 사용할 수 있는 네트워크 전송 허브입니다. 자세한 내용은 AWS Transit Gateway 설명서의 [전송 게이트웨이란 무엇입니까?](#)를 참조하세요.

### 트렁크 기반 워크플로

개발자가 기능 브랜치에서 로컬로 기능을 구축하고 테스트한 다음 해당 변경 사항을 기본 브랜치에 병합하는 접근 방식입니다. 이후 기본 브랜치는 개발, 프로덕션 이전 및 프로덕션 환경에 순차적으로 구축됩니다.

### 신뢰할 수 있는 액세스

사용자를 대신하여 AWS Organizations 및 해당 계정에서 조직에서 작업을 수행하도록 지정하는 서비스에 대한 권한 부여. 신뢰할 수 있는 서비스는 필요할 때 각 계정에 서비스 연결 역할을 생성하여 관리 작업을 수행합니다. 자세한 내용은 설명서의 [다른 AWS 서비스와 AWS Organizations 함께 사용](#)을 참조하세요 AWS Organizations .

### 튜닝

ML 모델의 정확도를 높이기 위해 훈련 프로세스의 측면을 여러 변경하는 것입니다. 예를 들어, 레이블링 세트를 생성하고 레이블을 추가한 다음 다양한 설정에서 이러한 단계를 여러 번 반복하여 모델을 최적화하는 방식으로 ML 모델을 훈련할 수 있습니다.

### 피자 두 판 팀

피자 두 판이면 충분한 소규모 DevOps 팀. 피자 두 판 팀 규모는 소프트웨어 개발에 있어 가능한 최상의 공동 작업 기회를 보장합니다.

## U

### 불확실성

예측 ML 모델의 신뢰성을 저해할 수 있는 부정확하거나 불완전하거나 알려지지 않은 정보를 나타내는 개념입니다. 불확실성에는 두 가지 유형이 있습니다. 인식론적 불확실성은 제한적이고 불완전한 데이터에 의해 발생하는 반면, 우연한 불확실성은 데이터에 내재된 노이즈와 무작위성에 의해 발생합니다.

### 차별화되지 않은 작업

애플리케이션을 만들고 운영하는 데 필요하지만 최종 사용자에게 직접적인 가치를 제공하거나 경쟁 우위를 제공하지 못하는 작업을 헤비 리프팅이라고도 합니다. 차별화되지 않은 작업의 예로는 조달, 유지보수, 용량 계획 등이 있습니다.

## 상위 환경

[환경](#)을 참조하세요.

## V

### 정리

스토리지를 회수하고 성능을 향상시키기 위해 증분 업데이트 후 정리 작업을 수반하는 데이터베이스 유지 관리 작업입니다.

### 버전 제어

리포지토리의 소스 코드 변경과 같은 변경 사항을 추적하는 프로세스 및 도구입니다.

### VPC 피어링

프라이빗 IP 주소를 사용하여 트래픽을 라우팅할 수 있게 하는 두 VPC 간의 연결입니다. 자세한 내용은 Amazon VPC 설명서의 [VPC 피어링이란?](#)을 참조하십시오.

### 취약성

시스템 보안을 손상시키는 소프트웨어 또는 하드웨어 결함입니다.

## W

### 웹 캐시

자주 액세스하는 최신 관련 데이터를 포함하는 버퍼 캐시입니다. 버퍼 캐시에서 데이터베이스 인스턴스를 읽을 수 있기 때문에 주 메모리나 디스크에서 읽는 것보다 빠릅니다.

### 웜 데이터

자주 액세스하지 않는 데이터입니다. 이런 종류의 데이터를 쿼리할 때는 일반적으로 적절히 느린 쿼리가 허용됩니다.

### 창 함수

현재 레코드와 어떤 식으로든 관련된 행 그룹에서 계산을 수행하는 SQL 함수입니다. 창 함수는 이동 평균을 계산하거나 현재 행의 상대적 위치를 기반으로 행 값에 액세스하는 등의 태스크를 처리하는 데 유용합니다.

## 워크로드

고객 대면 애플리케이션이나 백엔드 프로세스 같이 비즈니스 가치를 창출하는 리소스 및 코드 모음입니다.

## 워크스트림

마이그레이션 프로젝트에서 특정 작업 세트를 담당하는 직무 그룹입니다. 각 워크스트림은 독립적이지만 프로젝트의 다른 워크스트림을 지원합니다. 예를 들어, 포트폴리오 워크스트림은 애플리케이션 우선순위 지정, 웨이브 계획, 마이그레이션 메타데이터 수집을 담당합니다. 포트폴리오 워크스트림은 이러한 자산을 마이그레이션 워크스트림에 전달하고, 마이그레이션 워크스트림은 서버와 애플리케이션을 마이그레이션합니다.

## WORM

[Write Once, Read Many\(WORM\)](#)를 참조하세요.

## WQF

[AWS Workload Qualification Framework](#)를 참조하세요.

## Write Once Read Many(WORM)

데이터를 한 번 쓰고 데이터가 삭제되거나 수정되지 않도록 하는 스토리지 모델입니다. 권한 있는 사용자는 필요한 만큼 여러 번 데이터를 읽을 수 있지만 데이터를 변경할 수는 없습니다. 이 데이터 스토리지 인프라는 [변경 불가능](#)한 항목으로 간주됩니다.

## Z

### 제로데이 익스플로잇

[제로데이 취약성](#)을 악용하는 공격(일반적으로 맬웨어)입니다.

### 제로데이 취약성

프로덕션 시스템의 명백한 결함 또는 취약성입니다. 위협 행위자는 이러한 유형의 취약성을 사용하여 시스템을 공격할 수 있습니다. 개발자는 공격의 결과로 취약성을 인지하는 경우가 많습니다.

### 제로샷 프롬프팅

태스크를 수행하기 위해 [LLM](#)에 명령을 제공하지만 안내에 도움이 되는 예제(샷)는 제공하지 않습니다. LLM은 사전 훈련된 지식을 사용하여 태스크를 처리해야 합니다. 제로샷 프롬프팅의 효과는 태스크의 복잡성과 프롬프트의 품질에 따라 달라집니다. [퓨샷 프롬프팅](#)도 참조하세요.

## 좀비 애플리케이션

평균 CPU 및 메모리 사용량이 5% 미만인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하는 것이 일반적입니다.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.