



성능 문제를 방지하기 위한 Amazon RDS 및 Amazon Aurora에서 PostgreSQL  
데이터베이스 유지 관리 활동

# AWS 권장 가이드



# AWS 권장 가이드: 성능 문제를 방지하기 위한 Amazon RDS 및 Amazon Aurora에서 PostgreSQL 데이터베이스 유지 관리 활동

# Table of Contents

소개 .....	1
목표 비즈니스 성과 .....	1
다중 버전 동시성 제어(MVCC) .....	1
자동으로 테이블 백업 처리 및 분석 .....	3
Autovacuum 메모리 관련 파라미터 .....	4
Autovacuum 파라미터 조정 .....	5
클러스터 또는 인스턴스 수준 .....	5
테이블 수준 .....	5
테이블 수준에서 적극적인 autovacuum 설정 사용 .....	6
장점 및 제한 사항 .....	7
수동으로 테이블 백업 처리 및 분석 .....	8
백업 및 정리 작업을 병렬로 실행 .....	8
VACUUM FULL로 전체 테이블 다시 쓰기 .....	13
pg_repack을 사용하여 팽창 제거 .....	14
인덱스 다시 빌드 .....	16
새 인덱스 생성 .....	19
인덱스 다시 빌드 .....	20
예시 .....	20
예제: autovacuum 및 VACUUM FULL을 사용하여 공간 회수 .....	22
리소스 .....	27
문서 기록 .....	28
용어집 .....	29
# .....	29
A .....	30
B .....	33
C .....	35
D .....	38
E .....	41
F .....	44
G .....	45
H .....	46
I .....	48
L .....	50
M .....	51

---

O .....	55
P .....	58
Q .....	60
R .....	61
S .....	64
T .....	67
U .....	69
V .....	69
W .....	70
Z .....	71
.....	lxxii

# 성능 문제를 방지하기 위한 Amazon RDS 및 Amazon Aurora에서 PostgreSQL 데이터베이스 유지 관리 활동

Anuradha Chintla, Rajesh Madiwale, Srinivas Potlachervoo, Amazon Web Services(AWS)

2025년 8월([문서 기록](#))

Amazon Aurora PostgreSQL 호환 에디션 및 Amazon Relational Database Service(Amazon RDS) for PostgreSQL은 PostgreSQL 데이터베이스에 대한 완전관리형 관계형 데이터베이스입니다. 이러한 관리형 서비스를 사용하면 데이터베이스 관리자가 많은 유지 관리 및 관리 태스크를 수행하지 않아도 됩니다. 그러나 VACUUM과 같은 일부 유지 관리 태스크에는 데이터베이스 사용량에 따라 면밀한 모니터링 및 구성이 필요합니다. 이 가이드에서는 Amazon RDS 및 Aurora에서 PostgreSQL 유지 관리 활동을 설명합니다.

## 목표 비즈니스 성과

데이터베이스 성능은 비즈니스의 성공을 뒷받침하는 주요 척도입니다. Aurora PostgreSQL 호환 및 Amazon RDS for PostgreSQL 데이터베이스에서 유지 관리 활동을 수행하면 다음과 같은 이점이 있습니다.

- 최적의 쿼리 성능을 달성하도록 지원
- 향후 트랜잭션에서 재사용할 수 있도록 팽창된 공간 해제
- 트랜잭션 래퍼라운드 방지
- 옵티마이저에서 좋은 계획을 수립하도록 지원
- 적절한 인덱스 사용 보장

## 다중 버전 동시성 제어(MVCC)

PostgreSQL 데이터베이스 유지 관리에는 PostgreSQL의 메커니즘인 다중 버전 동시성 제어(MVCC)에 대한 이해가 필요합니다. 데이터베이스에서 여러 트랜잭션이 동시에 처리되면 MVCC는 원자성, 일관성, 격리성, 내구성(ACID) 트랜잭션의 두 가지 특성인 원자성과 격리성이 유지되도록 보장합니다. MVCC에서 모든 쓰기 작업은 새 버전의 데이터를 생성하고 이전 버전을 저장합니다. 리더와 라이터는 서로를 차단하지 않습니다. 트랜잭션이 데이터를 읽을 때 시스템은 트랜잭션 격리를 제공하기 위해 버전 중 하나를 선택합니다. PostgreSQL 및 일부 관계형 데이터베이스는 스냅샷 격리(SI)라는 MVCC

의 적응 기능을 사용합니다. 예를 들어 Oracle은 롤백 세그먼트를 사용하여 SI를 구현합니다. 쓰기 작업 중에 Oracle은 이전 버전의 데이터를 롤백 세그먼트에 쓰고 데이터 영역을 새 버전으로 덮어씁니다. PostgreSQL 데이터베이스는 가시성 검사 규칙을 사용하여 버전을 평가해 SI를 구현합니다. 새 데이터가 테이블 페이지에 배치되면 PostgreSQL은 이러한 규칙을 사용하여 읽기 작업에 적합한 버전의 데이터를 선택합니다.

테이블 행의 데이터를 수정하면 PostgreSQL은 MVCC를 사용하여 행의 여러 버전을 유지 관리합니다. 테이블에 대한 UPDATE 및 DELETE 작업 중에 데이터베이스는 데이터의 일관된 보기가 필요할 수 있는 실행 중인 다른 트랜잭션에서 이전 버전의 행을 유지합니다. 이러한 이전 버전을 데드 행(튜플)이라고 합니다. 데드 튜플 모음은 팽창을 생성합니다. 데이터베이스에 많은 양의 팽창이 발생하면 열악한 쿼리 계획 생성, 느린 쿼리 성능, 이전 버전을 저장하기 위한 디스크 공간 사용량 증가 등 여러 문제가 발생할 수 있습니다.

팽창을 제거하고 데이터베이스를 정상으로 유지하려면 정기적인 유지 관리가 필요합니다. 여기에는 다음 섹션에서 설명하는 이러한 활동이 포함됩니다.

- [자동으로 테이블 백업 처리 및 분석](#)
- [수동으로 테이블 백업 처리 및 분석](#)
- [pg\\_repack을 사용하여 팽창 제거](#)
- [인덱스 다시 빌드](#)

## 자동으로 테이블 백업 처리 및 분석

Autovacuum은 데드 튜플을 자동으로 백업 처리(정리)하고 스토리지를 회수하며 통계를 수집하는 대문(즉, 백그라운드에서 실행됨)입니다. 데이터베이스에서 팽창된 테이블을 확인하고 공간을 재사용하기 위해 팽창을 지웁니다. 데이터베이스 테이블과 인덱스를 모니터링하고 업데이트 또는 삭제 작업의 특정 임계치에 도달한 후 백업 작업에 추가합니다.

Autovacuum은 PostgreSQL VACUUM 및 ANALYZE 명령을 자동화하여 백업 처리를 관리합니다. VACUUM은 테이블에서 팽창을 제거하고 공간을 회수하는 반면 ANALYZE는 옵티마이저가 효율적인 계획을 생성할 수 있도록 통계를 업데이트합니다. 또한 VACUUM은 데이터베이스에서 트랜잭션 ID 랩어라운드 문제를 방지하기 위해 백업 처리 동결이라고 하는 주요 태스크를 수행합니다. 데이터베이스에서 업데이트되는 모든 행은 PostgreSQL 트랜잭션 제어 메커니즘에서 트랜잭션 ID를 수신합니다. 이러한 ID는 다른 동시 트랜잭션에 대한 행의 가시성을 제어합니다. 트랜잭션 ID는 32비트 숫자입니다. 20억 개의 ID는 항상 과거로 표시됩니다. 나머지 ID(약 22억 개)는 향후 발생할 트랜잭션에 대해 보존되며 현재 트랜잭션에서 숨겨집니다. PostgreSQL은 트랜잭션의 랩어라운드를 방지하고 새 트랜잭션이 생성될 때 오래된 기존 행이 보이지 않는 문제를 방지하기 위해 가끔 이전 행을 정리하고 동결해야 합니다. 자세한 내용은 PostgreSQL 설명서에서 [Preventing Transaction ID Wraparound Failures](#)를 참조하십시오.

Autovacuum이 권장되고 기본적으로 활성화됩니다. 해당 파라미터로 다음이 포함됩니다.

파라미터	설명	Amazon RDS의 기본값	Aurora의 기본값
autovacuum_vacuum_threshold	Autovacuum이 백업 처리하기 전에 테이블에서 수행해야 하는 튜플 업데이트 또는 삭제 작업의 최소 수.	50개 작업	50개 작업
autovacuum_analyze_threshold	Autovacuum이 테이블을 분석하기 전에 테이블에서 수행해야 하는 최소 튜플 삽입, 업데이트 또는 삭제 수.	50개 작업	50개 작업

<code>autovacuum_vacuum_scale_factor</code>	Autovacuum이 백업 처리하기 전에 테이블에서 수정해야 하는 튜플의 비율.	0.1	0.1
<code>autovacuum_analyze_scale_factor</code>	Autovacuum이 분석하기 전에 테이블에서 수정해야 하는 튜플의 비율.	0.05	0.05
<code>autovacuum_freeze_max_age</code>	트랜잭션 ID 랩어라운드 문제를 방지하기 위해 테이블이 백업 처리되기 전 동결된 ID의 최대 수명.	200,000,000개 트랜잭션	200,000,000개 트랜잭션

Autovacuum은 다음과 같이 특정 임계치 공식을 기반으로 처리할 테이블 목록을 만듭니다.

- 테이블에서 VACUUM을 실행하기 위한 임계치:

```
vacuum threshold = autovacuum_vacuum_threshold + (autovacuum_vacuum_scale_factor * Total row count of table)
```

- 테이블에서 ANALYZE을 실행하기 위한 임계치:

```
analyze threshold = autovacuum_analyze_threshold + (autovacuum_analyze_scale_factor * Total row count of table)
```

소형에서 중간 크기의 테이블인 경우 기본값으로 충분할 수 있습니다. 그러나 데이터를 자주 수정하는 대형 테이블에는 데드 튜플 수가 더 많습니다. 이 경우 autovacuum은 유지 관리를 위해 테이블을 자주 처리할 수 있으며, 큰 테이블이 완료될 때까지 다른 테이블의 유지 관리가 지연되거나 무시될 수 있습니다. 이를 방지하기 위해 다음 섹션에 설명된 autovacuum 파라미터를 조정할 수 있습니다.

## Autovacuum 메모리 관련 파라미터

### `autovacuum_max_workers`

동시에 실행할 수 있는 autovacuum 프로세스(autovacuum 런처 제외)의 최대 수를 지정합니다. 이 파라미터는 서버를 시작할 때만 설정할 수 있습니다. Autovacuum 프로세스가 큰 테이블에서 사용 중인 경우 이 파라미터는 다른 테이블에 대한 정리를 실행하는 데 도움이 됩니다.

### **maintenance\_work\_mem**

유지 관리 작업(예: VACUUM, CREATE INDEX, ALTER)에서 사용할 최대 메모리 양을 지정합니다.

Amazon RDS 및 Aurora에서는

$GREATEST(\{DBInstanceClassMemory/63963136*1024\}, 65536)$  수식을 사용하여 인스턴스 클래스를 기반으로 메모리가 할당됩니다. Autovacuum이 실행되면 계산된 값의 최대 autovacuum\_max\_workers 배를 할당할 수 있으므로 값을 너무 높게 설정하지 않도록 주의하세요. 이를 제어하기 위해 autovacuum\_work\_mem을 별도로 설정할 수 있습니다.

### **autovacuum\_work\_mem**

각 autovacuum 작업자 프로세스에서 사용할 최대 메모리를 지정합니다. 이 파라미터의 기본값은 -1이며, 이는 대신 maintenance\_work\_mem의 값을 사용해야 함을 나타냅니다.

Autovacuum 메모리 파라미터에 대한 자세한 내용은 Amazon RDS 설명서의 [Autovacuum에 메모리 할당](#)을 참조하세요.

## Autovacuum 파라미터 조정

사용자는 업데이트 및 삭제 작업에 따라 autovacuum 파라미터를 조정해야 할 수 있습니다. 다음 파라미터에 대한 설정은 테이블, 인스턴스 또는 클러스터 수준에서 설정할 수 있습니다.

### 클러스터 또는 인스턴스 수준

예를 들어 지속적 데이터 조작 언어(DML) 작업이 예상되는 बैं킹 데이터베이스를 살펴보겠습니다. 데이터베이스 상태를 유지하려면 Aurora의 클러스터 수준과 Amazon RDS의 인스턴스 수준에서 autovacuum 파라미터를 조정하고 리더에도 동일한 파라미터 그룹을 적용해야 합니다. 장애 조치가 발생하면 새 라이터에 동일한 파라미터를 적용해야 합니다.

### 테이블 수준

예를 들어 orders라는 단일 테이블에서 지속적인 DML 작업이 예상되는 음식 배달 데이터베이스의 경우 다음 명령을 사용하여 테이블 수준에서 autovacuum\_analyze\_threshold 파라미터를 조정하는 방법을 고려해야 합니다.

```
ALTER TABLE <table_name> SET (autovacuum_analyze_threshold = <threshold rows>)
```

## 테이블 수준에서 적극적인 autovacuum 설정 사용

기본 autovacuum 설정으로 인해 지속적 업데이트 및 삭제 작업이 있는 orders 테이블 예제가 배움 처리 후보가 됩니다. 이로 인해 잘못된 계획이 생성되고 쿼리가 느려집니다. 평창을 지우고 통계를 업데이트하려면 테이블 수준의 공격적인 autovacuum 설정이 필요합니다.

설정을 결정하려면 이 테이블에서 실행 중인 쿼리 시간을 추적하고 계획 변경으로 이어지는 DML 작업의 비율을 식별합니다. pg\_stat\_user\_tables 보기는 삽입, 업데이트 및 삭제 작업을 추적하는 데 도움이 됩니다.

예제:

orders 테이블의 5%가 변경될 때마다 최솟값이 잘못된 계획을 생성한다고 가정해 보겠습니다. 이 경우 다음과 같이 배움 인수 임계치를 2%로 변경해야 합니다.

```
ALTER TABLE orders SET (autovacuum_vacuum_scale_factor = 0.02)
```

### Tip

리소스 소비가 많지 않도록 적극적인 autovacuum 설정은 신중하게 선택합니다.

자세한 내용은 다음을 참조하세요.

- [Amazon RDS for PostgreSQL 환경의 autovacuum 이해](#)(AWS 블로그 게시물)
- [Automatic Vacuuming](#)(PostgreSQL 설명서)
- [Amazon RDS 및 Amazon Aurora에서 PostgreSQL 파라미터 튜닝](#)(AWS 권고 가이드)

Autovacuum이 효과적으로 작동하는지 확인하려면 데드 행, 디스크 사용량 및 마지막으로 autovacuum을 모니터링하거나 정기적으로 ANALYZE를 실행한 시간을 모니터링합니다. pg\_stat\_all\_tables 보기는 각 테이블에 대한 정보(relname)와 테이블에 있는 데드 튜플 수(n\_dead\_tup)를 제공합니다.

각 테이블, 특히 빈번하게 업데이트되는 테이블의 데드 튜플 수를 모니터링하면 autovacuum 프로세스가 데드 튜플을 주기적으로 제거하는지 확인할 수 있습니다. 이를 통해 디스크 공간을 재사용하여 성능

을 높일 수 있습니다. 다음 쿼리를 사용하여 데드 튜플 수와 테이블에서 마지막으로 autovacuum이 실행된 시간을 확인할 수 있습니다.

```
SELECT
relname AS TableName,n_live_tup AS LiveTuples,n_dead_tup AS DeadTuples,
last_autovacuum AS Autovacuum,last_autoanalyze AS Autoanalyze_FROM
pg_stat_user_tables;
```

## 장점 및 제한 사항

Autovacuum의 장점은 다음과 같습니다.

- 테이블에서 평창을 자동으로 제거합니다.
- 트랜잭션 ID 래퍼라운드를 방지합니다.
- 데이터베이스 통계를 최신 상태로 유지합니다.

제한 사항:

- 쿼리가 병렬 처리를 사용하는 경우 작업자 프로세스 수가 autovacuum에 충분하지 않을 수 있습니다.
- 피크 시간에 autovacuum이 실행되면 리소스 사용률이 증가할 수 있습니다. 이 문제를 처리하려면 파라미터를 조정해야 합니다.
- 테이블 페이지가 다른 세션에 있는 경우 autovacuum은 해당 페이지를 건너뛴 수 있습니다.
- Autovacuum은 임시 테이블에 액세스할 수 없습니다.

## 수동으로 테이블 백업 처리 및 분석

데이터베이스가 autovacuum 프로세스에 의해 백업 처리되는 경우 전체 데이터베이스에서 수동 백업을 너무 자주 실행하지 않는 것이 가장 좋습니다. 수동 백업으로 인해 불필요한 I/O 로드 또는 CPU 스파이크가 발생할 수 있으며 데드 튜플을 제거하지 못할 수도 있습니다. 라이브 튜플과 데드 튜플의 비율이 낮거나 autovacuum 사이에 간격이 긴 경우와 같이 실제로 필요한 경우에만 테이블 단위로 수동 백업을 실행합니다. 또한 최소한의 사용자 활동이 존재하는 경우 수동 백업을 실행해야 합니다.

또한 autovacuum은 테이블의 통계를 최신 상태로 유지합니다. ANALYZE 명령을 수동으로 실행하면 이러한 통계를 업데이트하는 대신 다시 빌드합니다. 일반 autovacuum 프로세스에 의해 이미 업데이트된 통계를 다시 빌드하면 시스템 리소스 사용률이 발생할 수 있습니다.

다음 시나리오에서는 [VACUUM](#) 및 [ANALYZE](#) 명령을 수동으로 실행하는 것이 좋습니다.

- 사용량이 많은 테이블의 피크 시간이 낮을 경우(autovacuum 처리로 충분하지 않을 수 있음).
- 데이터를 대상 테이블에 대량 로드한 직후. 이 경우 ANALYZE를 수동으로 실행하면 통계가 완전히 다시 빌드되므로 자동 백업이 시작될 때까지 기다리는 것보다 더 좋은 옵션입니다.
- 임시 테이블을 백업하려는 경우(autovacuum은 여기에 액세스할 수 없음).

동시 데이터베이스 활동에서 VACUUM 및 ANALYZE 명령을 실행할 때 I/O 영향을 줄이기 위해 vacuum\_cost\_delay 파라미터를 사용할 수 있습니다. 대부분의 경우 VACUUM 및 ANALYZE와 같은 유지 관리 명령은 빠르게 완료할 필요가 없습니다. 그러나 이러한 명령이 다른 데이터베이스 작업을 수행하는 시스템의 기능을 방해해서는 안 됩니다. 이를 방지하기 위해 vacuum\_cost\_delay 파라미터를 사용하여 비용 기반 백업 지연을 활성화할 수 있습니다. 이 파라미터는 수동으로 실행된 VACUUM 명령에 대해 기본적으로 비활성화되어 있습니다. 활성화하려면 0이 아닌 값으로 설정합니다.

## 백업 및 정리 작업을 병렬로 실행

VACUUM 명령 [PARALLEL](#) 옵션은 인덱스 백업 처리 및 인덱스 정리 단계에 병렬 작업자를 사용하며, 기본적으로 비활성화되어 있습니다. 병렬 작업자 수(병렬화 수준)는 테이블의 인덱스 수에 따라 결정되며 사용자가 지정할 수 있습니다. 정수 인수 없이 병렬 VACUUM 작업을 실행하는 경우 병렬화 수준은 테이블의 인덱스 수를 기준으로 계산됩니다.

다음 파라미터는 Amazon RDS for PostgreSQL 및 Aurora PostgreSQL 호환 버전에서 병렬 백업 처리를 구성하는 데 도움이 됩니다.

- [max\\_worker\\_processes](#)는 최대 동시 작업자 프로세스 수를 설정합니다.

- [min\\_parallel\\_index\\_scan\\_size](#)는 병렬 스캔을 고려하기 위해 스캔해야 하는 인덱스 데이터의 최소 양을 설정합니다.
- [max\\_parallel\\_maintenance\\_workers](#)는 단일 유틸리티 명령으로 시작할 수 있는 최대 병렬 작업자 수를 설정합니다.

**Note**

PARALLEL 옵션은 백업 용도로만 사용됩니다. ANALYZE 명령에는 영향을 주지 않습니다.

다음 예제에서는 데이터베이스에서 수동 VACUUM 및 ANALYZE를 사용할 때 데이터베이스 동작을 보여줍니다.

다음은 autovacuum이 비활성화된 테이블 샘플입니다(설명용으로 제공됨, autovacuum 비활성화는 권장되지 않음).

```
create table t1 ( a int, b int, c int );
alter table t1 set (autovacuum_enabled=false);
```

```
apgl=> \d+ t1
Table "public.t1"
Column | Type | Collation | Nullable | Default | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----+-----+-----
+-----+
a | integer | | | plain | |
b | integer | | | plain | |
c | integer | | | plain | |
Access method: heap
Options: autovacuum_enabled=false
```

테이블 t1에 1백만 개의 행을 추가합니다.

```
apgl=> select count(*) from t1;
count
1000000
(1 row)
```

테이블 t1의 통계:

```
select * from pg_stat_all_tables where relname='t1';
-[ RECORD 1 ]-----+-----
relid          | 914744
schemaname     | public
relname        | t1
seq_scan       | 0
seq_tup_read   | 0
idx_scan       |
idx_tup_fetch  |
n_tup_ins      | 1000000
n_tup_upd      | 0
n_tup_del      | 0
n_tup_hot_upd  | 0
n_live_tup     | 1000000
n_dead_tup     | 0
n_mod_since_analyze | 1000000
last_vacuum    |
last_autovacuum |
last_analyze   |
last_autoanalyze |
vacuum_count   | 0
autovacuum_count | 0
analyze_count  | 0
autoanalyze_count | 0
```

인덱스 추가:

```
create index i2 on t1 (b,a);
```

EXPLAIN 명령을 실행합니다(계획 1).

```
Bitmap Heap Scan on t1 (cost=10521.17..14072.67 rows=5000 width=4)
Recheck Cond: (a = 5)
# Bitmap Index Scan on i2 (cost=0.00..10519.92 rows=5000 width=0)
Index Cond: (a = 5)
(4 rows)
```

EXPLAIN ANALYZE 명령을 실행합니다(계획 2).

```
explain (analyze,buffers,costs off) select a from t1 where b = 5;
QUERY PLAN
```

```

Bitmap Heap Scan on t1 (actual time=0.023..0.024 rows=1 loops=1)
Recheck Cond: (b = 5)
Heap Blocks: exact=1
Buffers: shared hit=4
# Bitmap Index Scan on i2 (actual time=0.016..0.016 rows=1 loops=1)
Index Cond: (b = 5)
Buffers: shared hit=3
Planning Time: 0.054 ms
Execution Time: 0.076 ms
(9 rows)

```

테이블에서 autovacuum이 비활성화되고 ANALYZE 명령이 수동으로 수행되지 않았으므로 EXPLAIN 및 EXPLAIN ANALYZE 명령은 서로 다른 계획을 표시합니다. 이제 테이블의 값을 업데이트하고 EXPLAIN ANALYZE 계획을 다시 생성해 보겠습니다.

```

update t1 set a=8 where b=5;
explain (analyze,buffers, costs off) select a from t1 where b = 5;

```

이제 EXPLAIN ANALYZE 명령(계획 3)에서 다음을 표시합니다.

```

apgl=> explain (analyze,buffers, costs off) select a from t1 where b = 5;
QUERY PLAN
Bitmap Heap Scan on t1 (actual time=0.075..0.076 rows=1 loops=1)
Recheck Cond: (b = 5)
Heap Blocks: exact=1
Buffers: shared hit=5
# Bitmap Index Scan on i2 (actual time=0.017..0.017 rows=2 loops=1)
Index Cond: (b = 5)
Buffers: shared hit=3
Planning Time: 0.053 ms
Execution Time: 0.125 ms

```

계획 2와 계획 3 간의 비용을 비교하면 아직 통계를 수집하지 않았기 때문에 계획 및 실행 시간에서 차이가 표시됩니다.

이제 테이블에서 수동 ANALYZE를 실행한 다음 통계를 확인하고 계획을 다시 생성해 보겠습니다.

```

apgl=> analyze t1
apgl# ;
ANALYZE

```

```
Time: 212.223 ms
```

```
apgl=> select * from pg_stat_all_tables where relname='t1';
```

```
-[ RECORD 1 ]-----+-----
```

relid	914744
schemaname	public
relname	t1
seq_scan	3
seq_tup_read	1000000
idx_scan	3
idx_tup_fetch	3
n_tup_ins	1000000
n_tup_upd	1
n_tup_del	0
n_tup_hot_upd	0
n_live_tup	1000000
n_dead_tup	1
n_mod_since_analyze	0
last_vacuum	
last_autovacuum	
last_analyze	2023-04-15 11:39:02.075089+00
last_autoanalyze	
vacuum_count	0
autovacuum_count	0
analyze_count	1
autoanalyze_count	0

```
Time: 148.347 ms
```

EXPLAIN ANALYZE 명령을 실행합니다(계획 4).

```
apgl=> explain (analyze,buffers, costs off) select a from t1 where b = 5;
```

```
QUERY PLAN
```

```
Index Only Scan using i2 on t1 (actual time=0.022..0.023 rows=1 loops=1)
```

```
Index Cond: (b = 5)
```

```
Heap Fetches: 1
```

```
Buffers: shared hit=4
```

```
Planning Time: 0.056 ms
```

```
Execution Time: 0.068 ms
```

```
(6 rows)
```

```
Time: 138.462 ms
```

테이블을 수동으로 분석하고 통계를 수집한 후 모든 계획 결과를 비교하면 옵티마이저의 계획 4가 다른 계획보다 더 효과적이고 쿼리 실행 시간도 단축된다는 점을 알 수 있습니다. 이 예제에서는 데이터베이스에서 유지 관리 활동을 실행하는 것이 얼마나 중요한지 보여줍니다.

## VACUUM FULL로 전체 테이블 다시 쓰기

FULL 파라미터를 사용하여 VACUUM 명령을 실행하면 테이블의 전체 콘텐츠를 추가 공간 없이 새 디스크 파일에 다시 쓰고 미사용 공간을 운영 체제에 반환합니다. 이 작업은 속도가 훨씬 느리며 각 테이블에 ACCESS EXCLUSIVE 잠금이 필요합니다. 또한 테이블의 새 사본을 쓰고 작업이 완료될 때까지 이전 복사본을 릴리스하지 않기 때문에 추가 디스크 공간이 필요합니다.

VACUUM FULL은 다음과 같은 사례에 유용할 수 있습니다.

- 테이블에서 상당한 양의 공간을 회수하려는 경우.
- 프라이머리 키가 아닌 키 테이블에서 팽창 공간을 회수하려는 경우.

데이터베이스가 가동 중지 시간을 허용할 수 있는 경우 프라이머리 키가 아닌 키 테이블이 있는 경우 VACUUM FULL을 사용하는 것이 좋습니다.

VACUUM FULL에는 다른 작업보다 더 많은 잠금이 필요하기 때문에 중요한 데이터베이스에서 실행하는 데 더 많은 비용이 듭니다. 이 메서드를 교체하기 위해 [다음 섹션](#)에 설명된 pg\_repack 확장을 사용할 수 있습니다. 이 옵션은 VACUUM FULL과 유사하지만 최소한의 잠금이 필요하며 Amazon RDS for PostgreSQL 및 Aurora PostgreSQL 호환 모두에서 지원됩니다.

## pg\_repack을 사용하여 팽창 제거

pg\_repack 확장을 사용하여 데이터베이스 잠금을 최소화하면서 테이블 및 인덱스 팽창을 제거할 수 있습니다. 데이터베이스 인스턴스에서 이 확장을 생성하고 Amazon Elastic Compute Cloud(Amazon EC2) 또는 데이터베이스에 연결할 수 있는 컴퓨터에서 pg\_repack 클라이언트(클라이언트 버전이 확장 버전과 일치하는 경우)를 실행할 수 있습니다.

VACUUM FULL과 달리 pg\_repack에는 가동 중지 시간이나 유지 관리 기간이 필요하지 않으며 다른 세션을 차단하지 않습니다.

pg\_repack은 VACUUM FULL, CLUSTER 또는 REINDEX가 작동하지 않는 상황에서 유용합니다. 팽창된 테이블의 데이터가 포함된 새 테이블을 만들고 원본 테이블의 변경 사항을 추적한 다음 원본 테이블을 새 테이블로 바꿉니다. 새 테이블을 빌드하는 동안 읽기 또는 쓰기 작업을 위해 원본 테이블을 잠그지 않습니다.

전체 테이블 또는 인덱스에 pg\_repack을 사용할 수 있습니다. 태스크 목록을 보려면 [pg\\_repack 설명서](#)를 참조하세요.

제한:

- pg\_repack을 실행하려면 테이블에 프라이머리 키 또는 고유 인덱스가 있어야 합니다.
- pg\_repack 은 임시 테이블에서 작동하지 않습니다.
- pg\_repack은 글로벌 인덱스가 있는 테이블에서 작동하지 않습니다.
- pg\_repack이 진행 중이면 테이블에서 DDL 작업을 수행할 수 없습니다.

다음 표에서는 pg\_repack 및 VACUUM FULL 사이의 차이를 설명합니다.

VACUUM FULL	pg_repack
기본 제공 명령	Amazon EC2 또는 로컬 컴퓨터에서 실행하는 확장
테이블에서 작업하는 동안 ACCESS EXCLUSIVE 잠금 필요	짧은 시간 동안만 ACCESS EXCLUSIVE 잠금 필요
모든 테이블에서 작동	프라이머리 키 및 고유 키만 있는 테이블에서 작동

테이블 및 인덱스에서 소비하는 스토리지의 두 배 필요

테이블 및 인덱스에서 소비하는 스토리지의 두 배 필요

테이블에서 `pg_repack`을 실행하려면 다음 명령을 사용합니다.

```
pg_repack -h <host> -d <dbname> --table <tablename> -k
```

인덱스에서 `pg_repack`을 실행하려면 다음 명령을 사용합니다.

```
pg_repack -h <host> -d <dbname> --index <index name>
```

자세한 내용은 AWS 블로그 게시물 [Remove bloat from Amazon Aurora and RDS for PostgreSQL with pg\\_repack](#)을 참조하세요.

### 주의 사항

`pg_repack`의 `error-on-invalid-index` 오류는 일반적으로 테이블에서 하나 이상의 인덱스가 손상되었거나 유효하지 않음을 의미합니다. `pg_repack`은 재압축 프로세스 중 데이터 일관성을 위해 인덱스에 의존하기 때문에 유효하지 않은 인덱스가 있는 테이블에서 안전하게 작동할 수 없습니다.

이 오류는 다음과 같은 경우에 발생합니다.

- 인덱스가 유효하지 않은 항목으로 표시됩니다(예: 실패한 `CREATE INDEX CONCURRENTLY` 문이 원인임).
- 인덱스가 손상되었습니다(하드웨어 문제 또는 갑작스러운 종료의 원인이 될 수 있음).

다음 쿼리를 사용하여 유효하지 않은 인덱스를 식별하고 해당 인덱스를 찾으면 먼저 삭제합니다.

```
SELECT indexrelid::regclass, indisvalid FROM pg_index WHERE indrelid =
'orders'::regclass AND NOT indisvalid; Drop the invalid index: DROP INDEX
index_name;
```

## 인덱스 다시 빌드

PostgreSQL [REINDEX](#) 명령은 인덱스 테이블에 저장된 데이터를 사용하고 인덱스의 이전 복사본을 대체하여 인덱스를 다시 빌드합니다. 다음과 같은 시나리오에서는 REINDEX를 사용하는 것이 좋습니다.

- 인덱스가 손상되어 더 이상 유효한 데이터가 포함되지 않는 경우. 소프트웨어 또는 하드웨어 장애로 인해 이와 같은 경우가 발생할 수 있습니다.
- 이전에 인덱스를 사용한 쿼리는 해당 인덱스 사용을 중지하는 경우.
- 많은 빈 페이지 또는 거의 빈 페이지로 인덱스가 팽창되는 경우. 팽창률(bloat\_pct)이 20보다 큰 경우 REINDEX를 실행해야 합니다.

다음 쿼리는 bloat\_pct를 찾는 데 도움이 됩니다.

```
SELECT current_database(), nspname AS schemaname, tblname, idxname,
bs*(relpages)::bigint AS real_size,
bs*(relpages-est_pages)::bigint AS extra_size,
100 * (relpages-est_pages)::float / relpages AS extra_pct,
fillfactor,
CASE WHEN relpages > est_pages_ff
THEN bs*(relpages-est_pages_ff)
ELSE 0
END AS bloat_size,
100 * (relpages-est_pages_ff)::float / relpages AS bloat_pct,
is_na
-- , 100-(pst).avg_leaf_density AS pst_avg_bloat, est_pages, index_tuple_hdr_bm,
maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, reltuples, relpages -- (DEBUG
INFO)
FROM (
SELECT coalesce(1 +
ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
-- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
) AS est_pages,
coalesce(1 +
ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
) AS est_pages_ff,
bs, nspname, tblname, idxname, relpages, fillfactor, is_na
-- , pgstatindex(idxoid) AS pst, index_tuple_hdr_bm, maxalign, pagehdr,
nulldatawidth, nulldatahdrwidth, reltuples -- (DEBUG INFO)
FROM (
```

```

SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, idxoid,
fillfactor,
  ( index_tuple_hdr_bm +
    maxalign - CASE -- Add padding to the index tuple header to align on
MAXALIGN
      WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
      ELSE index_tuple_hdr_bm%maxalign
    END
  + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
MAXALIGN
      WHEN nulldatawidth = 0 THEN 0
      WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
      ELSE nulldatawidth::integer%maxalign
    END
  )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
-- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
FROM (
  SELECT n.nspname, i.tblname, i.idxname, i.reltuples, i.relpages,
    i.idxoid, i.fillfactor, current_setting('block_size')::numeric AS bs,
    CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
      WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|
amd64' THEN 8
      ELSE 4
    END AS maxalign,
    /* per page header, fixed size: 20 for 7.X, 24 for others */
    24 AS pagehdr,
    /* per page btree opaque data */
    16 AS pageopqdata,
    /* per tuple header: add IndexAttributeBitMapData if some cols are null-
able */
    CASE WHEN max(coalesce(s.null_frac,0)) = 0
      THEN 8 -- IndexTupleData size
      ELSE 8 + (( 32 + 8 - 1 ) / 8) -- IndexTupleData size +
IndexAttributeBitMapData size ( max num filed per index + 8 - 1 /8)
    END AS index_tuple_hdr_bm,
    /* data len: we remove null values save space using it fractionnal part
from stats */
    sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
    max( CASE WHEN i.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0
END ) > 0 AS is_na
  FROM (

```

```

SELECT ct.relname AS tblname, ct.relnamespace, ic.idxname, ic.attpos,
ic.indkey, ic.indkey[ic.attpos], ic.reltuples, ic.relpages, ic.tbloid, ic.idxoid,
ic.fillfactor,
    coalesce(a1.attnum, a2.attnum) AS attnum, coalesce(a1.attname,
a2.attname) AS attname, coalesce(a1.atttypid, a2.atttypid) AS atttypid,
    CASE WHEN a1.attnum IS NULL
    THEN ic.idxname
    ELSE ct.relname
    END AS attrelname
FROM (
    SELECT idxname, reltuples, relpages, tbloid, idxoid, fillfactor,
indkey,
        pg_catalog.generate_series(1,indnatts) AS attpos
    FROM (
        SELECT ci.relname AS idxname, ci.reltuples, ci.relpages,
i.indrelid AS tbloid,
            i.indexrelid AS idxoid,
            coalesce(substring(
                array_to_string(ci.reloptions, ' ')
                from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor,
            i.indnatts,
            pg_catalog.string_to_array(pg_catalog.textin(
                pg_catalog.int2vectorout(i.indkey)), ' ')::int[] AS indkey
        FROM pg_catalog.pg_index i
        JOIN pg_catalog.pg_class ci ON ci.oid = i.indexrelid
        WHERE ci.relam=(SELECT oid FROM pg_am WHERE amname = 'btree')
        AND ci.relpages > 0
    ) AS idx_data
    ) AS ic
JOIN pg_catalog.pg_class ct ON ct.oid = ic.tbloid
LEFT JOIN pg_catalog.pg_attribute a1 ON
    ic.indkey[ic.attpos] <> 0
    AND a1.attrelid = ic.tbloid
    AND a1.attnum = ic.indkey[ic.attpos]
LEFT JOIN pg_catalog.pg_attribute a2 ON
    ic.indkey[ic.attpos] = 0
    AND a2.attrelid = ic.idxoid
    AND a2.attnum = ic.attpos
) i
JOIN pg_catalog.pg_namespace n ON n.oid = i.relnamespace
JOIN pg_catalog.pg_stats s ON s.schemaname = n.nspname
    AND s.tablename = i.attrelname
    AND s.attname = i.attname

```

```

GROUP BY 1,2,3,4,5,6,7,8,9,10,11
) AS rows_data_stats
) AS rows_hdr_pdg_stats
) AS relation_stats
ORDER BY nspname, tblname, idxname;

```

완전히 비어 있는 인덱스 페이지는 재사용을 위해 회수됩니다. 그러나 페이지의 인덱스 키가 삭제되었지만 스페이스가 할당된 상태로 남아 있는 경우 주기적으로 재인덱싱하는 것이 좋습니다.

인덱스를 다시 생성하면 쿼리 성능이 향상됩니다. 다음 표에 설명된 대로 세 가지 방법으로 인덱스를 다시 생성할 수 있습니다.

메서드	설명	제한 사항
CONCURRENTLY 옵션과 함께 CREATE INDEX 및 DROP INDEX	새 인덱스를 빌드하고 이전 인덱스를 제거합니다. 옵티마이저는 이전 인덱스 대신 새로 생성된 인덱스를 사용하여 계획을 생성합니다. 피크가 낮은 기간에 이전 인덱스를 삭제할 수 있습니다.	인덱스 생성은 모든 수신 변경 사항을 추적해야 하므로 CONCURRENTLY 옵션을 사용할 때 더 많은 시간이 걸립니다. 변경 사항이 동결되면 프로세스가 완료로 표시됩니다.
CONCURRENTLY 옵션과 함께 REINDEX	다시 빌드 프로세스 중에 쓰기 작업을 잠급니다. PostgreSQL 버전 12 이상 버전은 이러한 잠금을 방지하는 CONCURRENTLY 옵션을 제공합니다.	CONCURRENTLY 를 사용하면 인덱스를 다시 빌드하는 데 시간이 더 오래 걸립니다.
pg_repack 확장	테이블에서 팡창을 정리하고 인덱스를 다시 빌드합니다.	이러한 확장은 EC2 인스턴스 또는 데이터베이스에 연결된 로컬 컴퓨터에서 실행해야 합니다.

## 새 인덱스 생성

DROP INDEX 및 CREATE INDEX 명령을 함께 사용하면 인덱스가 다시 빌드됩니다.

```
DROP INDEX <index_name>
```

```
CREATE INDEX <index_name> ON TABLE <table_name> (<column1>[,<column2>])
```

이 접근 방식의 단점은 테이블에서 배타적 잠금이 발생하는 것으로, 이 활동 중 성능에 영향을 미칩니다. DROP INDEX 명령으로 테이블에서 읽기 및 쓰기 작업을 모두 차단하는 배타적 잠금을 획득합니다. CREATE INDEX 명령은 테이블에서 쓰기 작업을 차단합니다. 읽기 작업은 허용되지만 인덱스 생성 중에는 비용이 많이 듭니다.

## 인덱스 다시 빌드

REINDEX 명령은 일관된 데이터베이스 성능을 유지하는 데 도움이 됩니다. 테이블에서 많은 DML 작업을 수행하면 테이블과 인덱스가 모두 팽창합니다. 인덱스는 쿼리 성능을 개선하기 위해 테이블 조회 속도를 높이는 데 사용됩니다. 인덱스 팽창은 조회 및 쿼리 성능에 영향을 미칩니다. 따라서 쿼리 성능의 일관성을 유지하기 위해 DML 작업이 많은 테이블에서 재인덱싱을 수행하는 것이 좋습니다.

이 REINDEX 명령은 기본 테이블에서 쓰기 작업을 잠가 인덱스를 처음부터 다시 빌드하지만 테이블에서 읽기 작업을 허용합니다. 그러나 인덱스에서 읽기 작업이 차단됩니다. 해당 인덱스를 사용하는 쿼리는 차단되지만 다른 쿼리는 차단되지 않습니다.

PostgreSQL 버전 12에는 인덱스를 처음부터 다시 빌드하지만 테이블 또는 인덱스를 사용하는 쿼리에서 쓰기 또는 읽기 작업을 잠그지 않는 새로운 선택적 파라미터, CONCURRENTLY가 도입되었습니다. 그러나 이 옵션을 사용하면 프로세스를 완료하는 데 시간이 더 오래 걸립니다.

## 예시

### 인덱스 생성 및 삭제

CONCURRENTLY 옵션을 사용하여 새 인덱스를 생성합니다.

```
create index CONCURRENTLY on table(columns) ;
```

CONCURRENTLY 옵션을 사용하여 이전 인덱스를 삭제합니다.

```
drop index CONCURRENTLY <index name> ;
```

### 인덱스 다시 빌드

단일 인덱스를 다시 빌드하는 방법:

```
reindex index <index name> ;
```

테이블에서 모든 인덱스를 다시 빌드하는 방법:

```
reindex table <table name> ;
```

스키마에서 모든 인덱스를 다시 빌드하는 방법:

```
reindex schema <schema name> ;
```

동시에 인덱스 다시 빌드

단일 인덱스를 다시 빌드하는 방법:

```
reindex index CONCURRENTLY <indexname> ;
```

테이블에서 모든 인덱스를 다시 빌드하는 방법:

```
reindex table CONCURRENTLY <tablename> ;
```

스키마에서 모든 인덱스를 다시 빌드하는 방법:

```
reindex schema CONCURRENTLY <schemaname> ;
```

인덱스만 다시 빌드 또는 재배포

단일 인덱스를 다시 빌드하는 방법:

```
pg_repack -h <hostname> -d <dbname> -i <indexname> -k
```

모든 인덱스를 다시 빌드하는 방법:

```
pg_repack -h <hostname> -d <dbname> -x <indexname> -t <tablename> -k
```

## 예제: autovacuum 및 VACUUM FULL을 사용하여 공간 회수

예를 들어 행이 500,000개인 emp 테이블을 생성한 다음 행을 새 값으로 업데이트하겠습니다.

Autovacuum이 활성화되어 있으므로 이 테이블에서 VACUUM 및 ANALYZE 명령을 모두 실행하여 팽창을 제거하고 공간을 회수합니다. 회수된 공간은 재사용할 수 있지만 운영 체제로 반환되지 않습니다.

다음 쿼리는 테이블의 팽창을 결정합니다.

```
-- WARNING: When run with a non-superuser role, the query inspects only indexes on
tables you are granted to read.
-- WARNING: Rows with is_na = 't' are known to have bad statistics ("name" type is not
supported).
-- This query is compatible with PostgreSQL 8.2 and later.
SELECT current_database(), nspname AS schemaname, tblname, idxname,
bs*(relpages)::bigint AS real_size,
bs*(relpages-est_pages)::bigint AS extra_size,
100 * (relpages-est_pages)::float / relpages AS extra_pct,
fillfactor,
CASE WHEN relpages > est_pages_ff
THEN bs*(relpages-est_pages_ff)
ELSE 0
END AS bloat_size,
100 * (relpages-est_pages_ff)::float / relpages AS bloat_pct,
is_na
-- , 100-(pst).avg_leaf_density AS pst_avg_bloat, est_pages, index_tuple_hdr_bm,
maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, reltuples, relpages -- (DEBUG
INFO)
FROM (
SELECT coalesce(1 +
ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
-- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
) AS est_pages,
coalesce(1 +
ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
) AS est_pages_ff,
bs, nspname, tblname, idxname, relpages, fillfactor, is_na
-- , pgstatindex(idxoid) AS pst, index_tuple_hdr_bm, maxalign, pagehdr,
nulldatawidth, nulldatahdrwidth, reltuples -- (DEBUG INFO)
FROM (
SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, idxoid,
fillfactor,
```

```

        ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on
MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
MAXALIGN
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
            ELSE nulldatawidth::integer%maxalign
          END
        )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
        -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
FROM (
  SELECT n.nspname, i.tblname, i.idxname, i.reltuples, i.relpages,
         i.idxoid, i.fillfactor, current_setting('block_size')::numeric AS bs,
         CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
           WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|
amd64' THEN 8
           ELSE 4
         END AS maxalign,
         /* per page header, fixed size: 20 for 7.X, 24 for others */
         24 AS pagehdr,
         /* per page btree opaque data */
         16 AS pageopqdata,
         /* per tuple header: add IndexAttributeBitMapData if some cols are null-
able */
         CASE WHEN max(coalesce(s.null_frac,0)) = 0
           THEN 8 -- IndexTupleData size
           ELSE 8 + (( 32 + 8 - 1 ) / 8) -- IndexTupleData size +
IndexAttributeBitMapData size ( max num filed per index + 8 - 1 /8)
         END AS index_tuple_hdr_bm,
         /* data len: we remove null values save space using it fractionnal part
from stats */
         sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
         max( CASE WHEN i.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0
END ) > 0 AS is_na
  FROM (
    SELECT ct.relname AS tblname, ct.relnamespace, ic.idxname, ic.attpos,
           ic.indkey, ic.indkey[ic.attpos], ic.reltuples, ic.relpages, ic.tbloid, ic.idxoid,
           ic.fillfactor,

```

```

        coalesce(a1.attnum, a2.attnum) AS attnum, coalesce(a1.attname,
a2.attname) AS attname, coalesce(a1.atttypid, a2.atttypid) AS atttypid,
        CASE WHEN a1.attnum IS NULL
        THEN ic.idxname
        ELSE ct.relname
        END AS attrelname
    FROM (
        SELECT idxname, reltuples, relpages, tbloid, idxoid, fillfactor,
i.indkey,
        pg_catalog.generate_series(1,indnatts) AS attpos
    FROM (
        SELECT ci.relname AS idxname, ci.reltuples, ci.relpages,
i.indrelid AS tbloid,
        i.indexrelid AS idxoid,
        coalesce(substring(
            array_to_string(ci.reloptions, ' ')
            from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor,
        i.indnatts,
        pg_catalog.string_to_array(pg_catalog.textin(
            pg_catalog.int2vectorout(i.indkey)), ' ')::int[] AS indkey
    FROM pg_catalog.pg_index i
    JOIN pg_catalog.pg_class ci ON ci.oid = i.indexrelid
    WHERE ci.relam=(SELECT oid FROM pg_am WHERE amname = 'btree')
    AND ci.relpages > 0
    ) AS idx_data
    ) AS ic
    JOIN pg_catalog.pg_class ct ON ct.oid = ic.tbloid
    LEFT JOIN pg_catalog.pg_attribute a1 ON
        ic.indkey[ic.attpos] <> 0
        AND a1.attrelid = ic.tbloid
        AND a1.attnum = ic.indkey[ic.attpos]
    LEFT JOIN pg_catalog.pg_attribute a2 ON
        ic.indkey[ic.attpos] = 0
        AND a2.attrelid = ic.idxoid
        AND a2.attnum = ic.attpos
    ) i
    JOIN pg_catalog.pg_namespace n ON n.oid = i.relnamespace
    JOIN pg_catalog.pg_stats s ON s.schemaname = n.nspname
        AND s.tablename = i.attrelname
        AND s.attname = i.attname

    GROUP BY 1,2,3,4,5,6,7,8,9,10,11
    ) AS rows_data_stats
) AS rows_hdr_pdg_stats

```

```
) AS relation_stats
ORDER BY nspname, tblname, idxname;
```

쿼리 결과에서 테이블의 팽창이 약 51%임을 보여줍니다.

```
current_database | schemaname | tblname | real_size | extra_size | extra_pct |
fillfactor | bloat_size | bloat_pct | is_na
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
apgl | public | emp | 60383232 | 30744576 | 50.91575091575091 | 100 | 30744576 |
50.91575091575091 | f
```

다음은 pg\_stat\_all\_tables 보기의 통계입니다.

```
relid          | 914748
schemaname     | public
relname        | emp
seq_scan       | 5
seq_tup_read   | 1500000
idx_scan       | 0
idx_tup_fetch  | 0
n_tup_ins      | 600000
n_tup_upd      | 500000
n_tup_del      | 0
n_tup_hot_upd  | 0
n_live_tup     | 500000
n_dead_tup     | 0
n_mod_since_analyze | 0
last_vacuum    |
last_autovacuum | 2023-04-15 11:59:54.957449+00
last_analyze   |
last_autoanalyze | 2023-04-15 11:59:55.016352+00
vacuum_count   | 0
autovacuum_count | 2
analyze_count  | 0
autoanalyze_count | 3
```

Autovacuum은 실행 후 last\_autovacuum 및 last\_autoanalyze 열을 업데이트했습니다.

이제 빈 공간도 팽창한 것으로 간주되므로 테이블에 일부 행을 삽입하고 extra\_size(bloat\_size)를 확인합니다.



## 리소스

- [Amazon RDS for PostgreSQL 환경의 autovacuum 이해](#)(AWS 블로그 게시물)
- [Automatic vacuuming](#)(PostgreSQL 설명서)
- [Autovacuum에 메모리 할당](#)(Amazon RDS 설명서)
- [트랜잭션 ID 래퍼라운드 장애 방지](#)(PostgreSQL 설명서)
- [Remove bloat from Amazon Aurora and RDS for PostgreSQL with pg\\_repack](#)(AWS 블로그 게시물)

## 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="#">업데이트</a>	오류를 수정하고 <a href="#">테이블 자동 정리 및 분석</a> 과 <a href="#">pg_repack</a> 으로 <a href="#">팽창 제거</a> 섹션에 정보를 추가했습니다.	2025년 8월 22일
<a href="#">reindex 구문 수정됨</a>	<a href="#">동시에 인덱스 다시 빌드</a> 에 관한 섹션에서 reindex 예제를 수정했습니다.	2025년 6월 30일
<a href="#">최초 게시</a>	—	2023년 12월 22일

# 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 슝) - 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.

### AI Ops

[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 전환 AWS 하기 위한 효율적이고 효과적인 계획을 개발하는 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 책인가요, 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 블로그에 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 수정 작업 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 채택하

면 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 서비스](#)를 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 마이그레이션 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 적

용합니다. [퓨샷 프롬프팅](#)은 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 게이트웨이라고도 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 제공합니다.

### 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

### laC

[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 VPCs. [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 작동하며 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???](#) 통신을 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 통합을 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 마이크로 서비스를 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 Framework에서는 [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.

이

[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 인터페이스

이스를 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 있는의 엔터티입니다. 이 엔터티는 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 마이크로서비스를 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 결

정권자 (A), 업무 수행 조연자 (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 대한 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 업데이트 하는 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 키 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 작업을 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 SI

조직 내 관리형 채널 외부에서 구축되거나 사용되는 승인되지 않은 [SI](#) 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 사용을 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

### 웜 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.