RDS아마존과 아마존 Aurora에서 Postgre SQL 파라미터 조정

AWS 규범적 지침

AWS 규범적 지침: RDS아마존과 아마존 Aurora에서 Postgre SQL 파라미터 조정

Table of Contents

아마존 RDS와 아마존 Aurora에서 PostgreSQL 파라미터 조정

수마나 야나만드라, 라무 자기니, 로히트 카푸어, Amazon Web Services ()AWS

2024년 2월 (문서 기록)

Amazon Aurora PostgreSQL 호환 에디션과 PostgreSQL용 아마존 관계형 데이터베이스 서비스 (Amazon RDS) 는 모든 기능을 제공하는 정교한 오픈 소스 관계형 데이터베이스 서비스입니다. 이러한 서비스를 사용하여 다양한 플랫폼 및 애플리케이션에서 PostgreSQL 데이터베이스를 설정할 수 있습니다.

Aurora와 Amazon RDS는 PostgreSQL 데이터베이스를 관리하고 운영할 수 있는 간소화된 방법을 제공합니다. 데이터베이스 인프라를 관리하고 애플리케이션 개발에 집중하는 동안 고가용성, 내구성 및 확장성을 제공하도록 설계되었습니다. 그러나 이러한 서비스의 기본 구성이 모든 워크로드에 적합하지는 않을 수 있습니다. 기본적으로 이러한 서비스는 취약성을 유발하지 않고 가능한 가장 적은 리소스로 어디서나 실행되도록 구성됩니다. 매개 변수를 조정하면 성능을 높이고, 가동 중지 시간을 줄이고, 전반적인 데이터베이스 효율성을 개선하는 데 도움이 될 수 있습니다. 특정 워크로드에 맞게 파라미터를 최적화하면 Amazon RDS와 Aurora가 제공하는 기능을 최대한 활용하고 이점을 극대화할 수 있습니다.

예를 들어, PostgreSQL용 Aurora와 Amazon RDS를 최적화하고 해당 파라미터를 구성하여 성능을 개선할 수 있습니다. 데이터베이스 쿼리를 생성할 때는 성능도 고려해야 합니다. 데이터베이스 설정을 최적화하더라도 쿼리에서 전체 테이블 스캔을 수행하거나, 인덱스를 활용하거나, 비용이 많이 드는 조인 또는 집계 작업을 실행하는 경우 시스템 성능이 저하될 수 있습니다.

이 가이드는 PostgreSQL 데이터베이스의 메모리, 자동 진공, 로깅 및 논리적 복제 매개 변수를 조정하려는 데이터베이스 개발자, 데이터베이스 엔지니어 및 관리자를 위한 것입니다. 또한 이 가이드에서는 PostgreSQL용 Amazon RDS 및 Aurora PostgreSQL과 호환되는 특정 파라미터도 다룹니다. 이러한 파라미터를 조정하면 데이터베이스 성능을 최적화하고 특정 워크로드의 리소스 사용량을 줄여 성능을 개선하고 비용을 절감할 수 있습니다.

DB 및 DB 클러스터 파라미터 그룹 사용

Amazon RDS와 Aurora는 데이터베이스 인스턴스 크기를 기반으로 특정 설정에 가장 적합한 파라미터 값을 자동으로 결정할 수 있습니다. 또한 데이터베이스 인스턴스 및 클러스터의 파라미터 그룹을 통한 성능 튜닝을 위한 파라미터 사용자 지정을 지원합니다.

DB 및 DB 클러스터 파라미터 그룹을 사용하여 메모리 사용량, 디스크 I/O, 네트워킹, 잠금 등 데이터베이스 엔진 동작의 다양한 측면을 제어하는 파라미터를 수정할 수 있습니다. 이러한 파라미터를 조정하면 데이터베이스 엔진을 특정 워크로드에 맞게 최적화하고 성능을 개선할 수 있습니다.

AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 Amazon RDS API를 사용하여 DB 및 DB 클러스터 파라미터 그룹을 생성하고 구성할 수 있습니다. 이 가이드에서는 를 사용하고 있다고 가정합니다. AWS CLI콘솔 및 API 지침은 Amazon RDS 설명서의 DB 파라미터 그룹 사용 및 DB 클러스터 파라미터 그룹 사용을 참조하십시오.

> **Important**
>
> 이 안내서에 제공된 AWS CLI 명령을 사용하려면 먼저 를 설치하고 구성해야 합니다. AWS CLI

DB 파라미터 그룹을 생성하고 구성하려면:

```
# Create a new DB parameter group
aws rds create-db-parameter-group \
  --db-parameter-group-name mydbparamgroup \
  --db-parameter-group-family postgres13 \
  --description "My DB Parameter Group"

# Modify a parameter on the DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <param group name> \
  --parameters "ParameterName=max_connections<parameter-
name>,ParameterValue=<value>,ApplyMethod=immediate"

# Verify DB parameters
aws rds describe-db-parameters \
  --db-parameter-group-name aurora-instance-1
```

DB 클러스터 파라미터 그룹을 만들고 구성하려면:

```
# Create a new DB cluster parameter group
aws rds create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --db-parameter-group-family postgres12 \
  --description "My new parameter group"

# Modify a parameter on the DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name aws-guide-cluster \
  --parameters "ParameterName=<parameter-name>,ParameterValue=,ApplyMethod=immediate"

# Allocate the new DB cluster parameter to your cluster
aws rds modify-db-cluster  \
  --db-cluster-identifier  \
  --db-cluster-parameter-group-name=-cluster

# Verify cluster parameters
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name=-cluster
```

> **Note**
>
> Aurora와 Amazon RDS는 변경할 수 없는 사전 구성된 값이 포함된 기본 파라미터 그룹을 제공합니다.

파라미터 그룹은 정적 또는 동적으로 설정할 수 있습니다. 동적 파라미터는 `ApplyMethod=immediate` 옵션 활성화 여부에 관계없이 즉시 적용됩니다. 정적 파라미터를 적용하려면 수동 재부팅이 필요합니다.

메모리 파라미터 조정

메모리 파라미터 조정은 Amazon RDS 및 Aurora PostgreSQL 호환 데이터베이스의 성능을 최적화하기 위한 필수 작업입니다. 쿼리 실행, 정렬, 인덱싱 및 캐싱과 같은 다양한 데이터베이스 작업에 메모리를 적절하게 할당하면 데이터베이스 성능을 크게 개선할 수 있습니다. 이 섹션에서는 Amazon RDS 및 Aurora PostgreSQL과 호환되는 가장 중요한 메모리 파라미터 중 일부에 대해 설명합니다. 여기에는 기본값, 적절한 값을 계산하는 공식, 변경 방법 등이 포함됩니다. 파라미터의 전체 목록은 Amazon RDS 설명서의 PostgreSQL용 RDS의 파라미터 사용 및 Aurora 설명서의 Amazon Aurora PostgreSQL 파라미터 사용을 참조하십시오.

이러한 파라미터를 최적화하려면 데이터베이스 워크로드와 Aurora 또는 Amazon RDS DB 인스턴스에서 사용 가능한 리소스를 깊이 이해해야 합니다. 시스템 성능은 크게 두 가지 범주의 파라미터, 즉 필수 파라미터와 우발적 파라미터의 영향을 받습니다.

필수 매개변수는 시스템 성능에 중요하고 직접적인 영향을 미치며 최적의 결과를 달성하는 데 필수적인 필수 매개변수입니다.

- 공유 버퍼
- 템프 버퍼
- 유효 캐시 크기
- 작업_메모
- maintenance_work_mem

조건부 매개변수는 시나리오 및 비즈니스별 매개변수입니다. 이들은 다른 요인에 따라 달라지며 중요한 매개변수를 지원하고 전체 시스템 성능을 극대화하는 데 간접적이면서도 중추적인 역할을 합니다.

- 랜덤_페이지_비용
- 시퀀스_페이지_비용
- 트랙_활동_쿼리_크기
- 유휴 상태_인_트랜잭션_세션_타임아웃
- 명세서_타임아웃
- 검색_경로
- 최대_연결

이러한 매개변수에 대해서는 다음 섹션에서 자세히 설명합니다.

shared_buffers

이 shared_buffers 파라미터는 PostgreSQL이 데이터를 메모리에 캐시하는 데 사용하는 메모리 양을 제어합니다. 이 매개 변수를 적절한 값으로 설정하면 쿼리 성능을 개선하는 데 도움이 될 수 있습니다.

Amazon RDS의 경우 DB 인스턴스의 사용 가능한 메모리를 기준으로 의 기본값이 {DBInstanceClassMemory/32768} 바이트로 shared_buffers 설정됩니다. Aurora의 경우 DB 인스턴스의 사용 가능한 메모리를 기준으로 기본값이 로 {DBInstanceClassMemory/12038,-50003} 설정됩니다. 이 파라미터의 최적값은 데이터베이스 크기, 동시 연결 수, 사용 가능한 인스턴스 메모리 등 여러 요인에 따라 달라집니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 shared_buffers 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify shared_buffers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=shared_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify shared_buffers on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=shared_buffers,ParameterValue=<new-
value>,ApplyMethod=immediate"
```

유형: 정적 (변경 사항을 적용하려면 재부팅해야 함)

기본값: {DBInstanceClassMemory/12038,-50003} PostgreSQL용 Amazon RDS에서는 {DBInstanceClassMemory/32768} 바이트, Aurora에서는 PostgreSQL과 호환됩니다. 대부분의 경우 이 방정식은 시스템 메모리의 약 25% 를 차지합니다. 이 지침에 따라 파라미터 그룹의 shared_buffers 설정은 바이트 또는 킬로바이트 대신 PostgreSQL의 기본 단위인 8K 버퍼를 사용하여 설정됩니다.

shared_buffers파라미터 설정은 성능에 상당한 영향을 미칠 수 있으므로 변경 내용을 철저하게 테스트하여 해당 값이 워크로드에 적합한지 확인하는 것이 좋습니다.

예

Amazon RDS 또는 Aurora에서 PostgreSQL 데이터베이스를 실행하는 금융 서비스 애플리케이션이 있다고 가정해 보겠습니다. 이 데이터베이스는 고객 거래 데이터를 저장하는 데 사용됩니다. 많은 수의 테이블이 있으며 많은 서버의 여러 애플리케이션에서 액세스할 수 있습니다. 응용 프로그램의 쿼리 성능이 느리고 CPU 사용량이 높습니다. shared_buffers매개 변수를 조정하면 성능 향상에 도움이 될 수 있다고 판단합니다.

Amazon RDS for PostgreSQL에서는 shared_buffers 기본값이 db.r5.xlarge 사용 가능한 메모리의 바이트 단위 (예: 3GB) 로 {DBInstanceClassMemory/32768} 설정됩니다. 적절한 값을 결정하기 위해 shared_buffers 사용 가능한 메모리의 기본값부터 시작하여 값을 점진적으로 늘려가면서 다양한 값을 사용하여 일련의 테스트를 실행합니다. shared_buffers 각 테스트에 대해 데이터베이스의 쿼리 성능과 CPU 사용량을 측정합니다.

테스트 결과를 바탕으로 값을 shared_buffers 8GB로 설정하면 워크로드에 대한 전체 쿼리 성능과 CPU 사용량이 가장 좋다고 판단합니다. 이 값은 데이터베이스 크기, 쿼리 수 및 복잡성, 동시 사용자 수, 사용 가능한 시스템 리소스 등 워크로드 특성에 대한 테스트 및 분석을 조합하여 결정됩니다. 변경 후에는 모니터링 시스템이 데이터베이스 성능을 검사하여 새 값이 워크로드에 적합한지 확인합니다. 그런 다음 필요에 따라 추가 매개 변수를 미세 조정하여 성능을 더욱 개선할 수 있습니다.

temp_buffers

temp_buffers는 Aurora PostgreSQL과 호환되고 PostgreSQL용 Amazon RDS의 주요 구성 파라미터로, 임시 테이블에 대한 정렬, 해시 및 집계 작업을 포함하는 워크로드의 성능에 상당한 영향을 미칠 수 있습니다. 이 파라미터는 임시 버퍼에 할당된 메모리 양을 결정하며, 이에 따라 해당 작업의 효율성과 속도에 영향을 미칩니다. 할당된 메모리가 충분하지 않으면 시스템이 임시 테이블에 대한 정렬temp_buffers, 해시 및 집계 작업에 더 느리고 덜 효율적인 방법을 사용해야 하므로 성능이 최적화되지 않을 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 temp_buffers 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify temp_buffers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=temp_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify temp_buffers on a DB cluster parameter group
```

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=temp_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 8MB

이 파라미터에 대한 자세한 내용은 PostgreSQL 설명서의 리소스 소비를 참조하십시오.

예

워크로드에 임시 테이블에 대한 정렬, 해싱 및 집계 작업이 많이 포함되는 경우 충분한 메모리를 할당하지 temp_buffers 못할 수 있습니다. 이 경우 시스템은 임시 테이블에 대한 정렬 작업을 수행해야 하므로 메모리 내 정렬, 해싱 및 집계 작업 대신 디스크 기반 메서드가 더 느려질 수 있습니다. 이로 인해 특히 대규모 데이터 세트를 포함하는 쿼리의 경우 쿼리 성능이 크게 저하될 수 있습니다. 의 값을 높이면 메모리에서 이러한 작업을 수행할 temp_buffers 수 있는 충분한 메모리가 확보되어 성능이 크게 향상될 수 있습니다.

의 temp_buffers 최적 값을 찾으려면 시스템 성능을 모니터링하고 성능이 최적화되지 않는 영역을 식별하십시오. 쿼리 응답 시간이 느리거나 CPU 사용률이 높으면 조정을 고려해 보세요. temp_buffers 예를 들어, 워크로드에 임시 테이블이 많이 포함되는 경우 의 값을 늘리면 이러한 테이블을 메모리에 저장하는 데 도움이 될 temp_buffers 수 있습니다. 이는 스토리지의 읽기/쓰기 I/O를 사용하는 것보다 훨씬 빠를 수 있습니다.

조금씩 다른 값을 시험해 보고 각 변경 후 시스템 성능을 주의 깊게 모니터링하십시오. temp_buffers 다양한 값이 성능에 미치는 영향을 분석하고 워크로드의 특정 특성에 따라 설정을 미세 조정하십시오.

effective_cache_size

이 effective_cache_size 매개 변수는 PostgreSQL이 데이터 캐싱에 사용할 수 있다고 가정해야 하는 메모리 양을 지정합니다. 이 매개 변수를 올바르게 설정하면 PostgreSQL에서 사용 가능한 메모리를 더 잘 활용할 수 있으므로 성능이 향상될 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 effective_cache_size 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify effective_cache_size on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=effective_cache_size,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify effective_cache_size on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=effective_cache_size,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 `ApplyMethod=immediate` 적용됨)

기본값: `SUM(DBInstanceClassMemory/12038,-50003)` KB

예

온라인 학습 플랫폼에는 사용자가 자주 액세스하는 강의 자료, 학생 데이터 및 기타 콘텐츠로 구성된 대규모 데이터베이스가 있습니다. 애플리케이션은 32GB 메모리가 있는 Amazon RDS PostgreSQL 용 Amazon `db.r5.xlarge` RDS 인스턴스에서 실행됩니다. 사용자가 자주 액세스하는 콘텐츠를 읽으려고 하면 애플리케이션 성능이 저하됩니다. 데이터베이스 서버의 리소스 사용량을 분석한 결과 PostgreSQL이 가용 메모리를 최적으로 사용하고 있지 않다는 것을 확인할 수 있습니다.

Amazon RDS for PostgreSQL의 `effective_cache_size` 파라미터는 서버에서 디스크 캐싱에 사용하는 메모리 양을 제어합니다. 인스턴스 클래스의 기본값은 `SUM({DBInstanceClassMemory/12038},-50003)` KB로 설정되지만 이 기본값이 모든 `db.r5.xlarge` 워크로드에 적절하지 않을 수 있습니다. 이 예시에서는 데이터베이스 서버에 자주 액세스하는 대량의 강좌 자료와 학생 데이터가 저장되어 있을 수 있습니다. `effective_cache_size`매개변수 값을 높이면 메모리에 더 많은 데이터가 캐시되어 필요한 디스크 읽기 횟수가 줄어들고 쿼리 성능이 향상될 수 있습니다.

쿼리를 실행하면 Amazon RDS for PostgreSQL은 먼저 쿼리에 필요한 데이터가 캐시에 이미 있는지 확인합니다. 그렇다면 데이터를 디스크에서 읽는 대신 메모리에서 읽을 수 있습니다. 데이터가 캐시에 없는 경우 디스크에서 읽어야 하므로 작업 속도가 느릴 수 있습니다.

온라인 학습 플랫폼의 경우 테스트 및 분석 후 16GB (사용 가능한 메모리의 절반) 로 `effective_cache_size` 설정하기로 결정할 수 있습니다. 이 값을 사용하면 PostgreSQL에서 사용 가능한 메모리를 더 잘 활용할 수 있으므로 필요한 디스크 읽기 수가 줄어들고 쿼리 성능이 향상됩니다.

work_mem

work_mem파라미터는 쿼리에서 정렬 및 해싱 작업에 사용하는 메모리 양을 제어합니다. 기본값은 4MB입니다. 쿼리에 여러 작업이 포함된 경우 각 작업에 최대 4MB를 사용할 수 있습니다. 정렬 또는 해싱이 필요한 작업에는 더 많은 메모리가 필요하므로 의 값을 늘리면 쿼리 성능이 향상될 work_mem 수 있습니다. 하지만 이 매개 변수를 너무 높게 설정하면 메모리 사용량이 너무 많아 성능이 저하될 수 있습니다.

다음 공식을 사용하여 최적값을 work_mem 계산할 수 있습니다.

```
work_mem = (available_memory / (max_connections * work_mem_fraction))
```

여기서 available_memory 은 서버에서 사용 가능한 총 메모리 양이고, max_connections는 허용되는 최대 연결 수이며, work_mem_fraction은 각 연결에 할당해야 하는 사용 가능한 메모리 양을 결정하는 분수입니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 work_mem 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify work_mem on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 4MB

예

소셜 미디어 분석 도구는 대량의 데이터를 처리하는데, 복잡한 정렬 및 조인 작업이 수반되는 쿼리로 인해 디스크 I/O가 높아지고 디스크로 유출됩니다. 값을 work_mem 4MB에서 16MB로 늘리면

PostgreSQL은 이러한 작업에 더 많은 메모리를 사용할 수 있습니다. 이렇게 하면 I/O 양이 줄어들고 쿼리 성능이 향상됩니다.

maintenance_work_mem

maintenance_work_mem파라미터는,, 인덱스 생성과 같은 유지 관리 작업에 사용되는 메모리 양을 제어합니다. VACUUM ANALYZE Amazon RDS와 Aurora에서 이 파라미터의 기본값은 64MB입니다.

이 파라미터의 적절한 값을 계산하려면 다음 공식을 사용할 수 있습니다.

```
maintenance_work_mem = (total_memory - shared_buffers) / (max_connections * 5)
```

Aurora PostgreSQL 호환 에디션과 PostgreSQL용 Amazon RDS는 다음 공식을 적용하여 최적의 값을 설정합니다.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 maintenance_work_mem 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify maintenance_work_mem on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=maintenance_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify maintenance_work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=maintenance_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 64MB

예

대규모 애플리케이션은 Aurora 또는 Amazon RDS에서 호스팅되는 PostgreSQL 데이터베이스를 사용합니다. 진공 청소 및 인덱싱과 같은 유지 관리 작업 중에는 데이터베이스가 느리고 응답하지 않는 것을 확인할 수 있습니다. 메모리 사용량, 유지 관리 작업 시간, CPU 사용량과 같은 메트릭을 모니터링하여 의 maintenance_work_mem 현재 값이 문제를 일으키는지 확인할 수 있습니다.

최적의 값을 결정하기 위해 maintenance_work_mem 파라미터를 조정하고 그 영향을 모니터링할 수 있습니다. 유지 관리 작업 중에 메모리 사용량이 지속적으로 높거나 작동 시간이 예상보다 길면 늘리는 maintenance_work_mem 것이 도움이 될 수 있습니다. 반대로 유지 관리 작업 중에 CPU 사용량이 지속적으로 높으면 줄이는 것이 maintenance_work_mem 도움이 될 수 있습니다. 조정 및 테스트를 반복하여 수행하면 메모리 사용량, 유지 관리 작업 시간 및 CPU 사용량이 최적의 균형을 이루는 최적의 값을 찾을 수 있습니다. maintenance_work_mem

조사 중에 64MB의 기본값이 데이터베이스 크기에 비해 너무 작다고 판단했다고 가정해 보겠습니다. maintenance_work_mem 따라서 유지 관리 작업을 완료하는 데 시간이 오래 걸리고, 다운타임이 과도하게 발생하고, 애플리케이션 성능이 저하됩니다. 이 문제를 해결하려면 maintenance_work_mem 매개 변수를 64MB에서 512MB (최적값으로 식별) 로 늘려 매개 변수를 조정해야 할 수 있습니다. 변경 사항을 적용하면 유지 관리 작업 시간을 2/3까지 개선할 수 있습니다. 예를 들어, 이전에는 완료하는 데 30분이 걸렸던 진공 작업이 이제는 10분밖에 걸리지 않을 수 있습니다. 이 최적화의 결과로 이제 데이터베이스가 유지 관리 활동을 더 효율적으로 처리할 수 있습니다.

random_page_cost

random_page_cost파라미터는 임의 페이지 액세스 수행 비용을 결정하는 데 도움이 됩니다. Amazon RDS와 Aurora의 쿼리 플래너는 테이블에 대한 다른 통계와 함께 이 파라미터를 사용하여 쿼리를 실행하기 위한 가장 효율적인 계획을 결정합니다.

seq_page_cost및 random_page_cost 매개 변수는 밀접하게 관련되어 있으며 일반적으로 플래너가 여러 액세스 방법의 비용을 비교하고 어느 것이 가장 효율적인지 결정하기 위해 함께 사용합니다. 따라서 이러한 매개 변수 중 하나를 변경하는 경우 다른 매개 변수를 조정해야 하는지 여부도 고려해야 합니다.

일반적으로 쿼리 플래너는 쿼리 실행 비용을 최소화하려고 합니다. 디스크 페이지 읽기 수와 값을 random_page_cost 조합하여 비용을 결정합니다. 값이 높을수록 순차 스캔이 선호되고 값이 낮을수록 인덱스 스캔이 선호되는 경향이 있습니다. random_page_cost 또한 값이 낮을수록 해시 조인 대신 중첩 루프 조인이 선호되는 경향이 있습니다.

random_page_cost파라미터 그룹이나 로컬 세션에 값이 설정되어 있지 않으면 파라미터는 PostgreSQL 엔진 디폴트 값 (4) 을 사용합니다. 서버 및 워크로드의 특정 특성에 따라 이 값을 조정할

수 있습니다. 워크로드에 사용되는 대부분의 인덱스가 메모리 또는 Aurora 계층형 캐시에 들어갈 경우 random_page_cost 의 값을 다음과 비슷한 값으로 변경하는 것이 적절합니다. seq_page_cost

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 random_page_cost 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify random_page_cost on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=random_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify random_page_cost on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=random_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 4

예

인덱싱되지 않은 열에 필터를 사용하여 자주 쿼리되는 테이블에 많은 양의 데이터를 저장하는 데이터베이스가 있다고 가정해 보겠습니다. 쿼리를 완료하는 데 시간이 오래 걸리고 쿼리 플래너가 데이터 액세스를 위한 가장 효율적인 계획을 선택하지 않습니다.

성능을 개선하는 한 가지 방법은 random_page_cost 파라미터를 줄이는 것입니다. 이 값을 1로 설정하면 임의 페이지 액세스 비용이 기본값보다 4배 저렴합니다. 기본값인 4를 그대로 두면 random_page_cost 임의 페이지 액세스가 순차적 페이지 액세스 (seq_page_cost매개 변수에 의해 결정됨, 기본값 1.0) 보다 4배 더 비쌉니다. 하지만 이 특정한 경우에는 저장소 유형에 따라 랜덤 페이지 액세스가 실제로 훨씬 더 비쌀 수 있습니다.

random_page_cost매개 변수 값을 줄이면 쿼리 플래너가 인덱스 기반 계획을 선택하거나 테이블의 특정 특성에 더 적합한 다른 액세스 방법을 사용할 가능성이 높아질 수 있습니다.

매개 변수를 변경한 후 쿼리 성능을 모니터링하고 필요에 따라 조정하는 것이 좋습니다. 또한 EXPLAIN 명령문을 사용하여 쿼리 플래너를 확인하여 효율적인 계획을 선택하고 있는지 확인해야 합니다.

이는 한 가지 예일 뿐입니다. 최적의 설정은 워크로드의 특정 특성에 따라 달라집니다. 또한 이는 성능 조정의 한 측면일 뿐이므로 쿼리 성능에 영향을 줄 수 있는 다른 매개 변수와 구성 옵션도 고려해야 합니다.

seq_page_cost

`seq_page_cost`파라미터는 순차적 페이지 액세스 수행 비용을 결정하는 데 도움이 됩니다. Amazon RDS와 Aurora의 쿼리 플래너는 테이블에 대한 다른 통계와 함께 이 파라미터를 사용하여 쿼리를 실행하기 위한 가장 효율적인 계획을 결정합니다.

`seq_page_cost`및 `random_page_cost` 매개 변수는 밀접하게 관련되어 있으며 일반적으로 플래너가 여러 액세스 방법의 비용을 비교하고 어느 것이 가장 효율적인지 결정하기 위해 함께 사용합니다. 따라서 이러한 매개 변수 중 하나를 변경하는 경우 다른 매개 변수를 조정해야 하는지 여부도 고려해야 합니다.

테이블에 순차적으로 액세스하는 경우 PostgreSQL은 운영 체제의 파일 시스템 캐시를 사용하여 더 빠르게 액세스할 수 있습니다. 기본적으로 `seq_page_cost` 는 1.0으로 설정되어 있으며, 이 경우 순차적 페이지 액세스가 단일 디스크 블록 읽기 비용만큼 저렴하다고 가정합니다. 주로 순차적으로 액세스하는 테이블이 있지만 IOPS가 적어 디스크 액세스가 느려지는 경우 디스크 액세스에 드는 추가 비용을 반영하여 의 값을 높이는 `seq_page_cost` 것이 좋습니다.

이 매개 변수의 값을 변경하면 시스템에서 실행되는 모든 쿼리에 영향을 미치므로 다른 값으로 쿼리를 테스트하여 특정 사용 사례에 맞는 최적의 값을 결정하는 것이 좋습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `seq_page_cost` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify seq_page_cost on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=seq_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify seq_page_cost on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=seq_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 `ApplyMethod=immediate` 적용됨)

기본값: 1.0

예

주로 순차적으로 액세스되는 테이블에 많은 양의 데이터를 저장하는 데이터베이스가 있다고 가정해 보겠습니다. 테이블은 주로 보고에 사용되며 쿼리 수행 속도가 매우 느리고 쿼리 플래너가 데이터 액세스를 위한 가장 효율적인 계획을 선택할 수 없습니다.

성능을 개선하는 한 가지 방법은 `seq_page_cost` 파라미터를 줄이는 것입니다. 기본값은 1.0이며, 이 경우 순차적 페이지 액세스가 단일 디스크 블록 읽기 비용만큼 저렴하다고 가정합니다. 하지만 이 특정한 경우에는 스토리지 유형 (I/O에 따라 다름) 때문에 순차적 페이지 액세스가 실제로 그보다 더 비쌀 수 있습니다. `seq_page_cost`0.5로 설정하면 순차적 페이지 액세스 비용이 기본값의 절반으로 줄어듭니다. 이렇게 변경하면 쿼리 플래너가 테이블의 특정 특성에 더 적합한 순차 액세스 방법을 사용하는 계획을 선택할 가능성이 높아질 수 있습니다.

매개 변수를 변경한 후 쿼리 성능을 모니터링하고 필요에 따라 조정하는 것이 좋습니다. 또한 `EXPLAIN` 명령문을 사용하여 쿼리 계획을 확인하여 효율적인 계획을 선택하고 있는지 확인해야 합니다.

이는 한 가지 예일 뿐입니다. 최적의 설정은 워크로드의 특정 특성에 따라 달라집니다. 또한 이는 성능 조정의 한 측면일 뿐이므로 쿼리 성능에 영향을 미치는 다른 매개 변수와 구성 옵션도 고려해야 합니다.

track_activity_query_size

`track_activity_query_size`파라미터는 뷰의 각 활성 세션에 대해 기록되는 쿼리 문자열의 크기를 제어합니다. `pg_stat_activity` 기본적으로 쿼리 문자열의 처음 1,024바이트만 PostgreSQL용 Amazon RDS에 기록되며, Aurora PostgreSQL과 호환되는 Aurora에는 4,096바이트가 로깅됩니다. 더 긴 쿼리를 기록하려면 이 파라미터를 더 높은 값으로 설정할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `track_activity_query_size` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify track_activity_query_size on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
```

```
  --parameters
 "ParameterName=track_activity_query_size,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify track_activity_query_size on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=track_activity_query_size,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 정적 (변경 사항을 적용하려면 재부팅해야 함)

기본값: 1,024바이트 (PostgreSQL용 Amazon RDS), 4,096바이트 (Aurora PostgreSQL과 호환 가능)

예

Amazon RDS for PostgreSQL의 데이터베이스에서 쿼리 성능이 느려지고 있으며, 이 문제가 장기 실행 쿼리와 관련이 있는 것 같습니다. 뷰에 더 긴 쿼리를 로깅하여 더 자세히 조사할 수 있습니다. `pg_stat_activity`

`track_activity_query_size`매개 변수 값을 늘리면 로깅이 증가하여 데이터베이스 성능에 영향을 미칠 수 있습니다. 문제가 해결된 후 매개변수를 기본값인 1,024로 다시 설정하는 것이 좋습니다.

idle_in_transaction_session_timeout

파라미터는 유휴 트랜잭션이 중지되기 전에 대기하는 시간을 제어합니다. `idle_in_transaction_session_timeout`

PostgreSQL용 Amazon RDS와 Aurora PostgreSQL과 호환되는 이 파라미터의 기본값은 86,400,000 밀리초입니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `idle_in_transaction_session_timeout` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify idle_in_transaction_session_timeout on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=idle_in_transaction_session_timeout,ParameterValue=<new_value>,ApplyMethod=imme

# Modify idle_in_transaction_session_timeout on a DB cluster parameter group
```

```
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=idle_in_transaction_session_timeout,ParameterValue=<new_value>,ApplyMethod=imme
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 86,400,000밀리초 (Aurora PostgreSQL 호환)

예

온라인 주문을 처리하는 전자 상거래 애플리케이션이 있습니다. 이 애플리케이션은 Amazon RDS 또는 Aurora에서 호스팅되는 PostgreSQL 데이터베이스를 사용합니다. 고객이 주문할 때마다 애플리케이션은 재고 및 주문 기록을 업데이트하기 위해 새 거래를 시작합니다.

트랜잭션을 오랫동안 유휴 상태로 두면 다른 트랜잭션이 동일한 레코드에 액세스하지 못하게 되어 성능 문제가 발생하고 잠재적으로 애플리케이션 다운타임이 발생할 수 있습니다. 또한 유휴 트랜잭션이 제대로 중지되지 않으면 메모리 및 CPU와 같은 귀중한 시스템 리소스가 소모될 수 있습니다.

이러한 문제를 방지하려면 애플리케이션에 적합한 값으로 idle_in_transaction_session_timeout 파라미터를 설정할 수 있습니다. 예를 들어 5분 (300초) 으로 설정하여 5분 이상 유휴 상태로 남아 있는 모든 트랜잭션이 자동으로 중지되도록 할 수 있습니다. 이렇게 하면 시스템 리소스를 효율적으로 사용하고 응용 프로그램이 속도 저하 없이 많은 주문을 처리할 수 있습니다. 에 적절한 값을 설정하면 Amazon RDS 또는 Aurora에서 애플리케이션이 최적의 성능을 발휘하도록 할 수 있습니다. idle_in_transaction_session_timeout

statement_timeout

statement_timeout파라미터는 쿼리가 중지되기 전까지 실행할 수 있는 최대 시간을 설정합니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 statement_timeout 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify statement_timeout on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=statement_timeout,ParameterValue=<new_value>,ApplyMethod=immediate"
```

```
# Modify statement_timeout on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=statement_timeout,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 0밀리초 (타임아웃 없음)

예

웹 애플리케이션을 통해 사용자는 대규모 제품 데이터베이스를 검색할 수 있습니다. 검색 쿼리를 완료하는 데 시간이 오래 걸려 사용자의 응답 시간이 느려질 수 있습니다. 이 문제를 해결하려면 statement_timeout 매개변수를 낮은 값 (예: 10초) 으로 설정하면 됩니다. 이렇게 하면 10초 이상 걸리는 모든 쿼리를 강제로 중지할 수 있습니다.

이는 과감한 조치처럼 보일 수 있지만 실제로는 성능 향상에 매우 효과적일 수 있습니다. 대부분의 경우 쿼리가 오래 실행되는 이유는 제대로 최적화되지 않은 SQL 쿼리나 비효율적인 인덱스로 인해 발생합니다. statement_timeout값을 낮게 설정하면 이러한 문제가 있는 쿼리를 식별하고 최적화 조치를 취할 수 있습니다.

예를 들어 특정 검색 쿼리의 제한 시간이 지속적으로 초과되는 것을 발견했다고 가정해 보겠습니다. EXPLAIN및 와 EXPLAIN ANALYZE 같은 도구를 사용하여 쿼리를 분석하고 성능 병목 현상을 식별할 수 있습니다. 문제가 확인되면 새 색인을 추가하거나, 쿼리를 다시 작성하거나, 다른 검색 알고리즘을 사용하여 쿼리를 최적화하는 조치를 취할 수 있습니다. 이러한 방식으로 SQL 쿼리를 지속적으로 분석하고 최적화하면 응용 프로그램의 성능을 크게 향상시킬 수 있습니다.

search_path

search_path 파라미터는 SQL 문에서 스키마에서 객체를 검색하는 순서를 결정합니다. 기본값은 입니다. 즉$user, public, PostgreSQL은 먼저 스키마에서 사용자 이름과 일치하는 객체를 검색한 다음 공용 스키마에서 객체를 검색합니다.

스키마 수가 많거나 특정 스키마의 객체에 액세스해야 하는 경우 search_path 매개 변수를 변경하면 성능을 개선하는 데 도움이 될 수 있습니다. 특정 search_path 스키마로 설정하면 PostgreSQL은 여러 스키마를 검색할 필요 없이 객체를 더 빨리 찾을 수 있습니다.

Amazon RDS 및 Aurora에서 search_path 파라미터를 변경하려면 레벨에 대해 다음 명령을 ROLE 사용할 수 있습니다.

```
ALTER ROLE <username> SET search_path = <schema>;
```

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 search_path 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify search_path on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=search_path,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify search_path on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=search_path,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: $user, public

예

Amazon RDS for PostgreSQL용 Amazon RDS 또는 Aurora PostgreSQL 호환 데이터베이스를 사용하는 각 테넌트마다 별도의 스키마가 있는 멀티 테넌트 애플리케이션이 있고, 여러 스키마의 데이터를 조인하는 쿼리를 실행해야 합니다.

기본적으로 Amazon RDS와 Aurora는 검색 경로를 사용하여 지정된 테이블에 사용할 스키마를 결정합니다. 검색 경로는 스키마 이름을 한정하지 않고 테이블을 참조할 때 PostgreSQL이 순서대로 검색하는 스키마 이름 목록입니다. 기본적으로 Amazon RDS와 Aurora는 먼저 스키마에서 현재 사용자와 이름이 같은 테이블을 찾은 다음 공개 스키마를 살펴봅니다.

및 라는 이름의 tenant1 여러 스키마의 테이블을 조인하는 쿼리를 실행한다고 가정해 보겠습니다. tenant2 tenant3 테넌트 스키마를 사용하려면 쿼리에 스키마 이름을 포함하면 됩니다.

```
SELECT *
FROM tenant1.table1
JOIN tenant2.table2 ON tenant1.table1.id = tenant2.table2.id
JOIN tenant3.table3 ON tenant2.table2.id = tenant3.table3.id;
```

그러나 더 효율적인 방법은 AWS CLI 구문 섹션의 명령을 사용하여 테넌트 스키마를 포함하도록 search_path 매개 변수를 변경하는 것입니다. PostgreSQL 세션에서도 이 SET 명령을 사용할 수 있습니다.

```
SET search_path = tenant1, tenant2, tenant3, public;
```

그러면 스키마 이름을 한정하지 않고 쿼리를 작성할 수 있습니다.

```
SELECT *
FROM table1
JOIN table2 ON table1.id = table2.id
JOIN table3 ON table2.id = table3.id;
```

이렇게 하면 쿼리를 더 간결하고 읽기 쉽게 만들 수 있으며, 여러 스키마의 테이블 조인과 관련된 쿼리가 많은 경우 애플리케이션 코드를 단순화할 수도 있습니다.

max_connections

이 max_connections 파라미터는 PostgreSQL 데이터베이스의 최대 동시 연결 수를 설정합니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 max_connections 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify max_connections on a DB parameter group
aws rds modify-db-parameter-group \
--db-parameter-group-name <parameter_group_name> \
--parameters
 "ParameterName=max_connections,ParameterValue=<new_value>,ApplyMethod=pending-reboot"

# Modify max_connections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name <parameter_group_name> \
--parameters
 "ParameterName=max_connections,ParameterValue=<new_value>,ApplyMethod=pending-reboot"
```

유형: 정적 (변경 사항을 적용하려면 재부팅해야 함)

기본값: LEAST(DBInstanceClassMemory/9531392,5000) 연결

Amazon RDS 또는 max_connections Aurora에서의 사용을 최적화하고 성능에 미치는 영향을 최소화하려면 다음 모범 사례를 고려하십시오.

- 사용 가능한 시스템 리소스를 기반으로 파라미터 값을 설정합니다.
- 연결 사용량을 모니터링하여 한도에 빠르게 도달하지 않도록 하십시오.
- 연결 풀링을 사용하여 필요한 연결 수를 줄이십시오.
- 연결 풀링에는 Amazon RDS 프록시를 사용하십시오.

PostgreSQL용 Amazon RDS 또는 PostgreSQL과 호환되는 Amazon Aurora를 사용할 때는 사용 가능한 인스턴스 유형과 할당된 리소스를 고려하고 메모리와 CPU 용량에 초점을 max_connections 맞추십시오. 스토리지 및 I/O 세부 정보는 에서 관리되므로 Amazon CloudWatch 또는 Amazon RDS 콘솔 등을 FreeableMemory 통해 일반적인 워크로드 특성과 시스템 지표를 모니터링하여 연결에 필요한 메모리가 충분한지 확인할 수 있습니다. AWS높은 CPUUtilization 값을 모니터링하면 조정이 필요함을 나타낼 수 있습니다. max_connections너무 높게 설정하면 메모리 사용량에 영향을 미치고 I/O에 간접적으로 영향을 미칠 수 있으므로 너무 높게 설정하지 마십시오. 각 연결마다 메모리가 소모된다는 점에 유의하세요. 적절한 균형을 찾으려면 천천히 속도를 max_connections 높여 시스템에 미치는 영향을 확인하십시오. 성능이 저하되거나 CPU 사용량이 늘어나는 징후가 있는지 살펴보세요. 애플리케이션이 여전히 잘 작동하는지 확인하세요. Aurora의 읽기 전용 복제본과 같은 기능을 사용하여 읽기 트래픽을 분산하고 기본 인스턴스의 부하를 줄일 수 있습니다. 관찰된 사용 패턴을 기반으로 max_connections 정기적으로 검토하고 조정하여 주어진 리소스 제약 내에서 데이터베이스 성능이 최적화되도록 하십시오.

오토진공 파라미터 조정

PostgreSQL용 Amazon RDS 데이터베이스 및 Aurora PostgreSQL과 호환되는 경우 진공 청소기로 알려진 정기적인 유지 관리가 필요합니다. Autovacuum은 오래되거나 불필요한 데이터를 제거하여 데이터베이스의 공간을 확보하는 내장된 PostgreSQL 유틸리티입니다. autovacuum 프로세스는 정기적으로 백그라운드에서 VACUUM 명령을 실행합니다.

자동 진공 설정을 조정하는 것은 PostgreSQL 또는 Aurora PostgreSQL과 호환되는 데이터베이스 시스템용 Amazon RDS의 성능, 안정성 및 가용성을 유지하기 위한 중요한 단계입니다. 워크로드와 데이터베이스 크기에 맞게 autovacuum 매개 변수를 조정하면 자동 진공 프로세스의 성능을 최적화하고 시스템 리소스에 미치는 영향을 줄여 데이터베이스의 전반적인 상태를 개선할 수 있습니다.

자동 진공 설정을 조정하는 것 외에도 Amazon RDS 및 Aurora에서 사용할 수 있는 도구 및 지표를 사용하여 데이터베이스 및 해당 구성 요소의 성능을 모니터링하는 것도 중요합니다. 팽창, 여유 공간, 쿼리 실행 시간과 같은 성능 지표를 모니터링하면 심각한 문제로 발전하기 전에 잠재적 문제를 식별하고 적절한 조치를 취해 문제를 해결할 수 있습니다.

이 섹션에서는 다음과 같은 autovacuum 주제 및 매개변수에 대해 설명합니다.

- 진공 및 분석 명령
- 팽창 여부 확인
- 자동 진공청소기
- autovacuum_work_mem
- autovacuum_naptime
- autovacuum_max_workers
- 자동진공_진공_스케일_팩터
- 자동진공_진공_임계값
- 자동진공_분석_스케일_팩터
- 자동진공_분석_임계값
- 자동진공청소기_비용_한도

자동진공에 대한 추가 정보는 다음 링크를 참조하십시오.

- PostgreSQL 환경을 위한 Amazon RDS의 자동 진공 청소기에 대한 이해 (블로그 게시물)
- PostgreSQL용 Amazon RDS에서 PostgreSQL 자동 진공 청소기로 작업하기 (Amazon RDS 설명서)

- PostgreSQL용 Amazon RDS와 Amazon Aurora PostgreSQL용 병렬 진공 청소기 (블로그 게시물)

진공 청소 및 분석 명령

VACUUM가비지 수집 및 데이터베이스 분석 (선택 사항) 대부분의 응용 프로그램에서는 autovacuum 데몬이 진공 청소를 수행하도록 하는 것으로 충분합니다. 하지만 일부 관리자는 autovacuum의 데이터베이스 매개 변수를 수정하거나 스케줄러에 따라 실행할 수 있는 수동 관리 VACUUM 명령을 사용하여 데몬의 활동을 보완하거나 대체하려고 할 수 있습니다.

VACUUM데드 튜플이 차지하고 있는 스토리지를 회수합니다. 표준 PostgreSQL 작업에서는 업데이트로 인해 튜플이 삭제되거나 더 이상 사용되지 않게 되더라도 작업이 수행될 때까지 테이블에서 물리적으로 제거되지 않습니다. VACUUM 따라서 VACUUM 정기적으로 실행하는 것이 좋으며, 특히 자주 업데이트되는 테이블에서는 실행하는 것이 좋습니다.

PostgreSQL 및 Aurora PostgreSQL과 호환되는 Amazon RDS에서는 조정 VACUUM 매개 변수가 특히 중요합니다. 이러한 관리형 데이터베이스 서비스는 자체 관리형 PostgreSQL 데이터베이스와 특성이 다르기 때문입니다. 이러한 차이는 진공 작업의 성능에 영향을 미칠 수 있습니다. 리소스 사용을 최적화하고 진공 작업이 데이터베이스 시스템의 성능 및 가용성에 부정적인 영향을 미치지 않도록 하려면 VACUUM 매개변수 조정이 필수적입니다.

다음은 Aurora PostgreSQL과 호환되고 PostgreSQL용 Amazon RDS에서 VACUUM 명령과 함께 사용할 수 있는 몇 가지 매개 변수입니다.

- FULL
- FREEZE
- VERBOSE
- ANALYZE
- DISABLE_PAGE_SKIPPING
- table_name
- column_name

VACUUM ANALYZE선택한 각 테이블에 대해 작업을 수행한 다음 VACUUM 작업을 수행합니다. ANALYZE 일상적인 유지 관리를 효율적으로 수행할 수 있는 방법을 제공합니다.

FULL옵션 없이 VACUUM 명령을 사용하면 재사용할 공간을 확보할 수 있습니다. 테이블을 독점적으로 잠글 필요가 없으므로 표준 읽기 및 쓰기 작업 중에 이 명령을 실행할 수 있습니다. 하지만 대부분의 경우 이 명령은 운영 체제에 추가 공간을 반환하지 않지만 동일한 테이블 내에서 다시 사용할 수 있도록

유지합니다. VACUUM FULL테이블의 전체 내용을 추가 공간 없이 새 디스크 파일에 다시 쓰고 사용하지 않은 공간을 운영 체제에 반환할 수 있습니다. 이 양식은 속도가 훨씬 느리고 각 테이블을 ACCESS EXCLUSIVE 잠가야 합니다.

이러한 파라미터에 대한 자세한 내용은 PostgreSQL 설명서를 참조하십시오.

Aurora와 Amazon RDS에서 autovacuum은 VACUUM 및 ANALYZE 명령을 정기적으로 실행하여 데이터베이스와 서버의 중복 데이터를 정리하는 데몬 (백그라운드 유틸리티) 프로세스입니다. 자동 진공 청소기를 사용하는 경우에도 최적의 성능을 보장하기 위해 다음 섹션에서 설명하는 자동 진공 설정을 검토하고 조정하는 것이 좋습니다.

팽창 여부 확인

다음 SQL 쿼리는 XML 스키마의 각 테이블을 검사하여 디스크 공간을 낭비하는 데드 행 (튜플) 을 식별합니다.

```
SELECT schemaname || '.' || relname as tuplename,
    n_dead_tup,
    (n_dead_tup::float / n_live_tup::float) * 100 as pfrag
FROM pg_stat_user_tables
WHERE schemaname = 'xml' and n_dead_tup > 0 and n_live_tup > 0 order by pfrag desc;
```

이 쿼리가 죽은 튜플의 높은 비율 (pfrag) 을 반환하는 경우 명령을 사용하여 공간을 확보할 수 있습니다. VACUUM

트랜잭션 전후의 데이터 크기를 모니터링하려면 특정 데이터베이스에 연결한 후 셸에서 다음 쿼리를 실행합니다.

```
SELECT pg_size_pretty(pg_relation_size('table_name'));
```

autovacuum

autovacuum구성 파라미터를 사용하여 autovacuum을 전역으로 설정하거나 테이블의 autovacuum_enabled 열을 특정 테이블로 설정하거나 특정 테이블에 맞게 설정하여 테이블별로 변경할 수 있습니다. pg_class true false

테이블에서 autovacuum을 활성화하면 데이터베이스 서버는 데이터베이스 관리자의 개입 없이 테이블에서 데드 행과 튜플을 정기적으로 스캔하고 백그라운드에서 제거합니다. 이렇게 하면 테이블을 작게 유지하고, 쿼리 성능을 개선하고, 백업 크기를 줄이는 데 도움이 됩니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹을 autovacuum 활성화합니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=autovacuum,ParameterValue=true,ApplyMethod=immediate"

# Modify autovacuum on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=autovacuum,ParameterValue=true,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 활성화됨

psql을 사용하여 특정 테이블에서 autovacuum을 비활성화하거나 활성화할 수도 있습니다.

```
ALTER TABLE <table_name> SET (autovacuum_enabled = true);
```

너무 많이 청소하면 성능에 영향을 줄 수 있으므로 자동 진공 프로세스의 성능과 데이터베이스 성능을 모니터링하고 필요에 따라 설정을 조정하는 것이 중요합니다.

예

PostgreSQL 데이터베이스에는 대량의 쓰기 및 삭제 작업을 수신하는 테이블이 있습니다. autovacuum이 없으면 이 테이블은 결국 데드 행 (즉, 삭제 대상으로 표시되었지만 테이블에서 아직 물리적으로 제거되지 않은 행) 으로 가득 차게 됩니다. 이렇게 데드 로우가 있으면 디스크 공간을 차지하고 쿼리 속도가 느려지고 백업 크기가 커집니다. 테이블에서 autovacuum을 활성화하여 데드 행을 자동으로 스캔하고 제거함으로써 이러한 문제를 완화할 수 있습니다.

autovacuum_work_mem

autovacuum_work_mem는 자동 진공 프로세스가 진공 청소 또는 분석과 같은 테이블 유지 관리 작업을 수행할 때 사용하는 메모리 양을 제어하는 PostgreSQL 구성 매개 변수입니다.

Aurora와 Amazon RDS에서는 값을 조정하여 성능을 최적화할 수 있습니다. autovacuum_work_mem

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹을 autovacuum_work_mem 활성화합니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_work_mem on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: Aurora PostgreSQL과 호환되는 경우 GREATEST({DBInstanceClassMemory/32768},131072) KB, PostgreSQL의 경우 Amazon RDS의 경우 64MB. 하지만 기본값은 사용 중인 Amazon RDS 또는 Aurora의 특정 버전에 따라 달라질 수 있습니다.

예

PostgreSQL용 Amazon RDS 데이터베이스에는 자주 업데이트되는 대용량 테이블이 있습니다. 시간이 지나면서 데이터베이스 속도가 느려지고 자동 진공 청소기를 완료하는 데 시간이 너무 오래 걸린다는 의심이 들 것입니다.

조사의 일환으로 시스템 로그를 확인하고, pg_stat_activity 뷰를 사용하여 현재 실행 중인 쿼리와 프로세스를 확인하고, pg_stat_user_tables 뷰를 확인하여 각 테이블의 통계를 확인하고, pg_settings 뷰를 사용하여 시스템의 가용 메모리와 값을 비교하고, 메모리 사용량이 급증하는지 모니터링합니다. autovacuum_work_mem 이 정보를 수집한 후에는 워크로드에 필요한 최적의 값을 설정할 autovacuum_work_mem 수 있습니다. 메모리 사용량과 성능 간의 적절한 균형을 찾기 위해 시스템에서 사용 가능한 메모리의 4분의 1로 설정하는 것이 좋습니다. 값을 변경한 후 데이터베이스의 성능을 모니터링한 결과 autovacuum이 이전보다 훨씬 빠르게 완료되고 데이터베이스 성능이 전반적으로 더 빨라지는 것을 확인할 수 있습니다.

autovacuum_naptime

autovacuum_naptime파라미터는 자동 진공 프로세스의 연속 실행 사이의 시간 간격을 제어합니다. 기본값은 PostgreSQL의 경우 Amazon RDS의 경우 15초이고 Aurora PostgreSQL과 호환되는 경우 5초입니다.

예를 들어, PostgreSQL용 Amazon RDS 데이터베이스에 대량의 쓰기 및 삭제 작업을 수신하는 테이블이 있다고 가정해 보겠습니다. 기본 설정을 유지할 경우 잦은 자동 진공 스캔으로 인해 트랜잭션 사용량이 많은 이 테이블에 지장을 줄 수 있습니다. 이 매개 변수를 높은 값으로 설정하면 연속 스캔 간격이 길어지고 데드 행이 제거되는 빈도도 줄어듭니다.

특히 사용량이 많고 이미 CPU 또는 I/O 부하가 높은 서버가 있는 경우 진공 프로세스로 인한 부하를 관리하는 autovacuum_naptime 데 사용할 수 있습니다. 낮잠 시간을 오래 설정할수록 autovacuum 실행 빈도가 줄어들어 서버의 부하가 줄어듭니다. 그러나 매우 높은 autovacuum_naptime 값으로 설정하면 PostgreSQL 테이블이 늘어나고 데드 행이 누적되어 성능이 저하될 수 있습니다. autovacuum 프로세스의 성능을 모니터링하고 필요에 따라 설정을 조정하는 것이 좋습니다. autovacuum_naptime

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 autovacuum_naptime 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_naptime on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_naptime,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_naptime on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_naptime,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 15초 (PostgreSQL용 Amazon RDS), 5초 (Aurora PostgreSQL과 호환)

autovacuum_max_workers

autovacuum_max_workers파라미터는 autovacuum 프로세스가 생성할 수 있는 최대 작업자 프로세스 수를 제어합니다. 각 작업자 프로세스는 단일 테이블을 진공 청소기로 청소하거나 분석하는 역할을 합니다.

예를 들어 자주 업데이트되고 삭제되는 테이블이 많은 대규모 데이터베이스가 있다고 가정해 보겠습니다. autovacuum_max_workers를 1과 같이 낮은 값으로 설정하면 한 번에 한 테이블만 비울 수 있고 모든 테이블을 정리하는 데 시간이 더 오래 걸립니다. 높은 값 (예: 8) 으로 설정하면 autovacuum_max_workers 최대 8개의 테이블을 동시에 청소할 수 있습니다. 이렇게 하면 테이블이 많이 포함된 데이터베이스의 정리 프로세스가 더 빨라질 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 autovacuum_max_workers 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_max_workers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_max_workers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_max_workers on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_max_workers,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 정적 (변경 사항을 적용하려면 재부팅해야 함)

기본값: GREATEST(DBInstanceClassMemory/64371566592,3) 작업자

autovacuum_max_workers설정을 높이면 서버의 부하가 증가하여 리소스가 충분하지 않을 경우 성능에 영향을 미칠 수 있습니다. 최적의 설정은 데이터베이스의 특정 요구 사항, 데이터베이스 크기, 포함된 테이블 수에 따라 달라집니다. 다양한 값을 실험하고 성능을 모니터링하여 사용 사례에 맞는 최적의 설정을 찾는 것이 좋습니다.

autovacuum_vacuum_scale_factor

구성 autovacuum_vacuum_scale_factor 파라미터는 테이블을 진공 청소기로 청소할 때 자동 진공 프로세스를 얼마나 적극적으로 실행해야 하는지를 제어합니다.

진공 스케일 팩터는 자동 진공 청소기로 테이블을 청소하기 전에 수정해야 하는 테이블 내 총 튜플 수의 일부에 불과합니다. 기본값은 0.1입니다. 즉, 튜플의 10% 를 수정해야 합니다. 예를 들어, 테이블에 1,000,000개의 튜플이 있고 그 중 100,000개의 튜플이 비활성 또는 삭제된 것으로 표시된 경우 autovacuum은 제어 요소인 의 값에 따라 테이블을 진공 청소기로 청소합니다. autovacuum_vacuum_threshold

autovacuum_vacuum_scale_factor파라미터를 통해 진공 공정의 실행 빈도를 제어할 수 있습니다. 테이블이 쓰기 작업을 많이 받는 경우 autovacuum이 더 자주 실행되도록 진공 스케일링 팩터를 낮추고 테이블을 더 작게 유지하는 것이 좋습니다. 반대로, 테이블에서 쓰기 작업을 거의 받지 않는 경우 autovacuum 실행 빈도를 줄이고 리소스를 절약할 수 있도록 진공 스케일 팩터를 높이는 것이 좋습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 autovacuum_vacuum_scale_factor 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_vacuum_scale_factor on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediate

# Modify autovacuum_vacuum_scale_factor on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediate
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 0.1

autovacuum_vacuum_scale_factor매개변수는autovacuum_vacuum_threshold, autovacuum_vacuum_cost_limit, and autovacuum_naptime 매개변수와 함께 작동합니다.

이 파라미터에 대한 자세한 내용은 PostgreSQL 환경용 Amazon RDS의 자동 진공 이해하기 AWS 블로그 게시물을 참조하십시오.

autovacuum_vacuum_threshold

`autovacuum_vacuum_threshold`파라미터는 autovacuum이 테이블을 진공 청소기로 청소하기 전에 테이블에서 발생해야 하는 최소 튜플 업데이트 또는 삭제 작업 수를 제어합니다. 이 설정은 작업 빈도가 높지 않은 테이블에서 불필요한 진공 청소를 방지하는 데 유용할 수 있습니다. 기본값은 50이며, 이는 PostgreSQL 엔진 기본값이며, PostgreSQL용 Amazon RDS와 Aurora PostgreSQL과 호환되는 경우 모두 PostgreSQL 엔진 기본값입니다.

예를 들어, 100,000개의 행으로 구성된 테이블이 50으로 설정되어 있다고 가정해 보겠습니다. `autovacuum_vacuum_threshold` 테이블이 업데이트 또는 삭제 횟수가 49회만 수신되는 경우 autovacuum은 테이블을 진공 청소기로 청소하지 않습니다. 테이블이 50회 이상 업데이트 또는 삭제를 받으면 autovacuum은 테이블에 테이블 행 수를 제어 요소로 `autovacuum_vacuum_scale_factor` 곱한 값에 따라 테이블을 진공 청소기로 청소합니다.

이 매개변수를 너무 높게 설정하면 테이블이 늘어나고 데드 행이 누적되어 성능에 영향을 미칠 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `autovacuum_vacuum_threshold` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_vacuum_threshold on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=autovacuum_vacuum_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_vacuum_threshold on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=autovacuum_vacuum_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 `ApplyMethod=immediate` 적용됨)

기본값: 50회 작업

autovacuum_vacuum_threshold매개변수는autovacuum_vacuum_scale_factor, autovacuum_vacuum_cost_limit, and autovacuum_naptime 매개변수와 함께 작동합니다. 최적의 설정은 데이터베이스의 특정 요구 사항 및 테이블 크기에 따라 달라집니다.

이 파라미터에 대한 자세한 내용은 PostgreSQL 환경용 Amazon RDS의 자동 진공 이해하기 AWS 블로그 게시물을 참조하십시오.

autovacuum_analyze_scale_factor

autovacuum_analyze_scale_factor파라미터는 테이블 내 데이터 분포에 대한 통계를 분석 (수집) 할 때 autovacuum 프로세스가 얼마나 적극적이어야 하는지를 제어합니다.

autovacuum 프로세스는 이 파라미터를 사용하여 테이블의 튜플 수를 기반으로 임계값을 계산합니다. 튜플 삽입, 업데이트 또는 삭제 수가 이 임계값을 초과하는 경우 autovacuum은 테이블을 분석합니다. PostgreSQL용 Amazon RDS와 PostgreSQL과 호환되는 Aurora 모두의 기본값은 0.05입니다 (즉, 튜플의 5% 를 수정해야 함).

예를 들어 테이블에 1,000,000개의 튜플이 있고 기본값을 0.05로 유지한다고 가정해 보겠습니다. autovacuum_analyze_scale_factor 테이블이 50,000회 이상 업데이트 또는 삭제된 경우 autovacuum은 autovacuum_analyze_threshold 값에 따라 테이블 행 수를 제어 요소로 더하여 테이블을 진공 청소기로 청소합니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 autovacuum_analyze_scale_factor 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_analyze_scale_factor on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediat

# Modify autovacuum_analyze_scale_factor on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediat
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 0.05 (5%)

쿼리 플래너가 데이터 액세스 방법 및 구성 방법 등 정보에 입각한 결정을 내리려면 통계를 수집해야 합니다. 따라서 autovacuum 프로세스의 성능을 모니터링하고 필요에 따라 설정을 조정하여 통계를 최신 상태로 유지하는 것이 좋습니다.

`autovacuum_analyze_scale_factor`매개변수는`autovacuum_analyze_threshold`, `autovacuum_analyze_cost_limit, and autovacuum_naptime` 매개변수와 함께 작동합니다. 최적의 설정은 데이터베이스의 특정 요구 사항과 테이블 크기, 업데이트 빈도에 따라 달라집니다. 이 파라미터에 대한 자세한 내용은 PostgreSQL 환경용 Amazon RDS의 자동 진공 이해하기 AWS 블로그 게시물을 참조하십시오.

autovacuum_analyze_threshold

`autovacuum_vacuum_threshold`파라미터는 다음과 비슷합니다. `autovacuum_analyze_threshold` autovacuum이 테이블을 분석하기 전에 테이블에서 발생해야 하는 최소 튜플 삽입, 업데이트 또는 삭제 횟수를 제어합니다. 이 설정은 작업 빈도가 높지 않은 테이블에서 불필요한 진공 청소를 방지하는 데 유용할 수 있습니다. 기본값은 50이며, 이는 PostgreSQL 엔진 기본값이며, PostgreSQL용 Amazon RDS와 Aurora PostgreSQL과 호환되는 경우 모두 PostgreSQL 엔진 기본값입니다.

예를 들어, 100,000개의 행으로 구성된 테이블이 있고 기본값을 50으로 유지한다고 가정해 보겠습니다. `autovacuum_analyze_threshold` 테이블에 삽입, 업데이트 또는 삭제가 49건만 접수되면 autovacuum은 테이블을 분석하지 않습니다. 테이블에 50개 이상의 삽입, 업데이트 또는 삭제가 접수되면 autovacuum은 테이블을 분석하여 의 값에 테이블 행 수를 `autovacuum_analyze_scale_factor` 곱한 값을 제어 요인으로 유지합니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `autovacuum_analyze_threshold` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_analyze_threshold on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_analyze_threshold on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
```

```
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 50회 작업

이 매개변수는 autovacuum_analyze_scale_factor 매개변수와 함께 작동하므로 autovacuum을 구성할 때는 두 설정을 모두 고려해야 합니다.

데이터 액세스 방법 및 구성 방법 등 정보에 입각한 결정을 내리기 위해서는 쿼리 플래너가 통계를 수집해야 합니다. autovacuum_analyze_threshold너무 높게 설정하면 통계가 오래되어 성능이 저하될 수 있습니다. 자동 진공 프로세스의 성능을 모니터링하고 필요에 따라 설정을 조정하는 것이 좋습니다.

이 파라미터에 대한 자세한 내용은 PostgreSQL 환경용 Amazon RDS의 자동 진공 이해하기 AWS 블로그 게시물을 참조하십시오.

autovacuum_vacuum_cost_limit

autovacuum_vacuum_cost_limit파라미터는 자동진공 작업자가 소비할 수 있는 CPU 및 I/O 리소스의 양을 제어합니다.

autovacuum 프로세스의 리소스 사용을 제한하면 동일한 시스템에서 실행되는 다른 쿼리의 성능에 영향을 미칠 수 있는 CPU 또는 디스크 I/O를 너무 많이 소비하는 것을 방지할 수 있습니다. 파라미터는 비용 한도를 지정합니다. 비용 한도는 작업자가 일시 중지하고 한도 미만인지 확인하기 전에 작업자가 수행할 수 있는 작업 단위입니다. 예를 들어, 매개변수를 2,000으로 설정하면 작업자는 일시 중지하기 전에 2,000단위의 작업을 처리할 수 있습니다.

PostgreSQL 세션에서 SET 명령을 사용하거나 명령을 사용하여 autovacuum_vacuum_cost_limit 매개 변수를 설정할 수 있습니다. AWS CLI

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 autovacuum_vacuum_cost_limit 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify autovacuum_vacuum_cost_limit on a DB parameter group
aws rds modify-db-parameter-group \
```

```
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_cost_limit,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_vacuum_cost_limit on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_cost_limit,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 `ApplyMethod=immediate` 적용됨)

기본값: 작업 `GREATEST({log(DBInstanceClassMemory/21474836480)*600},200)` 단위

값을 `autovacuum_vacuum_cost_limit` 너무 높게 설정하면 autovacuum 프로세스가 너무 많은 리소스를 소비하여 다른 쿼리 속도가 느려질 수 있습니다. 이 값을 너무 낮게 설정하면 autovacuum 프로세스에서 충분한 공간을 확보하지 못해 시간이 지남에 따라 테이블이 더 커질 수 있습니다. 시스템에 맞는 적절한 저울을 찾는 것이 중요합니다.

이 매개변수는 자동 진공 프로세스에만 영향을 주고 수동 `VACUUM` 명령에는 영향을 주지 않습니다. 또한 이 값은 의 autovacuum 프로세스에만 적용되며 대상 `VACUUM` 프로세스에는 적용되지 않습니다. `ANALYZE`

로깅 매개변수 조정

PostgreSQL에서 로깅 매개변수를 조정하면 시스템에 부담을 주는 대용량 로그를 생성하지 않고도 올바른 정보를 수집할 수 있습니다.

로그 세부 정보와 시스템 성능 및 디스크 사용량의 균형을 맞추려면 로깅 매개 변수를 최적화하는 것이 중요합니다. 시스템 성능 및 디스크 사용에 미치는 영향을 최소화하면서 로그에서 적절한 수준의 세부 정보를 캡처하고, 문제를 진단하고, 사고를 효과적으로 조사하도록 다음 로깅 매개 변수를 사용자 지정할 수 있습니다.

- rds.force_autovacuum_logging
- rds.force_admin_logging_level
- 로그_기간
- 로그_최소_기간_명세서
- 로그_오류_상세도
- 로그_스테이트먼트
- 로그_명세서_통계
- 로그_최소_오류_명령문
- 로그_최소 메시지
- 로그_임시 파일
- 로그_연결
- 로그_연결 끊기

이러한 매개변수에 대해서는 다음 섹션에서 자세히 설명합니다.

> **Warning**
>
> 이러한 매개 변수에 가장 적합한 설정은 조직의 정책 및 규정 준수 요구 사항에 따라 달라집니다. 그러나 로깅 매개 변수를 활성화하면 로그와 메시지 수가 많아져 스토리지를 소모하고 특히 사용량이 많은 데이터베이스의 경우 성능에 영향을 미칠 수 있습니다. 이러한 매개 변수는 주의해서 사용하는 것이 좋습니다. 예를 들어 성능이 느린 SQL 문으로 인한 문제의 범위를 좁히기 위해 임시로 활성화하고 모니터링 기간이 끝나면 비활성화하도록 결정할 수 있습니다.

rds.force_autovacuum_logging

rds.force_autovacuum_logging파라미터 (PostgreSQL용 Amazon RDS에서만 사용 가능) 는 자동 진공 작업이 서버 로그에 기록되는지 여부를 제어합니다. 값은disabled,,,,,debug5,debug4,debug3,,debug2,debug1, infonotice, warning 입니다. error log fatal panic 기본값은 warning입니다.

rds.force_autovacuum_logging활성화하면 자동 진공 프로세스의 모든 작업 (예: 프로세스 시작 시간, 종료 시간, 진공 청소기로 청소한 행 수) 이 기록됩니다. 이는 autovacuum 성능 문제를 디버깅하거나 해결하는 데 유용합니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 rds.force_autovacuum_logging 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify rds.force_autovacuum_logging on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_autovacuum_logging,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify rds.force_autovacuum_logging on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_autovacuum_logging,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: warning

예

rds.force_autovacuum_logging파라미터를 사용하여 쓰기 속도가 매우 높은 테이블에서 autovacuum 성능을 분석할 수 있습니다. 예를 들어, 테이블이 초당 많은 수의 쓰기 및 삭제 작업을 수신하고 성능이 저하되는 경우 파라미터를 사용하여 각 autovacuum 실행의 시작 및 종료 시간을 기록하고 청소된 행 수를 확인할 수 있습니다. 이를 통해 autovacuum 실행 빈도, 실행 시간, 진공 청소기로 청소하는 행 수에 대한 중요한 정보를 얻을 수 있습니다. 그런 다음 이 정보를 사용하여 autovacuum_vacuum_scale_factorautovacuum_vacuum_threshold, 등의 autovacuum 설정을 미세 조정하고 성능을 최적화할 수 있습니다. autovacuum_naptime

rds.force_admin_logging_level

rds.force_admin_logging_level파라미터 (Amazon RDS for PostgreSQL에서만 사용 가능) 는 진공 청소, 분석 및 재인덱싱과 같은 관리 작업을 통해 생성되는 로그의 세부 정보 수준을 제어합니다. debug5,,,,,,,debug4, debug3 debug2 debug1 log info noticewarning, error 및 (기본값) 값을 허용합니다. log fatal off 최적의 설정은 사용 사례에 따라 다릅니다. 예를 들어 문제를 해결하는 경우 파라미터를 디버그 수준으로 설정하는 것이 좋습니다. 그렇지 않으면 loginfo, 또는 warning 설정을 사용할 수 있습니다.

rds.force_admin_logging_level로 설정하면 시작 및 종료 시간debug1, 처리된 행 수, 프로세스 중에 발생하는 오류 또는 경고와 같은 색인 재지정 작업에 대한 세부 정보를 기록할 수 있습니다. 이를 통해 색인 재지정 프로세스의 수행 방식에 대한 중요한 정보를 얻을 수 있으며 발생하는 모든 문제를 해결하는 데 도움이 될 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 rds.force_admin_logging_level 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify rds.force_admin_logging_level on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_admin_logging_level,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify rds.force_admin_logging_level on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_admin_logging_level,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: off

예

를 rds.force_admin_logging_level 사용하여 대규모 데이터베이스의 여러 테이블에 대한 관리 작업의 성능을 모니터링하고 분석할 수 있습니다. 예를 들어, 테이블이 많은 대규모 데이터베이스에서 정기적으로 진공 청소 및 분석 작업을 실행하여 이러한 테이블의 성능을 최적화하려고 한다고 가정해

보겠습니다. rds.force_admin_logging_level매개변수를 info 또는 log 로 설정하면 각 작업의 시작 및 종료 시간과 영향을 받은 테이블을 기록할 수 있습니다. 이 정보를 사용하여 여러 테이블에 걸친 관리 작업의 성능을 추적하고 더 빈번하거나 더 엄격한 유지 관리가 필요할 수 있는 테이블을 식별할 수 있습니다.

일부 로깅 수준은 특히 사용량이 많은 데이터베이스가 있는 경우 많은 수의 로그 파일과 메시지를 생성하여 디스크 공간을 빠르게 채울 수 있습니다. 이 매개 변수를 주의해서 사용하고 모니터링 기간이 끝나면 이 매개 변수를 끄는 것이 좋습니다.

log_duration

log_duration파라미터는 각 쿼리의 지속 시간 (즉, 실행에 걸리는 시간) 을 쿼리와 함께 기록할지 여부를 제어합니다. 이 매개 변수를 로 on 설정하면 각 쿼리를 실행하는 데 걸리는 시간이 쿼리 텍스트와 함께 로그 출력에 포함됩니다. 시간은 밀리초 단위로 측정됩니다.

log_duration파라미터의 주요 사용 사례는 성능 조정 및 문제 해결을 돕는 것입니다. 각 쿼리의 지속 시간을 기록하면 실행 시간이 가장 오래 걸리는 쿼리를 식별한 다음 해당 쿼리를 최적화하는 데 집중할 수 있습니다. 이를 통해 성능 병목 현상을 식별 및 수정하고 데이터베이스의 전반적인 성능을 개선할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_duration 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_duration on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_duration,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_duration on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_duration,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: off

예

특정 쿼리 또는 쿼리 집합으로 인해 성능 문제가 발생한다고 의심되는 경우 이 매개 변수를 사용할 수 있습니다. log_duration매개 변수를 활성화하고 로그 출력을 검사하면 실행 시간이 가장 오래 걸리는 쿼리를 확인한 다음 인덱스 최적화, 새 인덱스 추가 또는 쿼리 재작성과 같은 적절한 조치를 취할 수 있습니다.

활성화하면 로그 log_duration 출력량이 증가할 수 있습니다. 저장소가 가득 차거나 로그를 읽기 어려워지지 않도록 필요할 때만 사용하고 표준 작업 중에는 끄는 것이 좋습니다.

log_min_duration_statement

log_min_duration_statement파라미터는 SQL 문이 로깅되기 전에 실행되는 최소 시간 (밀리초)을 제어합니다.

이 매개 변수를 사용하면 성능 문제를 일으킬 수 있는 장기 실행 쿼리를 식별할 수 있습니다. 임계값 (특정 워크로드에 비해 너무 긴 것으로 간주되는 런타임) 으로 설정하여 해당 임계값을 초과하는 쿼리를 캡처하고 잠재적인 성능 병목 현상을 식별할 수 있습니다. 사용 사례 예시는 이 가이드 뒷부분의 로깅 매개변수를 사용하여 바인드 변수 캡처를 참조하십시오.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_min_duration_statement 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_min_duration_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_ min_duration_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: 1 (비활성화됨, PostgreSQL 엔진 기본값임)

예

다음 명령은 실행하는 데 100밀리초 이상 걸리는 모든 명령문을 기록합니다.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=100,ApplyMethod=immediate"
```

log_error_verbosity

`log_error_verbosity`파라미터는 오류 수준 이상으로 기록된 오류 및 메시지의 로그 출력에 포함되는 세부 정보 수준을 제어합니다. 이 매개변수는 `tersedefault`, 또는 세 가지 값 중 하나를 사용할 수 `verbose` 있습니다.

- `terse`메시지 텍스트, 오류 수준, 오류가 발생한 파일 및 줄 번호만 포함합니다.
- `default`메시지 텍스트, 오류 수준, 파일 및 줄 번호, 오류 컨텍스트가 포함됩니다.
- `verbose`메시지 텍스트, 오류 수준, 파일 및 줄 번호, 오류 컨텍스트, 전체 오류 메시지를 포함합니다.

비프로덕션 환경에서의 문제 해결 및 디버깅에 대한 가장 자세한 정보를 `verbose` 얻으려면 매개 변수를 로 설정하십시오. 프로덕션 환경에서는 필수 정보만 제공하고 너무 많은 세부 정보로 로그 저장소를 채우지 않도록 설정하는 것이 좋습니다. `default` `terse`

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `log_error_verbosity` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_error_verbosity on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_error_verbosity,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_error_verbosity on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
```

```
  --parameters
 "ParameterName=log_error_verbosity,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: default

log_statement

log_statement 파라미터는 서버 로그에 기록되는 SQL 문을 제어합니다. 매개 변수는 다음 값 중 하나를 사용할 수 있습니다.

- none(기본값) 은 어떤 명령문도 기록하지 않습니다.
- ddl및 와 같은 CREATE TABLE 데이터 정의 언어 (DDL) 문만 기록합니다. ALTER TABLE
- mod, 및 같은 데이터 수정 명령문만 기록합니다. INSERT UPDATE DELETE
- all모든 SQL 문을 기록합니다.

log_statement파라미터를 사용하면 사용 사례와 관련된 특정 유형의 명령문만 문서화하여 로그에 기록되는 정보의 양을 제어할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_statement 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본 값: none

예

프로덕션 환경에서는 DDL 문만 기록하고 데이터베이스 스키마의 변경 사항을 `log_statement` `ddl` 추적하도록 설정하는 것이 좋습니다. 개발 환경에서는 디버깅 및 문제 해결에 도움이 되도록 모든 명령문을 `all` 기록하도록 매개 변수를 로 설정하는 것이 좋습니다. 또 다른 사용 사례의 예는 이 가이드 뒷부분의 로깅 매개변수를 사용하여 바인드 변수 캡처를 참조하십시오.

활성화하면 로그 출력 볼륨이 증가할 `log_statement` 수 있으므로 필요할 때만 사용하고 저장소가 가득 차거나 로그를 읽기 어렵게 만들지 않도록 끄십시오.

시스템을 모니터링하고 이 매개 변수의 값을 조정하여 기록되는 정보의 양과 시스템의 스토리지 및 성능 간에 적절한 균형을 이루는 것이 좋습니다.

log_statement_stats

`log_statement_stats`파라미터는 SQL 문 실행과 관련된 통계를 명령문과 함께 기록할지 여부를 제어합니다. 이 매개 변수를 켜면 영향을 받는 행 수, 읽고 쓴 디스크 블록 수, 명령문 실행에 걸리는 시간 등의 통계가 로그 출력에 포함됩니다.

`log_statement_stats`매개 변수를 사용하여 개별 명령문의 성능 및 전체 워크로드에 대한 추가 정보를 수집할 수 있습니다. 명령문 통계를 로깅하면 쿼리 성능 및 리소스 사용 패턴을 식별하고 해당 정보를 사용하여 데이터베이스를 최적화하고 전반적인 성능을 개선할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `log_statement_stats` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_statement_stats on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement_stats,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_statement_stas on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement_stats,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: off (PostgreSQL 엔진 기본값), 파라미터 그룹에서 설정하려면 0 또는 1 (Boolean) 을 사용합니다.

예

를 log_statement_stats 사용하여 특정 쿼리의 동작을 분석하고, 쿼리가 CPU, 메모리, 디스크 I/O와 같은 리소스를 어떻게 사용하는지 확인하고, 쿼리를 최적화할 수 있는지 여부를 식별할 수 있습니다. 또한 이 매개 변수를 사용하여 특정 테이블을 자주 읽는지 (이는 특정 열에 인덱스를 만들어야 함을 의미할 수 있음) 또는 테이블을 너무 자주 스캔하는지 확인할 수 있습니다.

log_statement_stats활성화하면 로그 출력 볼륨이 증가할 수 있으므로 필요할 때만 사용하고 저장 공간이 가득 차거나 로그를 읽기 어렵게 만드는 일이 없도록 끄십시오.

log_min_error_statement

log_min_error_statement파라미터는 오류를 초래하는 SQL 문을 로깅할 것인지를 제어합니다. 값은debug5,,,debug4,debug3, debug2debug1,info,notice,warning,error, logfatal, 및 panic 입니다. 이러한 설정은 로그에 기록되는 정보의 양을 제어하므로 심각도가 낮은 메시지를 필터링할 수 있습니다. 이 매개 변수를 더 높은 심각도 수준으로 설정하여 로그 출력량을 줄이고 중요한 메시지를 더 쉽게 찾을 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_min_error_statement 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_min_error_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_error_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_min_error_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_error_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: `error` (PostgreSQL 엔진 기본값)

예

특정 문제를 해결하고 오류를 일으키는 SQL 문의 오류 메시지를 확인하려는 `log_min_error_statement` 경우 사용을 고려할 수 있습니다.

log_min_messages

`log_min_messages`파라미터는 로그에 기록되는 심각도 수준을 제어합니다. 파라미터를`debug5`,`debug4`, `debug3``debug2`,`debug1`,`info`, `notice` `warning` `error` `logfatal`, 또는 로 설정할 수 `panic` 있습니다. 이러한 설정은 로그에 기록되는 정보의 양을 제어하므로 심각도가 낮은 메시지를 필터링할 수 있습니다. 이 매개 변수를 더 높은 심각도 수준으로 설정하여 로그 출력량을 줄이고 중요한 메시지를 더 쉽게 찾을 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 `log_min_messages` 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_min_messages on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_messages,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_min_messages on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_messages,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 `ApplyMethod=immediate` 적용됨)

기본 값: `notice`

예

특정 문제를 해결하면서 모든 오류 메시지를 보려면 이 매개 변수를 오류 및 더 높은 수준의 심각도 문제만 `error` 기록하도록 설정할 수 있습니다. 시스템 성능을 모니터링하려는 경우 이 매개 변수를 설정하여 `info` 각 명령문의 기간 및 통계와 같은 자세한 정보를 볼 수 있습니다.

심각도를 더 높게 설정하면 log_min_messages 로그 양이 줄어듭니다. 특정 사용 사례, 확인하려는 로그의 크기, 보유한 디스크 공간에 따라 이 매개 변수를 조정하는 것이 좋습니다.

log_temp_files

log_temp_files파라미터는 임시 파일 이름 및 크기의 로깅을 제어합니다. 정렬, 해시, 임시 쿼리 결과 등의 목적으로 생성된 임시 파일에 적용됩니다. 이 매개 변수를 활성화하면 삭제 시 각 임시 파일에 대한 로그 항목 (파일 크기 (바이트) 이 생성됩니다. 모든 임시 파일 정보를 포괄적으로 로깅하려면 이 매개 변수를 0으로 설정하고, 해당 크기를 초과하는 파일을 로깅하려면 양수 값 (단위가 지정되지 않은 경우 킬로바이트) 으로 설정할 수 있습니다. 이는 성능 병목 현상이나 임시 저장소와 관련된 기타 문제를 식별하고 해결하는 데 유용할 수 있습니다. 기본적으로 임시 파일 로깅은 비활성화됩니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_temp_files 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_temp_files on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_temp_files,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_temp_files on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_temp_files,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: -1 (PostgreSQL 엔진 기본값)

예

시스템에서 임시 스토리지를 너무 많이 사용하고 있거나 임시 파일이 제대로 삭제되지 않는 것으로 의심되는 경우 이 매개 변수를 활성화할 수 있습니다. 로그 출력을 살펴보면 임시 파일을 생성하는 쿼리 또는 작업과 이러한 파일이 사용되는 방식을 확인할 수 있습니다.

일부 쿼리 또는 작업은 많은 수의 임시 파일을 생성하므로 활성화가 시스템의 전체 성능에 영향을 log_temp_files 미칠 수 있습니다.

log_connections

log_connections파라미터는 데이터베이스 연결이 로깅되는지 여부를 제어합니다. 이 매개 변수를 로 설정하면 로그에는 클라이언트의 IP 주소on, 사용자 이름, 데이터베이스 이름, 연결 날짜 및 시간과 같은 각 데이터베이스 연결 성공 관련 정보가 포함됩니다.

log_connections매개 변수를 사용하여 데이터베이스 연결을 모니터링하고 문제를 해결할 수 있습니다. 데이터베이스에 연결하는 사용자, 애플리케이션, 터미널 및 봇, 연결 위치, 연결 빈도를 확인할 수 있습니다. 이 정보는 연결 관련 문제를 식별 및 해결하거나 사용 패턴을 추적하는 데 유용할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_connections 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_connections on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_connections,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_connections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_connections,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: off (PostgreSQL 엔진 기본값)

예

데이터베이스에 대한 연결이 너무 많거나 너무 자주 연결하는 특정 사용자 또는 IP 주소가 성능에 영향을 미치는 것으로 의심되는 경우 이 매개 변수를 사용할 수 있습니다. log_connections매개 변수를 활성화하고 로그 출력을 검사하면 모든 연결의 수와 세부 정보를 볼 수 있습니다.

이 매개 변수를 활성화하기 전에 조직의 정책을 확인하고 IP 주소 및 사용자 이름 로깅이 보안에 미치는 영향을 고려하십시오.

log_disconnections

log_disconnections파라미터는 데이터베이스와의 연결 끊김 로깅을 제어합니다. 이 매개 변수를 로 설정하면 클라이언트의 IP 주소on, 사용자 이름, 데이터베이스 이름, 연결 해제 날짜 및 시간과 같은 각 세션 종료에 대한 정보가 기록됩니다.

log_disconnections매개 변수를 사용하여 데이터베이스 세션 종료를 모니터링하고 문제를 해결할 수 있습니다. 데이터베이스에서 연결이 끊긴 사용자, 애플리케이션, 터미널 및 봇을 확인할 수 있습니다. 언제, 왜 그런지 확인할 수 있습니다. 예를 들어 충돌이나 관리자가 일으킨 연결 끊김과 같은 예상치 못한 종료를 검토할 수 있습니다. 이 정보는 연결 끊김과 관련된 문제를 식별 및 해결하거나 사용 패턴을 추적하는 데 유용할 수 있습니다.

AWS CLI 구문

다음 명령은 특정 DB 파라미터 그룹에 log_disconnections 대해 변경됩니다. 이 변경 사항은 파라미터 그룹을 사용하는 모든 인스턴스 또는 클러스터에 적용됩니다.

```
# Modify log_disconnections on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_disconnections,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_disconnections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_disconnections,ParameterValue=<new_value>,ApplyMethod=immediate"
```

유형: 동적 (설정하면 변경 사항이 즉시 ApplyMethod=immediate 적용됨)

기본값: off (PostgreSQL 엔진 기본값)

예

데이터베이스에서 연결을 끊는 사용자가 너무 많거나 특정 사용자 또는 IP 주소의 연결이 너무 자주 끊긴다고 의심되는 log_disconnections 경우에 사용할 수 있습니다. log_disconnections파라미터를 활성화하고 로그 출력을 검사하면 연결이 끊기기 전에 누가, 언제, 어떤 오류가 발생했는지 여부를 포함하여 모든 연결이 끊긴 횟수와 세부 정보를 확인할 수 있습니다.

이 매개 변수를 활성화하기 전에 조직의 정책을 확인하고 IP 주소 및 사용자 이름 로깅이 보안에 미치는 영향을 고려하십시오.

로깅 매개변수를 사용하여 바인드 변수를 캡처합니다.

PostgreSQL에서 바인드 변수를 캡처하는 일반적인 사용 사례는 SQL 쿼리를 디버깅하고 성능을 조정하는 것입니다. 바인드 변수를 사용하면 쿼리를 실행할 때 쿼리에 데이터를 전달할 수 있습니다. 바인드 변수를 캡처하면 쿼리에 전달된 입력 데이터를 확인할 수 있으므로 데이터 또는 쿼리 성능과 관련된 문제를 식별하는 데 도움이 될 수 있습니다. 바인드 변수를 캡처하면 입력 데이터를 감사하고 잠재적인 보안 위험이나 악의적인 활동을 탐지하는 데도 도움이 될 수 있습니다.

PostgreSQL의 바인드 변수를 캡처하는 방법에는 여러 가지가 있습니다. 한 가지 방법은 및 매개 변수를 활성화하는 것입니다. debug_print_parse debug_print_rewritten 이로 인해 PostgreSQL은 바인딩된 변수와 함께 구문 분석되고 다시 작성된 버전의 SQL 문을 서버 로그로 전송합니다.

- debug_print_parse: 이 매개 변수를 활성화하면 들어오는 쿼리의 구문 분석 트리가 서버 로그에 인쇄됩니다. 이는 쿼리의 구조와 바인딩된 매개 변수의 값을 이해하는 데 유용할 수 있습니다.
- debug_print_rewritten: 이 매개 변수를 활성화하면 들어오는 쿼리의 다시 작성된 형식이 서버 로그에 인쇄됩니다. 이는 쿼리 플래너가 쿼리를 해석하는 방식과 바인딩된 매개 변수의 값을 이해하는 데 유용할 수 있습니다.

Amazon RDS와 Aurora에서 두 개의 추가 파라미터를 사용하여 PostgreSQL 데이터베이스의 바인드 변수를 캡처할 수 있습니다.

- log_min_duration_statement: 이 파라미터는 명령문이 로그인되기 전 최소 기간 (밀리초) 을 설정합니다. 명령문이 지정된 기간보다 오래 걸리는 경우 해당 명령문의 바인드 값이 로그 출력에 포함됩니다.
- log_statement: 이 매개 변수는 로깅되는 SQL 문을 제어합니다. 이 매개 변수를 로 all 설정하거나 바인딩하여 로그에 바인딩된 값을 포함하십시오. 로깅 수준을 높이면 성능에 영향을 미치므로 문제 해결 후 변경 내용을 되돌리는 것이 좋습니다.

쿼리 텍스트 및 바인딩된 값을 포함하여 서버에서 실행하는 모든 SQL 문에 대한 성능 통계를 제공하는 pg_stat_statements 확장을 사용할 수도 있습니다. 이 확장을 사용하면 PgAdmin 또는 유사한 도구를 사용하여 쿼리 성능을 모니터링하고 분석할 수 있습니다.

또 다른 옵션은 pg_bind_parameter_status() 함수를 사용하여 준비된 명령문에서 바인딩된 매개 변수의 값을 가져오거나 pg_get_parameter_status (paramname) 함수를 사용하여 특정 런타임 매개 변수의 상태 또는 값을 검색하는 것입니다.

또한 PGBadger와 같은 타사 도구를 사용하여 PostgreSQL 로그를 분석하고 추가 분석을 위해 바인드 변수 및 기타 정보를 추출할 수 있습니다.

복제 매개변수 조정

PostgreSQL에서는 물리적 파일 기반 복제 대신 논리적 복제를 사용하여 PostgreSQL 데이터베이스 간에 데이터 변경 내용을 복제할 수 있습니다. 논리적 복제는 WAL (미리 쓰기 로그) 을 사용하여 변경 사항을 캡처하고 선택한 테이블 또는 전체 데이터베이스의 복제를 지원합니다.

PostgreSQL용 Amazon RDS와 Aurora PostgreSQL과 호환되는 둘 다 논리적 복제를 지원하므로 여러 소스의 읽기 및 쓰기 트래픽을 처리할 수 있는 가용성과 확장성이 뛰어난 데이터베이스 아키텍처를 설정할 수 있습니다. 이러한 서비스는 오픈 소스 PostgreSQL 확장인 pglogical을 사용하여 논리적 복제를 구현합니다.

Aurora와 Amazon RDS에서 논리적 복제를 조정하는 것은 최적의 성능, 확장성 및 가용성을 달성하는 데 중요합니다. pglogical 확장의 파라미터를 조정하여 논리적 복제의 성능을 관리할 수 있습니다. 예를 들어, 다음을 수행할 수 있습니다.

- 작업자 프로세스 수를 늘리거나 메모리 할당을 조정하여 복제 성능을 개선하십시오.
- 원본 데이터베이스와 복제 데이터베이스 간의 동기화 빈도를 조정하여 복제 지연 위험을 줄이십시오.
- 작업자 프로세스의 메모리 및 CPU 할당을 조정하여 리소스 사용을 최적화합니다.
- 복제 프로세스가 원본 데이터베이스의 성능에 과도한 영향을 주지 않는지 확인하십시오.

Aurora 및 Amazon RDS에서 다음 파라미터를 사용하여 논리적 복제를 제어하고 구성할 수 있습니다.

- `max_replication_slots`서버에 생성할 수 있는 최대 복제 슬롯 수를 설정합니다. 복제 슬롯은 WAL 데이터를 복제본으로 전송하기 위한 복제 연결을 위한 명명된 영구 예약입니다.
- `max_wal_senders`동시에 연결된 WAL 발신자 프로세스의 최대 수를 설정합니다. WAL 발신자 프로세스는 주 서버에서 복제본으로 WAL을 스트리밍하는 데 사용됩니다.
- `wal_sender_timeout`WAL 발신자가 중단하고 다시 연결하기 전에 복제본의 응답을 기다리는 최대 시간 (밀리초) 을 설정합니다.
- `wal_receiver_timeout`복제본이 제한 시간이 초과되기 전에 기본 데이터베이스의 WAL 데이터를 기다리는 최대 시간 (밀리초) 을 설정합니다.
- `log_replication_commands`로 설정하면 복제 관련 SQL on 문을 실행합니다.

`rds.logical_replication`매개 변수를 활성화하면 (1로 설정) `wal_level` 매개 변수가 로 설정됩니다. 즉`logical`, 데이터베이스의 모든 변경 사항이 복제본에서 읽고 적용할 수 있는 형식으로 WAL

에 기록됩니다. 논리적 복제를 활성화하려면 이 설정이 필요합니다. 이 설정을 사용하면 SELECT 명령문을 복제할 수도 있습니다.

wal_level로 logical 설정하면 WAL, 즉 디스크에 기록되는 데이터의 양이 증가하여 시스템 성능에 영향을 미칠 수 있습니다. 논리적 복제를 활성화할 때는 사용 가능한 디스크 공간과 시스템 성능을 고려하는 것이 좋습니다.

예

백업 및 재해 복구를 위해 기본 데이터베이스의 데이터를 보조 데이터베이스로 복제하려고 합니다. 하지만 보조 데이터베이스는 읽기 작업의 양이 많기 때문에 데이터 무결성을 손상시키지 않으면서 복제 프로세스를 최대한 빠르고 효율적으로 수행하는 것이 좋습니다.

Amazon RDS 및 Aurora의 논리적 복제 기본값은 성능보다 일관성을 우선시하므로 이 사용 사례에는 최적이 아닐 수 있습니다. 논리적 복제 설정을 최적화하여 속도와 효율성을 높이려면 다음과 같이 파라미터를 사용자 지정할 수 있습니다.

- 향후 잠재적인 성장 및 복제 요구 사항을 수용하려면 10개 (Amazon RDS의 경우 기본값) 또는 20개 (Aurora의 경우 기본값) max_replication_slots 에서 30으로 늘리십시오.
- 복제 수요를 따라잡기에 충분한 WAL 발신자 프로세스가 있는지 확인하려면 10개 (기본값) max_wal_senders 에서 20개로 늘리십시오.
- 30초 (기본값) wal_sender_timeout 에서 15초로 줄여 사용하지 않는 WAL 발신자 프로세스를 더 빨리 종료하여 활성 복제에 필요한 리소스를 확보할 수 있습니다.
- 30초 (기본값) wal_receiver_timeout 에서 15초로 줄여 유휴 상태인 WAL 수신기 프로세스를 더 빨리 종료하여 활성 복제에 필요한 리소스를 확보할 수 있습니다.
- 복제 수요를 충족하기에 충분한 논리적 복제 작업자 프로세스가 있는지 확인하려면 4개 (기본값) max_logical_replication_workers 에서 8개로 늘리십시오.

이러한 최적화는 데이터 무결성과 보안을 유지하면서 더 빠르고 효율적인 데이터 복제를 제공합니다.

예를 들어, 재해가 발생하여 기본 데이터베이스를 사용할 수 없게 된 경우 최적화된 복제 프로세스로 인해 보조 데이터베이스에서 이미 최신 데이터를 사용할 수 있게 됩니다. 이렇게 하면 비즈니스 운영에서 중단 없이 중요한 서비스를 계속 제공할 수 있습니다.

모범 사례

대규모 워크로드로 논리적 복제를 튜닝하는 것은 데이터세트의 크기, 복제되는 테이블 수, 복제본 수, 가용 리소스 등 다양한 요인에 따라 달라지는 복잡한 작업일 수 있습니다. 대규모 워크로드로 논리적 복제를 튜닝하기 위한 몇 가지 일반적인 팁은 다음과 같습니다.

- 복제 지연을 모니터링하세요. 복제 지연은 기본 서버와 대기 서버 간의 시간 차이입니다. 복제 지연을 모니터링하면 잠재적 병목 현상을 식별하고 복제 성능을 개선하기 위한 조치를 취하는 데 도움이 될 수 있습니다. `pg_current_wal_lsn()` 함수를 사용하여 현재 복제 지연을 확인할 수 있습니다.
- WAL 설정을 조정합니다. `pg_logical`확장 프로그램은 WAL을 사용하여 기본 서버의 변경 사항을 대기 서버로 전송합니다. WAL 설정이 제대로 조정되지 않으면 복제가 느려지고 불안정해질 수 있습니다. 워크로드에 따라 `max_wal_senders` 및 `max_replication_slots` 매개변수를 적절한 값으로 설정해야 합니다.
- 인덱싱 전략을 세우세요. 주 서버에 적절한 인덱스를 만들면 논리적 복제의 성능을 향상시키고, 주 서버의 I/O를 줄이고, 시스템 부하를 줄이는 데 도움이 될 수 있습니다.
- 병렬 복제를 사용합니다. 병렬 복제를 사용하면 여러 병렬 작업자 프로세스가 데이터를 복제할 수 있으므로 복제 속도를 높일 수 있습니다. 이 기능은 PostgreSQL 12 이상에서 사용할 수 있습니다.

다음 단계

PostgreSQL용 Amazon RDS 또는 Aurora PostgreSQL 호환 데이터베이스의 메모리, 복제, 자동 진공 및 로깅 파라미터를 최적화한 후에는 다음 단계를 고려하여 데이터베이스의 성능을 더욱 향상시키십시오.

- 데이터베이스를 모니터링하십시오. 내장된 모니터링 도구 또는 타사 솔루션을 사용하여 시간 경과에 따른 데이터베이스 성능을 추적하세요. CPU 사용률, 디스크 I/O, 메모리 사용량, 쿼리 런타임과 같은 주요 성능 지표를 모니터링하여 잠재적 병목 현상과 개선이 필요한 영역을 식별할 수 있습니다.
- 파라미터를 지속적으로 튜닝합니다. 워크로드가 발전함에 따라 데이터베이스 매개변수를 계속 모니터링하고 조정하여 최적의 성능을 보장하십시오. 시스템 로그, 오류 메시지 및 성능 지표를 정기적으로 확인하여 새로운 조정 기회를 식별하십시오.
- 캐싱을 구현하십시오. 캐싱을 사용하면 데이터베이스에 도달하는 쿼리 수를 줄일 수 있습니다. Memcached 또는 Redis와 같은 도구를 사용하여 애플리케이션 수준에서 캐싱을 구현하거나 ElastiCache Amazon을 사용하여 데이터베이스에 인메모리 캐시를 제공할 수 있습니다.
- 쿼리를 최적화하십시오. 잘못 설계된 쿼리는 데이터베이스 성능에 큰 영향을 미칠 수 있습니다. 기타 쿼리 조정 도구를 사용하여 `EXPLAIN` 느린 쿼리를 식별하고 최적화하며 불필요한 쿼리를 제거하십시오.

이 지침을 따르면 PostgreSQL용 Aurora 또는 Amazon RDS 데이터베이스의 성능을 최적화하고 데이터베이스 성능 향상, 안정성 향상, 가동 중지 시간 감소, 보안 강화 및 비용 절감을 통해 애플리케이션 요구 사항을 충족할 수 있습니다. 워크로드에 맞게 구성 매개 변수를 최적화하면 데이터베이스가 효율적으로 실행되고 리소스를 효과적으로 사용하므로 성능이 향상되고 애플리케이션 응답성이 향상될 수 있습니다. 또한 매개 변수를 적절하게 구성하면 오류 및 취약성이 발생할 가능성을 줄여 안정성을 높이고 보안을 강화할 수 있습니다. 이는 유지 관리 및 가동 중지 시간 감소와 전반적인 사용자 경험 및 만족도 측면에서 비용 절감으로 이어질 수 있습니다.

리소스

- Amazon Aurora PostgreSQL 파라미터, 1부: 메모리 및 쿼리 계획 관리 (블로그 게시물)AWS
- Amazon Aurora PostgreSQL 파라미터, 2부: 복제, 보안 및 로깅 (블로그 게시물)AWS
- Amazon Aurora PostgreSQL 파라미터, 3부: 옵티마이저 파라미터 (블로그 게시물)AWS
- Amazon Aurora PostgreSQL 파라미터, 4부: ANSI 호환성 옵션 (블로그 게시물)AWS
- Amazon Aurora PostgreSQL을 사용한 작업 (설명서)AWS
- PostgreSQL용 Amazon AWS RDS를 사용한 작업 (설명서)
- Amazon RDS의 Performance Insights를 사용하여 DB 로드 모니터링 (AWS 설명서)
- Amazon CloudWatch 지표 사용 (AWS 설명서)
- pg_stats_statements (PostgreSQL 문서)

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
메모리 및 autovacuum 파라미터에 대한 정보가 업데이트되었습니다.	random_page_cost 매개변수의 설명을 업데이트하고, 메모리 및 autovacuum 매개변수의 기본값에 누락된 단위를 추가하고, max_connections 매개변수의 구문을 업데이트했습니다. AWS CLI	2024년 2월 27일
에 대한 정보가 업데이트되었습니다. autovacuum	autovacuum 기본 설정을 수정했습니다 (활성화됨).	2023년 12월 27일
에 대한 정보가 업데이트되었습니다. max_connections	이 파라미터 조정에 대한 새로운 지침으로 max_connections 섹션을 업데이트했습니다.	2023년 11월 15일
최초 게시	—	2023년 10월 31일

AWS 규범적 지침 용어집

다음은 규범적 지침에서 제공하는 AWS 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 마이그레이션하십시오.
- 리플랫포밍(리프트 앤드 리셰이프) - 애플리케이션을 클라우드로 이동하고 일정 수준의 최적화를 도입하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 오라클용 Amazon RDS (Amazon RDS) 로 마이그레이션합니다. AWS 클라우드
- 재구매(드롭 앤드 숍) - 일반적으로 기존 라이선스에서 SaaS 모델로 전환하여 다른 제품으로 전환합니다. 예: 고객 관계 관리 (CRM) 시스템을 Salesforce.com으로 마이그레이션하십시오.
- 리호스팅(리프트 앤드 시프트) - 애플리케이션을 변경하지 않고 클라우드로 이동하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 의 EC2 인스턴스에서 Oracle로 마이그레이션합니다. AWS 클라우드
- 재배치(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 다음으로 마이그레이션하십시오. AWS
- 유지(보관) - 소스 환경에 애플리케이션을 유지합니다. 대규모 리팩터링이 필요하고 해당 작업을 나중으로 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

ABAC

속성 기반 액세스 제어를 참조하십시오.

추상화된 서비스

관리형 서비스를 참조하십시오.

산

원자성, 일관성, 격리성, 내구성을 참조하십시오.

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 마이그레이션보다 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.

집계 함수

행 그룹에서 연산을 수행하고 그룹에 대한 단일 반환값을 계산하는 SQL 함수입니다. 집계 함수의 예로는 및 등이 `SUM` 있습니다. `MAX`

AI

인공 지능을 참조하십시오.

AIOps

인공 지능 운영을 참조하십시오.

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

시스템을 멀웨어로부터 보호하기 위해 승인된 애플리케이션만 사용할 수 있는 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 검색 및 분석 프로세스의 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 가이드를 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 AWS ABAC for를 참조하십시오.

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 격리되고 동일한 지역 내 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하는 별도의 위치. AWS 리전

AWS 클라우드 채택 프레임워크 (AWS CAF)

조직이 클라우드로 성공적으로 AWS 전환하기 위한 효율적이고 효과적인 계획을 개발하는 데 도움이 되는 지침 및 모범 사례 프레임워크입니다. AWS CAF는 지침을 관점이라고 하는 6가지 중점 영역, 즉 비즈니스, 사람, 거버넌스, 플랫폼, 보안, 운영으로 분류합니다. 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 조직이 성공적인 클라우드 채택을 준비할 수 있도록 인력 개발, 교육 및 커뮤니케이션에 대한 지침을 제공합니다. 자세한 내용은 AWS CAF 웹 사이트와 AWS CAF 백서를 참조하십시오.

AWS 워크로드 검증 프레임워크 (AWS WQF)

데이터베이스 마이그레이션 워크로드를 평가하고 마이그레이션 전략을 권장하며 작업 예상치를 제공하는 도구입니다. AWS WQF는 () 에 포함됩니다. AWS Schema Conversion Tool AWS SCT데이터베이스 스키마 및 코드 객체, 애플리케이션 코드, 종속성 및 성능 특성을 분석하고 평가 보고서를 제공합니다.

B

배드 봇

개인이나 조직을 방해하거나 피해를 입히려는 의도를 가진 봇입니다.

BCP

비즈니스 연속성 계획을 참조하십시오.

동작 그래프

리소스 동작과 시간 경과에 따른 상호 작용에 대한 통합된 대화형 뷰입니다. Amazon Detective에서 동작 그래프를 사용하여 실패한 로그온 시도, 의심스러운 API 호출 및 유사한 작업을 검사할 수 있습니다. 자세한 내용은 Detective 설명서의 Data in a behavior graph를 참조하십시오.

빅 엔디안 시스템

가장 중요한 바이트를 먼저 저장하는 시스템입니다. 엔디안도 참조하십시오.

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

서로 다르지만 동일한 환경을 두 개 만드는 배포 전략입니다. 현재 애플리케이션 버전을 한 환경 (파란색) 에서 실행하고 다른 환경 (녹색) 에서 새 애플리케이션 버전을 실행합니다. 이 전략을 사용하면 영향을 최소화하면서 신속하게 롤백할 수 있습니다.

bot

인터넷을 통해 자동화된 작업을 실행하고 사람의 활동이나 상호 작용을 시뮬레이션하는 소프트웨어 애플리케이션입니다. 인터넷에서 정보를 인덱싱하는 웹 크롤러와 같은 일부 봇은 유용하거나 유용합니다. 배드 봇으로 알려진 일부 다른 봇은 개인이나 조직을 방해하거나 피해를 입히기 위한 것입니다.

봇넷

멀웨어에감염되어 봇 허더 또는 봇 운영자로 알려진 단일 당사자의 통제 하에 있는 봇 네트워크. 봇넷은 봇과 그 영향을 확장하는 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 브랜치 정보 (문서) 를 참조하십시오. GitHub

브레이크 글래스 액세스

예외적인 상황에서 승인된 프로세스를 통해 사용자가 일반적으로 액세스 권한이 없는 데이터에 빠르게 액세스할 수 AWS 계정 있는 수단입니다. 자세한 내용은 Well-Architected AWS 지침의 브레이크 글래스 절차 구현 표시기를 참조하십시오.

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

클라우드 채택 프레임워크를 참조하십시오AWS .

카나리아 배포

최종 사용자에게 버전을 느리고 점진적으로 릴리스하는 것입니다. 확신이 들면 새 버전을 배포하고 현재 버전을 완전히 교체합니다.

CCoE

클라우드 센터 오브 엑설런스를 참조하십시오.

CDC

변경 데이터 캡처를 참조하십시오.

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 장애를 일으키는 이벤트를 발생시키는 행위 AWS Fault Injection Service (AWS FIS) 를 사용하여 AWS 워크로드에 스트레스를 주는 실험을 수행하고 응답을 평가할 수 있습니다.

CI/CD

지속적 통합 및 지속적 전달을 참조하십시오.

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.

클라이언트측 암호화

대상이 데이터를 AWS 서비스 수신하기 전에 데이터를 로컬로 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 기업 전략 블로그의 CCoE 게시물을 참조하십시오.

클라우드 컴퓨팅

원격 데이터 스토리지와 IoT 디바이스 관리에 일반적으로 사용되는 클라우드 기술 클라우드 컴퓨팅은 일반적으로 엣지 컴퓨팅 기술과 연결됩니다.

클라우드 운영 모델

IT 조직에서 하나 이상의 클라우드 환경을 구축, 성숙화 및 최적화하는 데 사용되는 운영 모델입니다. 자세한 내용은 클라우드 운영 모델 구축을 참조하십시오.

클라우드 채택 단계

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

- 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

Stephen Orban은 기업 전략 블로그의 클라우드 우선주의를 향한 여정 및 채택 단계에 대한 블로그 게시물에서 이러한 단계를 정의했습니다. AWS 클라우드 이들이 AWS 마이그레이션 전략과 어떤 관련이 있는지에 대한 자세한 내용은 마이그레이션 준비 가이드를 참조하십시오.

CMDB

구성 관리 데이터베이스를 참조하십시오.

코드 리포지토리

소스 코드와 설명서, 샘플, 스크립트 등의 기타 자산이 버전 관리 프로세스를 통해 저장되고 업데이트되는 위치입니다. 일반 클라우드 리포지토리에는 또는 이 포함됩니다 GitHub . AWS CodeCommit코드의 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 분야. 예를 들어 AWS Panorama 는 온프레미스 카메라 네트워크에 CV를 추가하는 디바이스를 제공하고, SageMaker Amazon은 CV용 이미지 처리 알고리즘을 제공합니다.

구성 드리프트

워크로드의 경우 구성이 예상 상태에서 변경됩니다. 이로 인해 워크로드가 규정을 준수하지 않게 될 수 있으며, 일반적으로 점진적이고 의도하지 않은 방식으로 진행됩니다.

구성 관리 데이터베이스(CMDB)

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

규정 준수 팩

AWS Config 규정 준수 및 보안 검사를 사용자 지정하기 위해 조합할 수 있는 규칙 및 수정 조치 모음입니다. YAML 템플릿을 사용하여 한 AWS 계정 및 지역 또는 조직 전체에 단일 엔티티로 적합성 팩을 배포할 수 있습니다. 자세한 내용은 설명서의 적합성 팩을 참조하십시오. AWS Config

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 이점을 참조하십시오. CD는 지속적 배포를 의미하기도 합니다. 자세한 내용은 지속적 전달(Continuous Delivery)과 지속적인 개발을 참조하십시오.

CV

컴퓨터 비전을 참조하십시오.

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 프레임워크의 보안 핵심 요소입니다. 자세한 내용은 데이터 분류를 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 거버넌스와 함께 분산되고 분산된 데이터 소유권을 제공하는 아키텍처 프레임워크입니다.

데이터 최소화

꼭 필요한 데이터만 수집하고 처리하는 원칙입니다. 에서 데이터 최소화를 실천하면 개인 정보 보호 위험, 비용 및 분석에 따른 탄소 발자국을 줄일 AWS 클라우드 수 있습니다.

데이터 경계

신뢰할 수 있는 ID만 예상 네트워크에서 신뢰할 수 있는 리소스에 액세스하도록 하는 데 도움이 되는 AWS 환경 내 일련의 예방 가드레일입니다. 자세한 내용은 데이터 경계 구축을 참조하십시오. 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 시스템. 데이터 웨어하우스에는 일반적으로 대량의 과거 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

데이터베이스 정의 언어를 참조하십시오.

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.

defense-in-depth

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 AWS여러 컨트롤을 추가하여 리소스를 보호하는 데 도움이 됩니다. 예를 들어 다단계 인증, 네트워크 세분화, 암호화를 결합한 defense-in-depth 접근 방식을 사용할 수 있습니다.

위임된 관리자

에서 AWS Organizations호환 가능한 서비스는 AWS 구성원 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 Organizations와 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

환경을 참조하십시오.

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 Implementing security controls on AWS의 Detective controls를 참조하십시오.

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.

치수 표

스타 스키마에서 팩트 테이블의 양적 데이터에 대한 데이터 속성을 포함하는 작은 테이블입니다. 차원 테이블 속성은 일반적으로 텍스트처럼 동작하는 텍스트 필드 또는 불연속형 숫자입니다. 이러한 속성은 일반적으로 쿼리 제한, 필터링 및 결과 집합 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)

재해로 인한 다운타임과 데이터 손실을 최소화하기 위해 사용하는 전략과 프로세스입니다. 자세한 내용은 워크로드의 재해 복구 AWS: AWS Well-Architected 프레임워크에서의 클라우드 복구를 참조하십시오.

DML

데이터베이스 조작 언어를 참조하십시오.

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 방법을 참조하십시오.

DR

재해 복구를 참조하십시오.

드리프트 감지

기준 구성으로부터의 편차 추적. 예를 들어 시스템 리소스의 편차를 감지하는 AWS CloudFormation 데 사용하거나 거버넌스 요구 사항 준수에 영향을 미칠 수 있는 착륙 지대의 변경 사항을 탐지하는 AWS Control Tower 데 사용할 수 있습니다.

DVSM

개발 가치 흐름 매핑을 참조하십시오.

E

EDA

탐색적 데이터 분석을 참조하십시오.

엣지 컴퓨팅

IoT 네트워크의 엣지에서 스마트 디바이스의 컴퓨팅 성능을 개선하는 기술 클라우드 컴퓨팅과 비교할 때 엣지 컴퓨팅은 통신 대기 시간을 줄이고 응답 시간을 개선할 수 있습니다.

암호화

사람이 읽을 수 있는 일반 텍스트 데이터를 암호문으로 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 엔드포인트를 참조하십시오.

엔드포인트 서비스

Virtual Private Cloud(VPC)에서 호스팅하여 다른 사용자와 공유할 수 있는 서비스입니다. 다른 주체 AWS 계정 또는 AWS Identity and Access Management (IAM) 보안 주체에 권한을 부여하여 엔드포인트 서비스를 생성하고 권한을 부여할 수 있습니다. AWS PrivateLink 이러한 계정 또는 보안 주체는 인터페이스 VPC 엔드포인트를 생성하여 엔드포인트 서비스에 비공개로 연결할 수 있습니

다. 자세한 내용은 Amazon Virtual Private Cloud(VPC) 설명서의 엔드포인트 서비스 생성을 참조하십시오.

ERP (전사적 자원 관리)

기업의 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 암호화를 참조하십시오.

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, 사고 대응 등이 포함됩니다. AWS 마이그레이션 전략의 에픽에 대한 자세한 내용은 프로그램 구현 가이드를 참조하십시오.

ERP

엔터프라이즈 리소스 계획을 참조하십시오.

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 스키마의](glossary.md) 중앙 테이블. 비즈니스 운영에 대한 정량적 데이터를 저장합니다. 일반적으로 팩트 테이블에는 측정값이 포함된 열과 차원 테이블의 외부 키가 포함된 열 등 두 가지 유형의 열이 포함됩니다.

빨리 실패하세요

빈번하고 점진적인 테스트를 통해 개발 라이프사이클을 단축하는 철학. 이는 애자일 접근 방식의 중요한 부분입니다.

장애 격리 경계

장애 영향을 제한하고 워크로드의 복원력을 개선하는 데 도움이 되는 가용 영역 AWS 리전, 컨트롤 플레인 또는 데이터 플레인과 같은 경계 AWS 클라우드자세한 내용은 AWS 장애 격리 경계를 참조하십시오.

기능 브랜치

[브랜치를](glossary.md) 참조하십시오.

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 그레디언트와 같은 다양한 기법을 통해 계산할 수 있는 수치 점수로 표현됩니다. 자세한 내용은 다음을AWS사용한 기계 학습 모델 해석 가능성을 참조하십시오.

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.

FGAC

[세분화된 액세스 제어를](glossary.md) 참조하십시오.

세분화된 액세스 제어(FGAC)

여러 조건을 사용하여 액세스 요청을 허용하거나 거부합니다.

플래시컷 마이그레이션

단계별 접근 방식 대신 변경 데이터 캡처를 통한 지속적인 데이터 복제를 통해 최단 시간에 데이터를 마이그레이션하는 데이터베이스 마이그레이션 방법입니다. 목표는 가동 중지 시간을 최소화하는 것입니다.

G

지리적 차단

지리적 제한을 참조하십시오.

지리적 제한(지리적 차단)

CloudFrontAmazon에서는 특정 국가의 사용자가 콘텐츠 배포에 액세스하지 못하도록 하는 옵션을 제공합니다. 허용 목록 또는 차단 목록을 사용하여 승인된 국가와 차단된 국가를 지정할 수 있습니다. 자세한 내용은 설명서의 콘텐츠의 지리적 배포 제한을 참조하십시오. CloudFront

Gitflow 워크플로

하위 환경과 상위 환경이 소스 코드 리포지토리의 서로 다른 브랜치를 사용하는 방식입니다. Gitflow 워크플로는 레거시로 간주되며 트렁크 기반 워크플로는 현대적이고 선호되는 접근 방식입니다.

브라운필드 전략

새로운 환경에서 기존 인프라의 부재 시스템 아키텍처에 대한 그린필드 전략을 채택할 때 브라운필드라고도 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, 이들은, Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector 및 사용자 지정 AWS Lambda 검사를 사용하여 구현됩니다.

H

하

고가용성을 확인하세요.

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 SCT를 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.

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 리플랫포밍 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. 긴급성 때문에 핫픽스는 일반적으로 일반적인 DevOps 릴리스 워크플로 외부에서 만들어집니다.

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

인프라를 코드로 보세요.

자격 증명 기반 정책

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

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 인터넷을 참조하십시오.

불변의 인프라

기존 인프라를 업데이트, 패치 또는 수정하는 대신 프로덕션 워크로드용 새 인프라를 배포하는 모델입니다. 변경 불가능한 인프라는 기본적으로 변경 가능한 인프라보다 더 일관되고 안정적이며 예측 가능합니다. 자세한 내용은 Well-Architected AWS 프레임워크의 변경 불가능한 인프라를 사용한 배포 모범 사례를 참조하십시오.

인바운드(수신) VPC

AWS 다중 계정 아키텍처에서 VPC는 애플리케이션 외부에서 네트워크 연결을 허용, 검사 및 라우팅합니다. AWS Security Reference Architecture에서는 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 것

이 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

Klaus Schwab이 연결성, 실시간 데이터, 자동화, 분석 및 AI/ML의 발전을 통한 제조 프로세스의 현대화를 지칭하기 위해 2016년 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 구축을 참조하십시오.

검사 VPC

AWS 다중 계정 아키텍처에서 VPC (동일하거나 AWS 리전다른), 인터넷 및 온프레미스 네트워크 간의 네트워크 트래픽 검사를 관리하는 중앙 집중식 VPC입니다. AWS Security Reference Architecture에서는 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 Machine learning model interpretability with AWS를 참조하십시오.

IoT

사물 인터넷을 참조하십시오.

IT 정보 라이브러리(TIL)

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

IT 서비스 관리(TSM)

조직의 IT 서비스 설계, 구현, 관리 및 지원과 관련된 활동 클라우드 운영을 ITSM 도구와 통합하는 방법에 대한 자세한 내용은 운영 통합 가이드를 참조하십시오.

ITIL

IT 정보 라이브러리를 참조하십시오.

ITSM

IT 서비스 관리를 참조하십시오.

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

Landing Zone은 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 설정을 참조하십시오.

대규모 마이그레이션

300대 이상의 서버 마이그레이션입니다.

LBAC

레이블 기반 액세스 제어를 참조하십시오.

최소 권한

작업을 수행하는 데 필요한 최소 권한을 부여하는 보안 모범 사례입니다. 자세한 내용은 IAM 설명서의 최소 권한 적용을 참조하십시오.

리프트 앤드 시프트

7 R을 참조하십시오.

리틀 엔디안 시스템

가장 덜 중요한 바이트를 먼저 저장하는 시스템입니다. 엔디안 참조.

하위 환경

환경 참조.

M

기계 학습(ML)

패턴 인식 및 학습에 알고리즘과 기법을 사용하는 인공 지능의 한 유형입니다. ML은 사물 인터넷(IoT) 데이터와 같은 기록된 데이터를 분석하고 학습하여 패턴을 기반으로 통계 모델을 생성합니다. 자세한 내용은 기계 학습을 참조하십시오.

기본 브랜치

브랜치 참조.

악성 코드

컴퓨터 보안 또는 개인 정보를 침해하도록 설계된 소프트웨어 멀웨어는 컴퓨터 시스템을 방해하거나, 민감한 정보를 유출하거나, 무단 액세스를 얻을 수 있습니다. 멀웨어의 예로는 바이러스, 웜, 랜섬웨어, 트로이 목마, 스파이웨어, 키로거 등이 있습니다.

매니지드 서비스

AWS 서비스 인프라 계층, 운영 체제 및 플랫폼을 AWS 운영하며 사용자는 엔드포인트에 액세스하여 데이터를 저장하고 검색합니다. 관리형 서비스의 예로는 아마존 심플 스토리지 서비스 (Amazon S3) 와 아마존 DynamoDB가 있습니다. 이러한 서비스를 추상화된 서비스라고도 합니다.

제조 실행 시스템 (MES)

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

MAP

Migration Acceleration 프로그램을 참조하십시오.

기구

도구를 만들고 도구 채택을 유도한 다음 결과를 검토하여 조정하는 전체 프로세스입니다. 메커니즘은 작동하면서 자체적으로 강화되고 개선되는 사이클입니다. 자세한 내용은 AWS Well-Architected 프레임워크에서의 메커니즘 구축을 참조하십시오.

멤버 계정

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

MES

제조 실행 시스템을 참조하십시오.

메시지 큐 텔레메트리 전송 (MQTT)

퍼블리시/구독 패턴을 기반으로 하는 리소스가 제한된 IoT 디바이스를 위한 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 서버리스 서비스를 사용하여 마이크로서비스 통합을 참조하십시오. AWS

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 마이크로서비스 구현을 참조하십시오. AWS

Migration Acceleration Program(MAP)

조직이 클라우드로 전환하기 위한 강력한 운영 기반을 구축하고 초기 마이그레이션 비용을 상쇄할 수 있도록 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 전략의 세 번째 단계입니다.

마이그레이션 팩토리

자동화되고 민첩한 접근 방식을 통해 워크로드 마이그레이션을 간소화하는 다기능 팀입니다. 마이그레이션 팩토리 팀에는 일반적으로 운영, 비즈니스 분석가 및 소유자, 마이그레이션 엔지니어, 개발자 및 스프린트에서 일하는 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 가이드와 마이그레이션 팩토리에 대한 설명을 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, 계정 등이 있습니다. AWS

마이그레이션 패턴

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

Migration Portfolio Assessment(MPA)

로 마이그레이션하기 위한 비즈니스 사례를 검증하기 위한 정보를 제공하는 온라인 도구입니다. AWS 클라우드 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 컨설턴트와 APN 파트너 AWS 컨설턴트에게 무료로 제공됩니다.

마이그레이션 준비 상태 평가(MRA)

CAF를 사용하여 조직의 클라우드 준비 상태에 대한 통찰력을 얻고, 강점과 약점을 파악하고, 식별된 격차를 해소하기 위한 실행 계획을 수립하는 프로세스입니다. AWS 자세한 내용은 마이그레이션 준비 가이드를 참조하십시오. MRA는 AWS 마이그레이션 전략의 첫 번째 단계입니다.

마이그레이션 전략

워크로드를 로 마이그레이션하는 데 사용된 접근 방식. AWS 클라우드자세한 내용은 이 용어집의 7R 항목 및 대규모 마이그레이션 가속화를 위한 조직 동원을 참조하십시오.

ML

기계 학습을 참조하십시오.

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 의 AWS 클라우드애플리케이션 현대화 전략을 참조하십시오.

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 에서 애플리케이션의 현대화 준비 상태 평가를 참조하십시오. AWS 클라우드

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 분해를 참조하십시오.

MPA

마이그레이션 포트폴리오 평가를 참조하십시오.

MQTT

메시지 큐 원격 분석 전송을 참조하십시오.

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 워크로드를 위해 기존 인프라를 업데이트하고 수정하는 모델입니다. 일관성, 안정성 및 예측 가능성을 개선하기 위해 AWS Well-Architected Framework는 변경 불가능한 인프라를 모범 사례로 사용할 것을 권장합니다.

O

OAC

원본 액세스 제어를 참조하십시오.

좋아요

원본 액세스 ID를 참조하십시오.

OCM

조직 변경 관리를 참조하십시오.

오프라인 마이그레이션

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

OI

운영 통합을 참조하십시오.

안녕하세요.

운영 수준 계약을 참조하십시오.

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

오픈 프로세스 커뮤니케이션 - 통합 아키텍처를 참조하십시오.

오픈 프로세스 커뮤니케이션 - 통합 아키텍처 (OPC-UA)

산업 machine-to-machine 자동화를 위한 (M2M) 통신 프로토콜. OPC-UA는 데이터 암호화, 인증 및 권한 부여 체계와 함께 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 검토 (ORR)

인시던트 및 발생 가능한 실패의 범위를 이해, 평가, 예방 또는 줄이는 데 도움이 되는 질문 및 관련 모범 사례로 구성된 체크리스트입니다. 자세한 내용은 Well-Architected AWS 프레임워크의 운영 준비 상태 검토 (ORR) 를 참조하십시오.

운영 기술 (OT)

물리적 환경과 함께 작동하여 산업 운영, 장비 및 인프라를 제어하는 하드웨어 및 소프트웨어 시스템. 제조 분야에서는 OT와 정보 기술 (IT) 시스템의 통합이 인더스트리 4.0 혁신의 핵심 초점입니다.

운영 통합(OI)

클라우드에서 운영을 현대화하는 프로세스로 준비 계획, 자동화 및 통합을 수반합니다. 자세한 내용은 운영 통합 가이드를 참조하십시오.

조직 트레일

이를 통해 AWS CloudTrail 생성되는 트레일은 조직 AWS 계정 내 모든 사용자의 모든 이벤트를 기록합니다. AWS Organizations이 트레일은 조직에 속한 각 AWS 계정 에 생성되고 각 계정의 활동을 추적합니다. 자세한 내용은 CloudTrail 설명서에서 조직을 위한 트레일 만들기를 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. 클라우드 채택 프로젝트에 필요한 변화 속도 때문에 AWS 마이그레이션 전략에서는 이 프레임워크를 사용자 가속화라고 합니다. 자세한 내용은 사용 가이드를 참조하십시오.

오리진 액세스 제어(OAC)

CloudFront에서는 Amazon Simple Storage Service (Amazon S3) 콘텐츠의 보안을 위해 액세스를 제한하는 향상된 옵션을 제공합니다. OAC는 모든 S3 버킷 AWS 리전, AWS KMS (SSE-KMS) 를 사용한 서버 측 암호화, S3 버킷에 대한 동적 및 요청을 모두 지원합니다. PUT DELETE

오리진 액세스 ID(OAI)

CloudFront에서는 Amazon S3 콘텐츠 보안을 위해 액세스를 제한하는 옵션입니다. OAI를 사용하면 Amazon S3가 인증할 수 있는 보안 주체를 CloudFront 생성합니다. 인증된 보안 주체는 특정 배

포를 통해서만 S3 버킷의 콘텐츠에 액세스할 수 있습니다. CloudFront 더 세분화되고 향상된 액세스 제어를 제공하는 OAC도 참조하십시오.

또는

운영 준비 상태 검토를 참조하십시오.

아니요

운영 기술을 참조하십시오.

아웃바운드(송신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 내에서 시작되는 네트워크 연결을 처리하는 VPC입니다. AWS Security Reference Architecture에서는 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 경계를 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 정보를 참조하십시오.

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

프로그래머블 로직 컨트롤러를 참조하십시오.

PLM

제품 라이프사이클 관리를 참조하십시오.

정책

권한을 정의 (ID 기반 정책 참조) 하거나, 액세스 조건을 지정 (리소스 기반 정책 참조) 하거나, 조직 내 모든 계정에 대한 최대 권한을 정의 AWS Organizations (서비스 제어 정책 참조) 할 수 있는 개체입니다.

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 스토어를 사용하면 마이크로서비스를 더 쉽게 구현하고 성능과 확장성을 높일 수 있습니다. 자세한 내용은 마이크로서비스에서 데이터 지속성 활성화를 참조하십시오.

포트폴리오 평가

마이그레이션을 계획하기 위해 애플리케이션 포트폴리오를 검색 및 분석하고 우선순위를 정하는 프로세스입니다. 자세한 내용은 마이그레이션 준비 상태 평가를 참조하십시오.

조건자

일반적으로 조항에 있는 `true` `false` OR를 반환하는 쿼리 조건입니다. `WHERE`

조건부 푸시다운

전송하기 전에 쿼리의 데이터를 필터링하는 데이터베이스 쿼리 최적화 기법입니다. 이렇게 하면 관계형 데이터베이스에서 검색하고 처리해야 하는 데이터의 양이 줄어들고 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 controls를 참조하십시오.

보안 주체

작업을 수행하고 리소스에 액세스할 수 AWS 있는 엔티티 이 엔티티는 일반적으로 IAM 역할의 루트 사용자 또는 사용자입니다. AWS 계정자세한 내용은 IAM 설명서의 역할 용어 및 개념의 보안 주체를 참조하십시오.

개인 정보 보호 중심 설계

전체 엔지니어링 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 작업을 참조하십시오.

사전 예방 제어

규정을 준수하지 않는 리소스의 배포를 방지하도록 설계된 보안 제어입니다. 이러한 컨트롤은 리소스를 프로비저닝하기 전에 리소스를 스캔합니다. 리소스가 컨트롤과 호환되지 않으면 프로비저닝되지 않습니다. 자세한 내용은 AWS Control Tower 설명서의 컨트롤 참조 안내서를 참조하고 보안 제어 구현의 사전 제어를 참조하십시오. AWS

제품 라이프사이클 관리 (PLM)

설계, 개발, 출시부터 성장 및 성숙도, 폐기 및 제거에 이르는 전체 라이프사이클에 걸쳐 제품에 대한 데이터 및 프로세스를 관리하는 것입니다.

프로덕션 환경

환경을 참조하십시오.

프로그래머블 로직 컨트롤러 (PLC)

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

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 (게시/구독)

마이크로서비스 간의 비동기 통신을 통해 확장성과 응답성을 개선할 수 있는 패턴입니다. 예를 들어 마이크로서비스 기반 MES에서 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 일련의 단계 (예: 지침).

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

RACI (책임, 책임, 상담, 정보 제공) 를 참조하십시오.

랜섬웨어

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

RASCI 매트릭스

책임, 책임, 상담, 정보 제공 (RACI) 을 참조하십시오.

RCAC

행 및 열 액세스 제어를 참조하십시오.

읽기 전용 복제본

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

재설계

7 R을 참조하십시오.

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

7 R을 참조하십시오.

리전

지리적 AWS 영역별 리소스 모음. AWS 리전 각각은 격리되어 있고 서로 독립적이므로 내결함성, 안정성 및 복원력을 제공합니다. 자세한 내용은 사용할 수 있는 AWS 리전 계정 지정을 참조하십시오.

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

7 R을 참조하십시오.

release

배포 프로세스에서 변경 사항을 프로덕션 환경으로 승격시키는 행위입니다.

고쳐 놓다

7 R을 참조하십시오.

리플랫포밍

7 R을 참조하십시오.

환매

7 R을 참조하십시오.

복원력

장애를 견디거나 장애를 복구할 수 있는 애플리케이션의 능력 고가용성 및 재해 복구는 복원력을 계획할 때 일반적으로 고려해야 할 사항입니다. AWS 클라우드자세한 내용은 AWS 클라우드 복원력을 참조하십시오.

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 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 Implementing security controls on AWS의 Responsive controls를 참조하십시오.

retain

7 R을 참조하십시오.

은퇴

7 R을 참조하십시오.

회전

공격자가 자격 증명에 액세스하는 것을 더 어렵게 만들기 위해 암호를 주기적으로 업데이트하는 프로세스입니다.

행 및 열 액세스 제어(RCAC)

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

RPO

복구 지점 목표를 참조하십시오.

RTO

복구 시간 목표를 참조하십시오.

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 (IdPs) 가 사용하는 개방형 표준입니다. 이 기능을 사용하면 페더레이션 싱글 사인온 (SSO) 이 가능하므로 조직의 모든 사용자를 위해 IAM에서 사용자를 생성하지 않고도 사용자가 AWS API 작업에 AWS Management Console 로그인하거나 API 작업을 호출할 수 있습니다.

SAML 2.0 기반 페더레이션에 대한 자세한 내용은 IAM 설명서의 SAML 2.0 기반 페더레이션 정보를 참조하십시오.

SCADA

감독 제어 및 데이터 수집을 참조하십시오.

SCP

서비스 제어 정책을 참조하십시오.

secret

에는 AWS Secrets Manager암호화된 형태로 저장하는 비밀번호나 사용자 자격 증명과 같은 기밀 또는 제한된 정보. 비밀 값과 해당 메타데이터로 구성됩니다. 비밀 값은 바이너리, 단일 문자열 또는 여러 문자열일 수 있습니다. 자세한 내용은 Secrets Manager 시크릿에는 무엇이 들어 있나요? 를 참조하십시오. Secrets Manager 설명서에서 확인할 수 있습니다.

보안 제어

위협 행위자가 보안 취약성을 악용하는 능력을 방지, 탐지 또는 감소시키는 기술적 또는 관리적 가드레일입니다. 보안 제어에는 예방적, 탐정적, 대응적, 사전 예방적 제어의 네 가지 기본 유형이 있습니다.

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 대응 자동화

보안 이벤트에 자동으로 대응하거나 보안 이벤트를 해결하도록 설계된 사전 정의되고 프로그래밍된 조치입니다. 이러한 자동화는 보안 모범 사례를 구현하는 데 도움이 되는 탐지 또는 대응형 보안 제어 역할을 합니다. AWS 자동 응답 조치의 예로는 VPC 보안 그룹 수정, Amazon EC2 인스턴스 패치, 자격 증명 교체 등이 있습니다.

서버 측 암호화

수신자에 의한 목적지의 데이터 암호화 AWS 서비스

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 참조하십시오.

서비스 엔드포인트

의 진입점 URL입니다 AWS 서비스. 엔드포인트를 사용하여 대상 서비스에 프로그래밍 방식으로 연결할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

서비스 수준에 관한 계약(SLA)

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

서비스 수준 표시기 (SLI)

오류율, 가용성 또는 처리량과 같은 서비스의 성능 측면을 측정하는 것입니다.

서비스 수준 목표 (SLO)

서비스 수준 지표로 측정되는 서비스 상태를 나타내는 대상 지표입니다.

공동 책임 모델

클라우드 보안 및 규정 준수에 AWS 대한 책임을 공유하는 것을 설명하는 모델입니다. AWS 클라우드의 보안을 책임지는 반면, 사용자는 클라우드에서의 보안을 담당합니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

시엠

보안 정보 및 이벤트 관리 시스템을 참조하십시오.

단일 장애 지점 (SPOF)

응용 프로그램의 중요한 단일 구성 요소에서 발생한 오류로 인해 시스템이 중단될 수 있습니다.

SLA

SLA (서비스 수준 계약) 를 참조하십시오.

SLI

서비스 수준 표시기 참조.

SLO

서비스 수준 목표를 참조하십시오.

split-and-seed 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 의 애플리케이션 현대화를 위한 단계별 접근 방식을 참조하십시오. AWS 클라우드

SPOF

단일 장애 지점 보기.

스타 스키마

하나의 큰 팩트 테이블을 사용하여 트랜잭션 또는 측정 데이터를 저장하고 하나 이상의 작은 차원 테이블을 사용하여 데이터 속성을 저장하는 데이터베이스 구성 구조입니다. 이 구조는 데이터 웨어하우스에서 사용하거나 비즈니스 인텔리전스 용도로 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 Fowler가 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 방법을 참조하십시오.

서브넷

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

감독 통제 및 데이터 수집 (SCADA)

제조 시 하드웨어와 소프트웨어를 사용하여 물리적 자산과 생산 작업을 모니터링하는 시스템입니다.

대칭 암호화

동일한 키를 사용하여 데이터를 암호화하고 복호화하는 암호화 알고리즘입니다.

합성 테스트

잠재적 문제를 감지하거나 성능을 모니터링하기 위해 사용자 상호 작용을 시뮬레이션하는 방식으로 시스템을 테스트합니다. Amazon CloudWatch Synthetics를 사용하여 이러한 테스트를 생성할 수 있습니다.

T

tags

리소스 구성을 위한 메타데이터 역할을 하는 키-값 쌍. AWS 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색 및 필터링할 수 있습니다. 자세한 내용은 AWS 리소스에 태그 지정을 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

환경을 참조하십시오.

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.

전송 게이트웨이

VPC와 온프레미스 네트워크를 상호 연결하는 데 사용할 수 있는 네트워크 전송 허브입니다. 자세한 내용은 AWS Transit Gateway 설명서의 트랜짓 게이트웨이란 무엇입니까? 를 참조하십시오.

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

조직 내 AWS Organizations 및 해당 계정에서 사용자를 대신하여 작업을 수행하도록 지정한 서비스에 권한 부여 신뢰할 수 있는 서비스는 필요할 때 각 계정에 서비스 연결 역할을 생성하여 관

리 작업을 수행합니다. 자세한 내용은 AWS Organizations 설명서의 다른 AWS 서비스와 AWS Organizations 함께 사용을 참조하십시오.

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 판만 들고 배불리 먹을 수 있는 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. 자세한 내용은 Quantifying uncertainty in deep learning systems 가이드를 참조하십시오.

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

환경을 보세요.

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.

윈도우 함수

현재 레코드와 어떤 식으로든 관련된 행 그룹에 대해 계산을 수행하는 SQL 함수입니다. 윈도우 함수는 이동 평균을 계산하거나 현재 행의 상대적 위치를 기반으로 행 값에 액세스하는 등의 작업을 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.

웜

한 번 쓰고, 많이 읽으세요.

WQF

AWS 워크로드 검증 프레임워크를 참조하십시오.

한 번 작성하고 여러 번 읽기 (WORM)

데이터를 한 번 쓰고 데이터가 삭제되거나 수정되지 않도록 하는 스토리지 모델입니다. 인증된 사용자는 필요한 만큼 데이터를 여러 번 읽을 수 있지만 변경할 수는 없습니다. 이 데이터 스토리지 인프라는 변경할 수 없는 것으로 간주됩니다.

Z

제로데이 익스플로잇

제로데이 취약점을 악용하는 공격 (일반적으로 멀웨어) 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.

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.