Terraform AWS 공급자 사용 모범 사례

AWS 권장 가이드

AWS 권장 가이드: Terraform AWS 공급자 사용 모범 사례

Table of Contents

Terraform AWS 공급자 사용 모범 사례

Michael Begin, Amazon Web Services의 Senior DevOps Consultant(AWS)

2025년 8월(문서 기록)

에서 Terraform을 사용하여 코드형 인프라(IaC)를 관리 AWS 하면 일관성, 보안 및 민첩성 향상과 같은 중요한 이점을 얻을 수 있습니다. 그러나 Terraform 구성의 크기와 복잡성이 증가함에 따라 위험을 방지하기 위해 모범 사례를 따르는 것이 중요합니다.

이 가이드에서는 HashiCorp의 Terraform AWS 공급자를 사용하기 위한 권장 모범 사례를 제공합니다. Terraform을 최적화하기 위한 적절한 버전 관리, 보안 제어, 원격 백엔드, 코드베이스 구조 및 커뮤니티 공급자를 안내합니다 AWS. 각 섹션에서는 이러한 모범 사례 적용의 세부 사항에 대해 자세히 설명합니다.

- 보안
- 백엔드
- 코드 기반 구조 및 조직
- AWS 공급자 버전 관리
- 커뮤니티 모듈

목표

이 가이드는 Terraform AWS Provider에 대한 운영 지식을 얻는 데 도움이 되며 보안, 신뢰성, 규정 준수 및 개발자 생산성에 대한 IaC 모범 사례를 따르면 달성할 수 있는 다음과 같은 비즈니스 목표를 해결합니다.

- Terraform 프로젝트 전반에서 인프라 코드 품질과 일관성을 개선합니다.
- 개발자 온보딩과 인프라 코드에 기여할 수 있는 능력을 가속화합니다.
- 더 빠른 인프라 변경을 통해 비즈니스 민첩성을 높입니다.
- 인프라 변경과 관련된 오류 및 가동 중지 시간을 줄입니다.
- IaC 모범 사례를 따라 인프라 비용을 최적화합니다.
- 모범 사례 구현을 통해 전반적인 보안 태세를 강화합니다.

대상 청중

이 가이드의 대상에는 IaC용 Terraform을 사용하는 팀을 감독하는 기술 책임자와 관리자가 포함됩니다 AWS. 다른 잠재적 독자로는 인프라 엔지니어, DevOps 엔지니어, 솔루션 아키텍트, Terraform을 적극적으로 사용하여 AWS 인프라를 관리하는 개발자 등이 있습니다.

이러한 모범 사례를 따르면 시간을 절약하고 이러한 역할에 대한 IaC의 이점을 활용할 수 있습니다.

개요

Terraform 공급자는 Terraform이 다양한 API와 상호 작용할 수 있도록 하는 플러그인입니다. 테라폼 AWS 제공자는 Terraform을 사용하여 코드형 AWS 인프라 (IaC) 를 관리하기 위한 공식 플러그인입니다. Terraform 구문을 AWS API 호출로 변환하여 리소스를 생성, 읽기, 업데이트 및 삭제합니다. AWS

AWS 공급자는 인증을 처리하고, Terraform 구문을 AWS API 호출로 변환하고, 리소스를 프로비저닝합니다. AWS Terraform `provider` 코드 블록을 사용하여 Terraform이 API와 상호 작용하는 데 사용하는 공급자 플러그인을 구성합니다. AWS 여러 AWS 공급자 블록을 구성하여 서로 다른 지역 및 지역의 리소스를 관리할 수 있습니다. AWS 계정

다음은 별칭이 있는 여러 AWS 공급자 블록을 사용하여 다른 지역 및 계정에 복제본이 있는 Amazon RDS (관계형 데이터베이스 서비스) 데이터베이스를 관리하는 Terraform 구성의 예입니다. 기본 공급자와 보조 공급자는 서로 다른 (IAM) 역할을 맡습니다. AWS Identity and Access Management

```
# Configure the primary AWS Provider
provider "aws" {
  region = "us-west-1"
  alias  = "primary"
}

# Configure a secondary AWS Provider for the replica Region and account
provider "aws" {
  region      = "us-east-1"
  alias       = "replica"
  assume_role {
    role_arn     = "arn:aws:iam::<replica-account-id>:role/<role-name>"
    session_name = "terraform-session"
  }
}

# Primary Amazon RDS database
resource "aws_db_instance" "primary" {
  provider = aws.primary

  # ... RDS instance configuration
}

# Read replica in a different Region and account
resource "aws_db_instance" "read_replica" {
  provider = aws.replica
```

```
  # ... RDS read replica configuration
  replicate_source_db = aws_db_instance.primary.id
}
```

이 예제에서는 다음이 적용됩니다.

- 첫 번째 `provider` 블록은 해당 `us-west-1` 리전의 기본 AWS 공급자를 별칭으로 구성합니다. `primary`
- 두 번째 `provider` 블록은 해당 지역의 보조 AWS 공급자를 별칭으로 구성합니다. `us-east-1` `replica` 이 공급자는 다른 지역 및 계정에 있는 기본 데이터베이스의 읽기 전용 복제본을 만드는 데 사용됩니다. `assume_role`블록은 복제본 계정에서 IAM 역할을 맡는 데 사용됩니다. 는 맡을 IAM 역할의 Amazon 리소스 이름 (ARN) 을 `role_arn` 지정하며 `session_name` Terraform 세션의 고유 식별자입니다.
- `aws_db_instance.primary`리소스는 `us-west-1` 리전의 `primary` 공급자를 사용하여 기본 Amazon RDS 데이터베이스를 생성합니다.
- `aws_db_instance.read_replica`리소스는 공급자를 사용하여 `us-east-1` 리전에 있는 기본 데이터베이스의 읽기 전용 복제본을 생성합니다. `replica` `replicate_source_db`속성은 `primary` 데이터베이스의 ID를 참조합니다.

보안 모범 사례

Terraform AWS 공급자를 안전하게 사용하려면 인증, 액세스 제어 및 보안을 올바르게 관리하는 것이 중요합니다. 이 섹션에서는 다음에 대한 모범 사례를 간략하게 설명합니다.

- 최소 권한 액세스를 위한 IAM 역할 및 권한
- AWS 계정 및 리소스에 대한 무단 액세스를 방지하는 데 도움이 되는 보안 인증 정보 보호
- 민감한 데이터를 보호하는 데 도움이 되는 원격 상태 암호화
- 잘못된 구성을 식별하기 위한 인프라 및 소스 코드 스캔
- 원격 상태 스토리지에 대한 액세스 제어
- 거버넌스 가드레일을 구현하기 위한 감시 정책 적용

이러한 모범 사례를 따르면 Terraform을 사용하여 AWS 인프라를 관리할 때 보안 태세를 강화하는 데 도움이 됩니다.

최소 권한 원칙을 따릅니다.

최소 권한은 사용자, 프로세스 또는 시스템이 의도한 기능을 수행하는 데 필요한 최소 권한만 부여하는 것을 가리키는 기본 보안 원칙입니다. 이는 액세스 제어의 핵심 개념이며 무단 액세스 및 잠재적 데이터 침해에 대한 예방 조치입니다.

최소 권한의 원칙은 Terraform이와 같은 클라우드 공급자에 대해 작업을 인증하고 실행하는 방법과 직접 관련이 있으므로이 섹션에서는 여러 번 강조됩니다 AWS.

Terraform을 사용하여 AWS 리소스를 프로비저닝하고 관리하는 경우 API 호출에 적절한 권한이 필요한 엔터티(사용자 또는 역할)를 대신합니다. 최소 권한을 따르지 않으면 주요 보안 위험이 발생합니다.

- Terraform에 필요한 것 이상의 과도한 권한이 있는 경우 의도하지 않은 잘못된 구성으로 인해 원치 않는 변경 또는 삭제가 발생할 수 있습니다.
- 지나치게 허용적인 액세스 권한 부여는 Terraform 상태 파일 또는 자격 증명이 손상된 경우 영향 범위를 늘립니다.
- 최소 권한을 따르지 않으면 필요한 최소 액세스 권한을 부여하기 위한 보안 모범 사례 및 규정 준수 요구 사항에 위배됩니다.

IAM 역할 사용

가능하면 IAM 사용자 대신 IAM 역할을 사용하여 Terraform AWS 공급자의 보안을 강화합니다. IAM 역할은 자동으로 교체되는 임시 보안 자격 증명을 제공하므로 장기 액세스 키를 관리할 필요가 없습니다. 또한 역할은 IAM 정책을 통해 정확한 액세스 제어를 제공합니다.

IAM 정책을 사용하여 최소 권한 액세스 권한 부여

IAM 정책을 신중하게 구성하여 역할과 사용자가 워크로드에 필요한 최소 권한 집합만 갖도록 합니다. 빈 정책으로 시작하고 허용된 서비스 및 작업을 반복적으로 추가합니다. 이를 수행하는 방법은 다음과 같습니다.

- IAM Access Analyzer를 활성화하여 정책을 평가하고 제거할 수 있는 미사용 권한을 강조 표시합니다.
- 정책을 수동으로 검토하여 역할의 의도한 책임에 필요하지 않은 기능을 제거합니다.
- IAM 정책 변수 및 태그를 사용하여 권한 관리를 간소화합니다.

잘 구성된 정책은 워크로드의 책임을 완수할 수 있는 충분한 액세스 권한만 부여합니다. 작업 수준에서 작업을 정의하고 특정 리소스의 필수 APIs에 대한 호출만 허용합니다.

이 모범 사례를 따르면 영향 범위가 줄어들고 업무 분리 및 최소 권한 액세스라는 기본 보안 원칙을 따릅니다. 열기를 시작하고 나중에 액세스를 제한하는 대신 필요에 따라 엄격하고 열린 액세스를 점진적으로 시작합니다.

로컬 인증을 위한 IAM 역할 수임

Terraform을 로컬에서 실행할 때는 정적 액세스 키를 구성하지 마십시오. 대신 IAM 역할을 사용하여 장기 자격 증명을 노출하지 않고 일시적으로 권한 있는 액세스 권한을 부여합니다.

먼저 필요한 최소 권한을 가진 IAM 역할을 생성하고 사용자 계정 또는 페더레이션 자격 증명에서 IAM 역할을 수임할 수 있는 신뢰 관계를 추가합니다. 이렇게 하면 역할의 임시 사용이 승인됩니다.

신뢰 관계 정책 예제:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/terraform-execution"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

그런 다음 aws sts assume-role AWS CLI 명령을 실행하여 역할에 대한 수명이 짧은 자격 증명을 검색합니다. 이러한 자격 증명은 일반적으로 1시간 동안 유효합니다.

AWS CLI 명령 예제:

```
aws sts assume-role --role-arn arn:aws:iam::111122223333:role/terraform-execution --
role-session-name terraform-session-example
```

명령의 출력에는 AWS다음을 인증하는 데 사용할 수 있는 액세스 키, 보안 키 및 세션 토큰이 포함되어 있습니다.

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:terraform-session-example",
        "Arn": "arn:aws:sts::111122223333:assumed-role/terraform-execution/terraform-
session-example"
    },
    "Credentials": {
        "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": " AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2024-03-15T00:05:07Z",
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE"
    }
}
```

AWS 공급자는 역할 수임을 자동으로 처리할 수도 있습니다.

IAM 역할을 수임하기 위한 공급자 구성 예제:

```
provider "aws" {
```

```
  assume_role {
    role_arn     = "arn:aws:iam::111122223333:role/terraform-execution"
    session_name = "terraform-session-example"
  }
}
```

이렇게 하면 Terraform 세션 기간 동안 엄격하게 승격된 권한이 부여됩니다. 임시 키는 최대 세션 기간 후에 자동으로 만료되므로 누출될 수 없습니다.

이 모범 사례의 주요 이점으로는 장기 액세스 키에 비해 보안 개선, 최소 권한을 위한 역할에 대한 세분화된 액세스 제어, 역할의 권한을 수정하여 액세스를 쉽게 취소할 수 있는 기능 등이 있습니다. 또한 IAM 역할을 사용하면 보안 암호를 스크립트 또는 디스크에 로컬로 직접 저장할 필요가 없으므로 팀 간에 Terraform 구성을 안전하게 공유할 수 있습니다.

Amazon EC2 인증에 IAM 역할 사용

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 Terraform을 실행하는 경우 장기 자격 증명을 로컬에 저장하지 마세요. 대신 IAM 역할 및 인스턴스 프로파일을 사용하여 최소 권한 권한을 자동으로 부여합니다.

먼저 최소 권한이 있는 IAM 역할을 생성하고 인스턴스 프로파일에 역할을 할당합니다. 인스턴스 프로파일을 사용하면 EC2 인스턴스가 역할에 정의된 권한을 상속할 수 있습니다. 그런 다음 해당 인스턴스 프로파일을 지정하여 인스턴스를 시작합니다. 인스턴스는 연결된 역할을 통해 인증됩니다.

Terraform 작업을 실행하기 전에 인스턴스 메타데이터에 역할이 있는지 확인하여 자격 증명이 성공적으로 상속되었는지 확인합니다.

```
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/
iam/security-credentials/
```

이 접근 방식은 영구 AWS 키를 인스턴스 내의 스크립트 또는 Terraform 구성으로 하드코딩하지 않도록 합니다. 임시 자격 증명은 인스턴스 역할 및 프로파일을 통해 Terraform에 투명하게 제공됩니다.

이 모범 사례의 주요 이점으로는 장기 자격 증명에 대한 보안 향상, 자격 증명 관리 오버헤드 감소, 개발, 테스트 및 프로덕션 환경 간의 일관성 등이 있습니다. IAM 역할 인증은 최소 권한 액세스를 적용하면서 EC2 인스턴스에서 Terraform 실행을 간소화합니다.

HCP Terraform 워크스페이스에 동적 자격 증명 사용

HCP Terraform은 HashiCorp에서 제공하는 관리형 서비스로, 팀이 Terraform을 사용하여 여러 프로젝트 및 환경에서 인프라를 프로비저닝하고 관리할 수 있도록 지원합니다. HCP Terraform에서 Terraform을 실행할 때 동적 자격 증명을 사용하여 AWS 인증을 간소화하고 보호합니다. Terraform은 IAM 역할 가정 없이 각 실행에서 임시 자격 증명을 자동으로 교환합니다.

더 쉬운 보안 암호 교체, 워크스페이스 전반의 중앙 집중식 자격 증명 관리, 최소 권한 권한, 하드코딩된 키 제거 등의 이점이 있습니다. 해시된 임시 키를 사용하면 수명이 긴 액세스 키에 비해 보안이 향상됩니다.

에서 IAM 역할 사용 AWS CodeBuild

에서 CodeBuild 프로젝트에 할당된 IAM 역할을 사용하여 빌드를 AWS CodeBuild실행합니다. 이렇게 하면 각 빌드가 장기 키를 사용하는 대신 역할에서 임시 자격 증명을 자동으로 상속할 수 있습니다.

HCP Terraform에서 원격으로 GitHub Actions 실행

HCP Terraform 워크스페이스에서 Terraform을 원격으로 실행하도록 GitHub Actions 워크플로를 구성합니다. GitHub 보안 암호 관리 대신 동적 보안 인증 및 원격 상태 잠금을 사용합니다.

OIDC에서 GitHub 작업 사용 및 AWS 자격 증명 작업 구성

OpenID Connect(OIDC) 표준을 사용하여 IAM을 통해 GitHub Actions 자격 증명을 페더레이션합니다. 자격 AWS 증명 구성 작업을 사용하면 장기 액세스 키 없이 GitHub 토큰을 임시 AWS 자격 증명으로 교환할 수 있습니다.

GitLab을 OIDC 및와 함께 사용 AWS CLI

OIDC 표준을 사용하여 임시 액세스를 위해 IAM을 통해 GitLab 자격 증명을 페더레이션합니다. OIDC를 사용하면 GitLab 내에서 장기 AWS 액세스 키를 직접 관리할 필요가 없습니다. 자격 증명은 just-in-time 교환되므로 보안이 향상됩니다. 또한 사용자는 IAM 역할의 권한에 따라 최소 권한 액세스 권한을 얻습니다.

레거시 자동화 도구와 함께 고유한 IAM 사용자 사용

IAM 역할 사용에 대한 기본 지원이 없는 자동화 도구 및 스크립트가 있는 경우 개별 IAM 사용자를 생성하여 프로그래밍 방식 액세스 권한을 부여할 수 있습니다. 최소 권한의 원칙은 여전히 적용됩니다. 정책 권한을 최소화하고 각 파이프라인 또는 스크립트에 대해 별도의 역할을 사용합니다. 최신 도구 또는 스크립트로 마이그레이션할 때 기본적으로 역할 지원을 시작하고 점진적으로 역할로 전환합니다.

> **Warning**
>
> IAM 사용자에게는 보안 위험이 있는 장기 자격 증명이 있습니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다.

Jenkins AWS 자격 증명 플러그인 사용

AWS Jenkins의 자격 증명 플러그인을 사용하여 중앙에서 자격 AWS 증명을 구성하고 빌드에 동적으로 주입합니다. 이렇게 하면 보안 암호를 소스 제어로 확인하지 않습니다.

최소 권한을 지속적으로 모니터링, 검증 및 최적화

시간이 지남에 따라 필요한 최소 정책을 초과할 수 있는 추가 권한이 부여될 수 있습니다. 액세스를 지속적으로 분석하여 불필요한 권한을 식별하고 제거합니다.

액세스 키 사용량을 지속적으로 모니터링

액세스 키 사용을 피할 수 없는 경우 IAM 자격 증명 보고서를 사용하여 90일이 지난 미사용 액세스 키를 찾고 사용자 계정과 시스템 역할 모두에서 비활성 키를 취소합니다. 관리자에게 활성 직원 및 시스템의 키 제거를 수동으로 확인하도록 알립니다.

키 사용을 모니터링하면 미사용 권한을 식별하고 제거할 수 있으므로 권한을 최적화하는 데 도움이 됩니다. 액세스 키 교체로이 모범 사례를 따르면 자격 증명 수명이 제한되고 최소 권한 액세스가 적용됩니다.

AWS 는 관리자에 대한 알림 및 알림을 설정하는 데 사용할 수 있는 여러 서비스와 기능을 제공합니다. 다음은 몇 가지 옵션입니다.

- AWS Config: AWS Config 규칙을 사용하여 IAM 액세스 키를 포함한 AWS 리소스의 구성 설정을 평가할 수 있습니다. 사용자 지정 규칙을 생성하여 특정 일수보다 오래된 미사용 액세스 키와 같은 특정 조건을 확인할 수 있습니다. 규칙을 위반하면는 문제 해결을 위한 평가를 시작하거나 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 보낼 AWS Config 수 있습니다.
- AWS Security Hub CSPM: Security Hub CSPM은 AWS 계정의 보안 태세에 대한 포괄적인 보기를 제공하며 미사용 또는 비활성 IAM 액세스 키를 포함한 잠재적 보안 문제를 감지하고 알리는 데 도움이 될 수 있습니다. Security Hub CSPM은 채팅 애플리케이션에서 Amazon EventBridge 및 Amazon SNS 또는 Amazon Q Developer와 통합하여 관리자에게 알림을 보낼 수 있습니다.

- AWS Lambda: Amazon CloudWatch Events 또는 AWS Config 규칙을 비롯한 다양한 이벤트에서 Lambda 함수를 호출할 수 있습니다. 사용자 지정 Lambda 함수를 작성하여 채팅 애플리케이션에서 Amazon SNS 또는 Amazon Q Developer와 같은 서비스를 사용하여 IAM 액세스 키 사용량을 평가하고, 추가 검사를 수행하고, 알림을 보낼 수 있습니다.

IAM 정책을 지속적으로 검증

IAM Access Analyzer를 사용하여 역할에 연결된 정책을 평가하고 부여된 미사용 서비스 또는 초과 작업을 식별합니다. 주기적 액세스 검토를 구현하여 정책이 현재 요구 사항과 일치하는지 수동으로 확인합니다.

기존 정책을 IAM Access Analyzer에서 생성된 정책과 비교하고 불필요한 권한을 제거합니다. 또한 사용자에게 보고서를 제공하고 유예 기간이 지나면 미사용 권한을 자동으로 취소해야 합니다. 이렇게 하면 최소 정책이 계속 적용됩니다.

더 이상 사용되지 않는 액세스를 선제적으로 자주 취소하면 위반 중에 위험할 수 있는 자격 증명이 최소화됩니다. 자동화는 지속 가능한 장기 자격 증명 위생 및 권한 최적화를 제공합니다. 이 모범 사례를 따르면 AWS 자격 증명 및 리소스에 대해 최소 권한을 사전에 적용하여 영향 범위를 제한할 수 있습니다.

보안 원격 상태 스토리지

원격 상태 스토리지란 Terraform이 실행 중인 시스템에 로컬로 저장하는 대신 Terraform 상태 파일을 원격으로 저장하는 것을 말합니다. 상태 파일은 Terraform에서 프로비저닝한 리소스와 메타데이터를 추적하기 때문에 매우 중요합니다.

원격 상태를 보호하지 못하면 상태 데이터 손실, 인프라 관리 불가, 의도하지 않은 리소스 삭제, 상태 파일에 존재할 수 있는 민감한 정보의 노출과 같은 심각한 문제가 발생할 수 있습니다. 이러한 이유로 프로덕션급 Terraform을 사용하려면 원격 상태 스토리지를 보호하는 것이 중요합니다.

암호화 및 액세스 제어 활성화

Amazon Simple Storage Service(Amazon S3) 서버 측 암호화(SSE)를 사용하여 저장 시 원격 상태를 암호화합니다.

협업 워크플로에 대한 직접 액세스 제한

- HCP Terraform 또는 Git 리포지토리 내의 CI/CD 파이프라인에서 공동 작업 워크플로를 구조화하여 직접 상태 액세스를 제한합니다.

- 풀 요청, 실행 승인, 정책 확인 및 알림을 사용하여 변경 사항을 조정합니다.

이러한 지침을 따르면 민감한 리소스 속성을 보호하고 팀원의 변경 사항과의 충돌을 방지하는 데 도움이 됩니다. 암호화 및 엄격한 액세스 보호는 공격 표면을 줄이는 데 도움이 되며 협업 워크플로는 생산성을 높입니다.

사용 AWS Secrets Manager

Terraform에는 보안 암호 값을 상태 파일에 일반 텍스트로 저장하는 많은 리소스와 데이터 소스가 있습니다. 보안 암호를 상태로 저장하지 말고 AWS Secrets Manager 대신 사용하세요.

민감한 값을 수동으로 암호화하려는 대신 민감한 상태 관리에 대한 Terraform의 기본 지원을 사용합니다. 민감한 값을 출력으로 내보낼 때는 값이 민감한 것으로 표시되어야 합니다.

인프라 및 소스 코드를 지속적으로 스캔

인프라와 소스 코드를 모두 지속적으로 검사하여 보안 태세를 강화하기 위해 노출된 자격 증명이나 잘못된 구성과 같은 위험이 있는지 확인합니다. 리소스를 재구성하거나 패치하여 조사 결과를 즉시 해결합니다.

동적 스캔에 AWS 서비스 사용

Amazon Inspector, AWS Security Hub CSPM, Amazon Detective, Amazon GuardDuty와 같은 AWS 네이티브 도구를 사용하여 계정 및 리전 간에 프로비저닝된 인프라를 모니터링합니다. Security Hub CSPM에서 반복 스캔을 예약하여 배포 및 구성 드리프트를 추적합니다. EC2 인스턴스, Lambda 함수, 컨테이너, S3 버킷 및 기타 리소스를 스캔합니다.

정적 분석 수행

Checkov와 같은 정적 분석기를 CI/CD 파이프라인에 직접 임베드하여 배포 전에 Terraform 구성 코드(HCL)를 스캔하고 위험을 선제적으로 식별합니다. 이렇게 하면 보안 검사가 개발 프로세스의 이전 지점(왼쪽 이동이라고 함)으로 이동하고 잘못 구성된 인프라를 방지할 수 있습니다.

프롬프트 문제 해결 보장

모든 스캔 결과에 대해 Terraform 구성을 업데이트하거나, 패치를 적용하거나, 리소스를 수동으로 적절하게 재구성하여 즉각적인 문제 해결을 보장합니다. 근본 원인을 해결하여 위험 수준을 낮춥니다.

인프라 스캔과 코드 스캔을 모두 사용하면 Terraform 구성, 프로비저닝된 리소스 및 애플리케이션 코드 전반에 걸쳐 계층화된 인사이트를 얻을 수 있습니다. 이를 통해 소프트웨어 개발 수명 주기(SDLC) 초기에 보안을 포함하면서 예방, 탐지 및 대응 제어를 통해 위험 및 규정 준수 범위를 극대화할 수 있습니다.

정책 검사 적용

HashiCorp Sentinel 정책과 같은 코드 프레임워크를 사용하여 Terraform을 사용한 인프라 프로비저닝을 위한 거버넌스 가드레일과 표준화된 템플릿을 제공합니다.

Sentinel 정책은 조직 표준 및 모범 사례에 맞게 Terraform 구성에 대한 요구 사항 또는 제한을 정의할 수 있습니다. 예를 들어 Sentinel 정책을 사용하여 다음을 수행할 수 있습니다.

- 모든 리소스에 태그가 필요합니다.
- 인스턴스 유형을 승인된 목록으로 제한합니다.
- 필수 변수를 적용합니다.
- 프로덕션 리소스의 파괴를 방지합니다.

정책 검사를 Terraform 구성 수명 주기에 포함하면 표준 및 아키텍처 지침을 선제적으로 적용할 수 있습니다. Sentinel은 승인되지 않은 관행을 방지하면서 개발을 가속화하는 데 도움이 되는 공유 정책 로직을 제공합니다.

백엔드 모범 사례

적절한 원격 백엔드를 사용하여 상태 파일을 저장하는 것은 공동 작업을 활성화하고, 잠금을 통해 상태 파일 무결성을 보장하고, 신뢰할 수 있는 백업 및 복구를 제공하고, CI/CD 워크플로와 통합하고, HCP Terraform과 같은 관리형 서비스에서 제공하는 고급 보안, 거버넌스 및 관리 기능을 활용하는 데 중요합니다.

Terraform은 Kubernetes, HashiCorp Consul 및 HTTP와 같은 다양한 백엔드 유형을 지원합니다. 그러나 이 가이드는 대부분의 AWS 사용자에게 최적의 백엔드 솔루션인 Amazon S3에 중점을 둡니다.

높은 내구성과 가용성을 제공하는 완전관리형 객체 스토리지 서비스인 Amazon S3는 Terraform 상태를 관리하기 위한 안전하고 확장 가능하며 저렴한 백엔드를 제공합니다 AWS. Amazon S3의 글로벌 공간 및 복원력은 대부분의 팀이 상태 스토리지를 자체 관리하여 달성할 수 있는 것보다 뛰어납니다. 또한 AWS 액세스 제어, 암호화 옵션, 버전 관리 기능 및 기타 서비스와 기본적으로 통합되므로 Amazon S3는 편리한 백엔드 선택입니다.

기본 대상 고객은 AWS 고객이므로이 가이드는 Kubernetes 또는 Consul과 같은 다른 솔루션에 대한 백엔드 지침을 제공하지 않습니다. 완전히에 있는 팀의 경우 AWS 클라우드 Amazon S3는 일반적으로 Kubernetes 또는 HashiCorp Consul 클러스터보다 이상적인 선택입니다. Amazon S3 상태 스토리지의 단순성, 복원력 및 긴밀한 AWS 통합은 AWS 모범 사례를 따르는 대부분의 사용자에게 최적의 기반을 제공합니다. 팀은 AWS 서비스의 내구성, 백업 보호 및 가용성을 활용하여 원격 Terraform 상태를 높은 복원력으로 유지할 수 있습니다.

이 섹션의 백엔드 권장 사항을 따르면 오류 또는 무단 수정의 영향을 제한하면서 더 협업적인 Terraform 코드 베이스가 됩니다. 팀은 잘 설계된 원격 백엔드를 구현하여 Terraform 워크플로를 최적화할 수 있습니다.

모범 사례:

- 원격 스토리지에 Amazon S3 사용
- 팀 협업 촉진
- 각 환경의 백엔드 분리
- 원격 상태 활동을 적극적으로 모니터링

원격 스토리지에 Amazon S3 사용

Amazon S3에 Terraform 상태를 원격으로 저장하고 Amazon DynamoDB를 사용하여 상태 잠금 및 일관성 검사를 구현하면 로컬 파일 스토리지보다 큰 이점을 얻을 수 있습니다. 원격 상태를 사용하면 팀 협업, 변경 추적, 백업 보호 및 원격 잠금을 통해 안전성을 높일 수 있습니다.

임시 로컬 스토리지 또는 자체 관리형 솔루션 대신 Amazon S3를 S3 Standard 스토리지 클래스(기본값)와 함께 사용하면 99.999999999%의 내구성과 99.99%의 가용성 보호 기능을 제공하여 우발적 상태 데이터 손실을 방지할 수 있습니다. Amazon S3 및 DynamoDB와 같은 AWS 관리형 서비스는 대부분의 조직이 스토리지를 자체 관리할 때 달성할 수 있는 수준을 초과하는 서비스 수준 계약(SLAs)을 제공합니다. 이러한 보호 기능을 사용하여 원격 백엔드에 액세스할 수 있습니다.

원격 상태 잠금 활성화

상태 잠금은 동시 쓰기 작업을 방지하도록 액세스를 제한하고 여러 사용자의 동시 수정으로 인한 오류를 줄입니다. Terraform은 Amazon S3 백엔드에 대해 두 가지 잠금 메커니즘을 지원합니다.

- Amazon S3 네이티브 상태 잠금(권장): Terraform 1.10.0부터 사용할 수 있으며, Amazon S3의 네이티브 잠금 기능을 사용합니다.
- DynamoDB 상태 잠금(사용되지 않음): 향후 Terraform 버전에서 제거될 레거시 접근 방식

```
terraform {
  backend "s3" {
    bucket       = "myorg-terraform-states"
    key          = "myapp/production/tfstate"
    region       = "us-east-1"
    use_lockfile = true
  }
}
```

마이그레이션 중에 이전 버전과의 호환성을 위해 Amazon S3 잠금과 DynamoDB 잠금을 동시에 구성할 수 있습니다. 그러나 DynamoDB 기반 잠금은 더 이상 사용되지 않으므로 Amazon S3 네이티브 잠금으로 마이그레이션하는 것이 좋습니다.

버전 관리 및 자동 백업 활성화

추가 보호를 위해 Amazon S3 백엔드 AWS Backup 에서를 사용하여 자동 버전 관리 및 백업을 활성화합니다. 버전 관리는 변경이 이루어질 때마다 모든 이전 버전의 상태를 유지합니다. 또한 원치 않는 변경 사항을 롤백하거나 사고로부터 복구해야 하는 경우 이전 작업 상태 스냅샷을 복원할 수 있습니다.

필요한 경우 이전 버전 복원

버전이 지정된 Amazon S3 상태 버킷을 사용하면 이전에 알려진 정상 상태 스냅샷을 복원하여 변경 사항을 쉽게 되돌릴 수 있습니다. 이렇게 하면 우발적인 변경으로부터 보호하고 추가 백업 기능을 제공할 수 있습니다.

HCP Terraform 사용

HCP Terraform은 자체 상태 스토리지를 구성하는 대신 완전 관리형 백엔드 대안을 제공합니다. HCP Terraform은 추가 기능을 잠금 해제하면서 상태의 보안 스토리지와 암호화를 자동으로 처리합니다.

HCP Terraform을 사용하면 상태가 기본적으로 원격으로 저장되므로 조직 전체에서 상태 공유 및 잠금이 가능합니다. 세부 정책 제어는 상태 액세스 및 변경을 제한하는 데 도움이 됩니다.

추가 기능에는 버전 관리 통합, 정책 가드레일, 워크플로 자동화, 변수 관리, SAML과의 Single Sign-On 통합이 포함됩니다. Sentinel 정책을 코드로 사용하여 거버넌스 제어를 구현할 수도 있습니다.

HCP Terraform은 서비스형 소프트웨어(SaaS) 플랫폼을 사용해야 하지만 많은 팀에서 보안, 액세스 제어, 자동화된 정책 확인 및 공동 작업 기능에 대한 이점을 통해 Amazon S3 또는 DynamoDB를 사용한 자체 관리 상태 스토리지보다 최적의 선택입니다.

GitHub 및 GitLab과 같은 서비스와 마이너 구성으로 쉽게 통합하면 더 나은 팀 워크플로를 위해 클라우드 및 SaaS 도구를 완전히 수용하는 사용자에게도 유용합니다.

팀 협업 촉진

원격 백엔드를 사용하여 Terraform 팀의 모든 구성원 간에 상태 데이터를 공유합니다. 이렇게 하면 전체 팀에 인프라 변경 사항에 대한 가시성을 제공하기 때문에 협업이 용이해집니다. 상태 기록 투명성과 결합된 공유 백엔드 프로토콜은 내부 변경 관리를 간소화합니다. 모든 인프라 변경은 확립된 파이프라인을 통과하여 기업 전반의 비즈니스 민첩성을 높입니다.

를 사용하여 책임 개선 AWS CloudTrail

Amazon S3 버킷 AWS CloudTrail 과 통합하여 상태 버킷에 대한 API 호출을 캡처합니다. CloudTrail 이벤트를 필터링하여 및 기타 관련 호출`PutObject`을 `DeleteObject`추적합니다.

CloudTrail 로그에는 상태 변경을 위해 각 API를 호출한 보안 주체의 AWS 자격 증명이 표시됩니다. 사용자의 자격 증명은 시스템 계정 또는 백엔드 스토리지와 상호 작용하는 팀원과 일치시킬 수 있습니다.

CloudTrail 로그를 Amazon S3 상태 버전 관리와 결합하여 인프라 변경 사항을 적용한 보안 주체와 연결합니다. 여러 개정을 분석하여 모든 업데이트를 시스템 계정 또는 담당 팀원에게 귀속할 수 있습니다.

의도하지 않거나 방해가 되는 변경이 발생하면 상태 버전 관리는 롤백 기능을 제공합니다. CloudTrail은 사용자에 대한 변경 사항을 추적하므로 예방 개선 사항에 대해 논의할 수 있습니다.

또한 IAM 권한을 적용하여 상태 버킷 액세스를 제한하는 것이 좋습니다. 전반적으로 S3 버전 관리 및 CloudTrail 모니터링은 인프라 변경 전반에 걸쳐 감사를 지원합니다. 팀은 Terraform 상태 기록에 대한 책임, 투명성 및 감사 기능을 개선합니다.

각 환경의 백엔드 분리

각 애플리케이션 환경에 대해 고유한 Terraform 백엔드를 사용합니다. 별도의 백엔드는 개발, 테스트 및 프로덕션 간에 상태를 격리합니다.

영향 범위 축소

상태를 격리하면 하위 환경의 변경 사항이 프로덕션 인프라에 영향을 주지 않도록 할 수 있습니다. 개발 및 테스트 환경의 사고 또는 실험은 영향이 제한적입니다.

프로덕션 액세스 제한

프로덕션 상태 백엔드에 대한 권한을 대부분의 사용자에 대한 읽기 전용 액세스로 잠급니다. 프로덕션 인프라를 CI/CD 파이프라인으로 수정하고 유리 역할을 해제할 수 있는 사람을 제한합니다.

액세스 제어 간소화

백엔드 수준에서 권한을 관리하면 환경 간 액세스 제어가 간소화됩니다. 각 애플리케이션 및 환경에 대해 고유한 S3 버킷을 사용하면 전체 백엔드 버킷에 광범위한 읽기 또는 쓰기 권한을 부여할 수 있습니다.

공유 워크스페이스 방지

Terraform 워크스페이스를 사용하여 환경 간에 상태를 분리할 수 있지만 고유한 백엔드는 더 강력한 격리를 제공합니다. 워크스페이스를 공유한 경우에도 사고는 여러 환경에 영향을 미칠 수 있습니다.

환경 백엔드를 완전히 격리된 상태로 유지하면 단일 장애 또는 위반의 영향을 최소화할 수 있습니다. 또한 별도의 백엔드는 환경의 민감도 수준에 맞게 액세스 제어를 조정합니다. 예를 들어 프로덕션 환경에 대한 쓰기 보호와 개발 및 테스트 환경에 대한 광범위한 액세스를 제공할 수 있습니다.

원격 상태 활동을 적극적으로 모니터링

잠재적 문제를 조기에 감지하려면 원격 상태 활동을 지속적으로 모니터링하는 것이 중요합니다. 비정상적인 잠금 해제, 변경 또는 액세스 시도가 있는지 확인합니다.

의심스러운 잠금 해제에 대한 알림 받기

대부분의 상태 변경은 CI/CD 파이프라인을 통해 실행되어야 합니다. 개발자 워크스테이션을 통해 직접 상태 잠금 해제가 발생하면 알림을 생성하여 무단 또는 테스트되지 않은 변경 사항을 알릴 수 있습니다.

액세스 시도 모니터링

상태 버킷의 인증 실패는 정찰 활동을 나타낼 수 있습니다. 여러 계정이 상태에 액세스하려고 하거나 비정상적인 IP 주소가 나타나 손상된 자격 증명을 나타내는지 확인합니다.

코드 기반 구조 및 조직에 대한 모범 사례

대규모 팀과 엔터프라이즈에서 Terraform 사용량이 증가함에 따라 적절한 코드 기반 구조와 조직이 중요합니다. 잘 설계된 코드 베이스를 사용하면 대규모로 협업하는 동시에 유지 관리를 강화할 수 있습니다.

이 섹션에서는 품질과 일관성을 지원하는 Terraform 모듈성, 명명 규칙, 설명서 및 코딩 표준에 대한 권장 사항을 제공합니다.

지침에는 환경 및 구성 요소별로 구성을 재사용 가능한 모듈로 나누고, 접두사 및 접미사를 사용하여 이름 지정 규칙을 설정하고, 모듈을 문서화하고, 입력 및 출력을 명확하게 설명하고, 자동 스타일 검사를 사용하여 일관된 형식 지정 규칙을 적용하는 것이 포함됩니다.

추가 모범 사례에서는 구조화된 계층 구조에서 모듈과 리소스를 논리적으로 구성하고, 문서에서 퍼블릭 및 프라이빗 모듈을 카탈로그화하고, 모듈에서 불필요한 구현 세부 정보를 추상화하여 사용을 간소화하는 방법을 다룹니다.

모듈성, 설명서, 표준 및 논리적 조직에 대한 코드 기반 구조 지침을 구현하면 팀 간의 광범위한 협업을 지원하는 동시에 Terraform을 조직 전체의 사용량 분산에 따라 유지 관리할 수 있습니다. 규칙과 표준을 적용하면 조각화된 코드 베이스의 복잡성을 피할 수 있습니다.

모범 사례:

- 표준 리포지토리 구조 구현
- 모듈화를 위한 구조
- 이름 지정 규칙 준수
- 첨부 파일 리소스 사용
- 기본 태그 사용
- Terraform 레지스트리 요구 사항 충족
- 권장 모듈 소스 사용
- 코딩 표준 준수

표준 리포지토리 구조 구현

다음 리포지토리 레이아웃을 구현하는 것이 좋습니다. 모듈 간에 이러한 일관성 사례를 표준화하면 검색 가능성, 투명성, 조직 및 신뢰성을 개선하는 동시에 여러 Terraform 구성에서 재사용할 수 있습니다.

- 루트 모듈 또는 디렉터리: Terraform 루트 모듈과 재사용 가능한 모듈의 기본 진입점이어야 하며 고유할 것으로 예상됩니다. 아키텍처가 더 복잡한 경우 중첩 모듈을 사용하여 간단한 추상화를 만들 수 있습니다. 이렇게 하면 물리적 객체가 아닌 아키텍처 측면에서 인프라를 설명할 수 있습니다.
- README: 루트 모듈과 중첩된 모듈에는 README 파일이 있어야 합니다. 이 파일의 이름은 이어야 합니다`README.md`. 여기에는 모듈에 대한 설명과 모듈의 용도가 포함되어야 합니다. 이 모듈을 다른 리소스와 함께 사용하는 예를 포함하려면 `examples` 디렉터리에 배치합니다. 모듈이 생성할 수 있는 인프라 리소스와 그 관계를 보여주는 다이어그램을 포함하는 것이 좋습니다. terraform-docs를 사용하여 모듈의 입력 또는 출력을 자동으로 생성합니다.
- main.tf: 기본 진입점입니다. 간단한 모듈의 경우이 파일에 모든 리소스가 생성될 수 있습니다. 복잡한 모듈의 경우 리소스 생성이 여러 파일에 분산될 수 있지만 중첩된 모듈 호출은 `main.tf` 파일에 있어야 합니다.
- variables.tf 및 outputs.tf: 이러한 파일에는 변수 및 출력에 대한 선언이 포함되어 있습니다. 모든 변수와 출력에는 목적을 설명하는 한 문장 또는 두 문장 설명이 있어야 합니다. 이러한 설명은 설명서에 사용됩니다. 자세한 내용은 변수 구성 및 출력 구성에 대한 HashiCorp 설명서를 참조하세요.
  - 모든 변수에는 정의된 유형이 있어야 합니다.
  - 변수 선언에는 기본 인수도 포함될 수 있습니다. 선언에 기본 인수가 포함된 경우 변수는 선택 사항으로 간주되며, 모듈을 호출하거나 Terraform을 실행할 때 값을 설정하지 않으면 기본값이 사용됩니다. 기본 인수에는 리터럴 값이 필요하며 구성의 다른 객체를 참조할 수 없습니다. 변수를 필수로 만들려면 변수 선언에서 기본값을 생략하고 설정이 `nullable = false` 적절한지 고려합니다.
  - 환경과 무관한 값(예: `disk_size`)이 있는 변수의 경우 기본값을 제공합니다.
  - 환경별 값(예: `project_id`)이 있는 변수의 경우 기본값을 제공하지 마십시오. 이 경우 호출 모듈은 의미 있는 값을 제공해야 합니다.
  - 변수를 비워 두는 것이 기본 APIs가 거부하지 않는 유효한 기본 설정인 경우에만 빈 문자열 또는 목록과 같은 변수에 대해 빈 기본값을 사용합니다.
  - 변수를 사용할 때는 신중해야 합니다. 각 인스턴스 또는 환경에 따라 변경해야 하는 경우에만 값을 파라미터화합니다. 변수를 노출할지 여부를 결정할 때 해당 변수를 변경하는 구체적인 사용 사례가 있는지 확인합니다. 변수가 필요할 가능성이 작으면 노출하지 마십시오.
    - 기본값으로 변수를 추가하는 것은 이전 버전과 호환됩니다.
    - 변수 제거는 이전 버전과 호환되지 않습니다.
    - 리터럴이 여러 위치에서 재사용되는 경우 변수로 노출하지 않고 로컬 값을 사용해야 합니다.
  - 출력을 입력 변수를 통해 직접 전달하지 마십시오. 이렇게 하면 종속성 그래프에 제대로 추가되지 않습니다. 암시적 종속성이 생성되도록 하려면가 리소스의 속성을 참조하도록 해야 합니다. 인스턴스의 입력 변수를 직접 참조하는 대신 속성을 전달합니다.

- locals.tf:이 파일에는 표현식에 이름을 할당하는 로컬 값이 포함되어 있으므로 표현식을 반복하는 대신 모듈 내에서 이름을 여러 번 사용할 수 있습니다. 로컬 값은 함수의 임시 로컬 변수와 같습니다. 로컬 값의 표현식은 리터럴 상수로 제한되지 않으며 변수, 리소스 속성 또는 기타 로컬 값을 포함하여 모듈의 다른 값을 참조하여 결합할 수도 있습니다.
- providers.tf:이 파일에는 terraform 블록과 공급자 블록이 포함되어 있습니다. `provider` 블록은 모듈 소비자가 루트 모듈에서만 선언해야 합니다.

  HCP Terraform을 사용하는 경우 빈 클라우드 블록도 추가합니다. `cloud` 블록은 CI/CD 파이프라인의 일부로 환경 변수 및 환경 변수 자격 증명을 통해 완전히 구성되어야 합니다.
- versions.tf:이 파일에는 required_providers 블록이 포함되어 있습니다. 모든 Terraform 모듈은 Terraform이 이러한 공급자를 설치하고 사용할 수 있도록 필요한 공급자를 선언해야 합니다.
- data.tf: 간단한 구성을 위해 데이터 소스를 참조하는 리소스 옆에 데이터 소스를 배치합니다. 예를 들어 인스턴스를 시작하는 데 사용할 이미지를 가져오는 경우 자체 파일에 데이터 리소스를 수집하는 대신 인스턴스와 함께 배치합니다. 데이터 소스 수가 너무 많아지면 전용 `data.tf` 파일로 이동하는 것이 좋습니다.
- .tfvars 파일: 루트 모듈의 경우 `.tfvars` 파일을 사용하여 민감하지 않은 변수를 제공할 수 있습니다. 일관성을 위해 변수 파일의 이름을 로 지정합니다`terraform.tfvars`. 리포지토리의 루트에 공통 값을 배치하고 `envs/` 폴더 내에 환경별 값을 배치합니다.
- 중첩 모듈: 중첩 모듈은 `modules/` 하위 디렉터리 아래에 있어야 합니다. 가 있는 중첩 모듈은 외부 사용자가 사용할 수 있는 `README.md` 것으로 간주됩니다. `README.md`가 없는 경우 모듈은 내부 전용으로 간주됩니다. 중첩 모듈을 사용하여 복잡한 동작을 사용자가 신중하게 선택하고 선택할 수 있는 여러 개의 작은 모듈로 분할해야 합니다.

  루트 모듈에 중첩 모듈에 대한 호출이 포함된 경우 이러한 호출은 Terraform이 해당 호출을 다시 다운로드하는 대신 동일한 리포지토리 또는 패키지의 일부로 간주`./modules/sample-module`하도록와 같은 상대 경로를 사용해야 합니다.

  리포지토리 또는 패키지에 여러 중첩 모듈이 포함된 경우 직접 서로를 호출하고 깊이 중첩된 모듈 트리를 생성하는 대신 호출자가 구성 가능한 것이 이상적입니다.
- 예: 재사용 가능한 모듈을 사용하는 예는 리포지토리 루트의 `examples/` 하위 디렉터리 아래에 있어야 합니다. 각 예제에 대해 README를 추가하여 예제의 목표와 사용량을 설명할 수 있습니다. 하위 모듈의 예제도 루트 `examples/` 디렉터리에 배치해야 합니다.

  예제는 사용자 지정을 위해 다른 리포지토리로 복사되는 경우가 많으므로 모듈 블록의 소스는 상대 경로가 아니라 외부 호출자가 사용할 주소로 설정되어야 합니다.

- 서비스 명명된 파일: 사용자는 종종 여러 파일의 서비스별로 Terraform 리소스를 분리하려고 합니다. 이 방법은 최대한 사용하지 않는 것이 좋으며 대신에 리소스를 정의해야 합니다`main.tf`. 그러나 리소스 모음(예: IAM 역할 및 정책)이 150줄을 초과하는 경우와 같은 자체 파일로 나누는 것이 좋습니다`iam.tf`. 그렇지 않으면 모든 리소스 코드를에 정의해야 합니다`main.tf`.
- 사용자 지정 스크립트: 필요한 경우에만 스크립트를 사용합니다. Terraform은 스크립트를 통해 생성된 리소스의 상태를 설명하거나 관리하지 않습니다. Terraform 리소스가 원하는 동작을 지원하지 않는 경우에만 사용자 지정 스크립트를 사용합니다. Terraform에서 호출한 사용자 지정 스크립트를 `scripts/` 디렉터리에 배치합니다.
- 헬퍼 스크립트: `helpers/` 디렉터리에서 Terraform에서 호출하지 않는 헬퍼 스크립트를 구성합니다. 설명 및 예제 호출과 함께 `README.md` 파일에 헬퍼 스크립트를 문서화합니다. 헬퍼 스크립트가 인수를 수락하는 경우 인수 확인 및 `--help` 출력을 제공합니다.
- 정적 파일: Terraform이 참조하지만 실행되지 않는 정적 파일(예: EC2 인스턴스에 로드된 시작 스크립트)은 `files/` 디렉터리로 구성되어야 합니다. HCL과 별도로 길이가 긴 문서를 외부 파일에 배치합니다. file() 함수를 사용하여 참조합니다.
- 템플릿: Terraform templatefile 함수가 읽는 파일의 경우 파일 확장명를 사용합니다`.tftpl`. 템플릿은 `templates/` 디렉터리에 배치해야 합니다.

루트 모듈 구조

Terraform은 항상 단일 루트 모듈의 컨텍스트에서 실행됩니다. 전체 Terraform 구성은 루트 모듈과 하위 모듈 트리(루트 모듈에서 호출하는 모듈, 해당 모듈에서 호출하는 모듈 등 포함)로 구성됩니다.

Terraform 루트 모듈 레이아웃 기본 예제:

```
.
### data.tf
### envs
#   ### dev
#   #   ### terraform.tfvars
#   ### prod
#   #   ### terraform.tfvars
#   ### test
#       ### terraform.tfvars
### locals.tf
### main.tf
### outputs.tf
### providers.tf
### README.md
```

```
### terraform.tfvars
### variables.tf
### versions.tf
```

재사용 가능한 모듈 구조

재사용 가능한 모듈은 루트 모듈과 동일한 개념을 따릅니다. 모듈을 정의하려면 루트 모듈을 정의하는 것처럼 새 디렉터리를 생성하고 내부에 `.tf` 파일을 배치합니다. Terraform은 로컬 상대 경로 또는 원격 리포지토리에서 모듈을 로드할 수 있습니다. 여러 구성에서 모듈을 재사용할 것으로 예상되는 경우 해당 모듈을 자체 버전 관리 리포지토리에 배치합니다. 모듈을 서로 다른 조합으로 더 쉽게 재사용할 수 있도록 모듈 트리를 비교적 평평하게 유지하는 것이 중요합니다.

Terraform 재사용 가능 모듈 레이아웃 기본 예제:

```
.
### data.tf
### examples
#   ### multi-az-new-vpc
#   #   ### data.tf
#   #   ### locals.tf
#   #   ### main.tf
#   #   ### outputs.tf
#   #   ### providers.tf
#   #   ### README.md
#   #   ### terraform.tfvars
#   #   ### variables.tf
#   #   ### versions.tf
#   #   ### vpc.tf
#   ### single-az-existing-vpc
#   #   ### data.tf
#   #   ### locals.tf
#   #   ### main.tf
#   #   ### outputs.tf
#   #   ### providers.tf
#   #   ### README.md
#   #   ### terraform.tfvars
#   #   ### variables.tf
#   #   ### versions.tf
### iam.tf
### locals.tf
### main.tf
### outputs.tf
```

```
### README.md
### variables.tf
### versions.tf
```

모듈화를 위한 구조

원칙적으로 모든 리소스와 기타 구문을 모듈로 결합할 수 있지만 중첩된 모듈과 재사용 가능한 모듈을 과도하게 사용하면 전체 Terraform 구성을 이해하고 유지 관리하기가 더 어려워질 수 있으므로 이러한 모듈을 조정에 사용하세요.

적절한 경우 구성을 재사용 가능한 모듈로 나누면 리소스 유형으로 구성된 아키텍처의 새로운 개념을 설명하여 추상화 수준을 높일 수 있습니다.

인프라를 재사용 가능한 정의로 모듈화할 때는 개별 구성 요소나 지나치게 복잡한 컬렉션 대신 논리적 리소스 세트를 목표로 하세요.

단일 리소스를 래핑하지 마십시오.

다른 단일 리소스 유형 주위에 얇은 래퍼인 모듈을 생성해서는 안 됩니다. 모듈 내부에 있는 기본 리소스 유형의 이름과 다른 모듈의 이름을 찾는 데 문제가 있는 경우 모듈이 새로운 추상화를 생성하지 않을 수 있습니다. 불필요한 복잡성이 추가됩니다. 대신 호출 모듈에서 직접 리소스 유형을 사용합니다.

논리적 관계 캡슐화

네트워킹 기반, 데이터 계층, 보안 제어 및 애플리케이션과 같은 관련 리소스 세트를 그룹화합니다. 재사용 가능한 모듈은 기능을 활성화하기 위해 함께 작동하는 인프라 부분을 캡슐화해야 합니다.

상속을 평평하게 유지

하위 디렉터리에 모듈을 중첩할 때는 하나 또는 두 개 이상의 레벨을 깊이 이동하지 마십시오. 깊이 중첩된 상속 구조는 구성과 문제 해결을 복잡하게 만듭니다. 모듈은 다른 모듈을 기반으로 빌드해야 합니다. 모듈을 통해 터널을 빌드해서는 안 됩니다.

팀은 아키텍처 패턴을 나타내는 논리적 리소스 그룹에 모듈을 집중하여 신뢰할 수 있는 인프라 기반을 빠르게 구성할 수 있습니다. 과다 엔지니어링 또는 과다 단순화 없이 추상화의 균형을 맞춥니다.

출력의 참조 리소스

재사용 가능한 모듈에 정의된 모든 리소스에 대해 리소스를 참조하는 출력을 하나 이상 포함합니다. 변수와 출력을 사용하면 모듈과 리소스 간의 종속성을 추론할 수 있습니다. 출력이 없으면 사용자는 Terraform 구성과 관련하여 모듈을 올바르게 정렬할 수 없습니다.

환경 일관성, 목적 기반 그룹화 및 내보낸 리소스 참조를 제공하는 잘 구성된 모듈을 사용하면 조직 전체의 Terraform 공동 작업을 대규모로 수행할 수 있습니다. 팀은 재사용 가능한 빌딩 블록에서 인프라를 조합할 수 있습니다.

공급자 구성 안 함

공유 모듈은 공급자를 호출 모듈로부터 상속하지만 모듈은 공급자 설정 자체를 구성해서는 안 됩니다. 모듈에서 공급자 구성 블록을 지정하지 마십시오. 이 구성은 전역적으로 한 번만 선언해야 합니다.

필수 공급자 선언

공급자 구성은 모듈 간에 공유되지만 공유 모듈은 자체 공급자 요구 사항도 선언해야 합니다. 이 방법을 통해 Terraform은 구성의 모든 모듈과 호환되는 공급자의 단일 버전이 있는지 확인하고 공급자의 글로벌(모듈에 구애받지 않음) 식별자 역할을 하는 소스 주소를 지정할 수 있습니다. 그러나 모듈별 공급자 요구 사항은와 같이 공급자가 액세스할 원격 엔드포인트를 결정하는 구성 설정을 지정하지 않습니다 AWS 리전.

모듈은 버전 요구 사항을 선언하고 하드 코딩된 공급자 구성을 방지함으로써 공유 공급자를 사용하여 Terraform 구성 전반에서 이식성과 재사용성을 제공합니다.

공유 모듈의 경우의 required_providers 블록에서 필요한 최소 공급자 버전을 정의합니다versions.tf.

모듈에 특정 버전의 AWS 공급자가 필요하다고 선언하려면 required_providers 블록 내에서 terraform 블록을 사용합니다.

```
terraform {
  required_version = ">= 1.0.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 4.0.0"
    }
```

```
  }
}
```

공유 모듈이 특정 버전의 AWS 공급자만 지원하는 경우 가장 오른쪽 버전 구성 요소만 증분할 수 있는 비관적 제약 연산자(~>)를 사용합니다.

```
terraform {
  required_version = ">= 1.0.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
```

이 예제에서는 `4.57.1` 및의 설치를 `~> 4.0` 허용`4.67.0`하지만의 설치는 허용하지 않습니다`5.0.0`. 자세한 내용은 HashiCorp 설명서의 버전 제약 조건 구문을 참조하세요.

이름 지정 규칙 준수

설명이 포함된 명확한 이름은 모듈의 리소스와 구성 값의 목적 간의 관계를 이해하는 데 도움이 됩니다. 스타일 지침과의 일관성은 모듈 사용자와 유지 관리자 모두의 가독성을 향상시킵니다.

리소스 이름 지정 지침을 따릅니다.

- 모든 리소스 이름이 Terraform 스타일 표준과 일치하도록 snake_case(소문자 용어는 밑줄로 구분됨)를 사용합니다. 이 방법은 리소스 유형, 데이터 소스 유형 및 기타 사전 정의된 값에 대한 명명 규칙과의 일관성을 보장합니다. 이 규칙은 이름 인수에는 적용되지 않습니다.
- 유형 중 유일한 리소스(예: 전체 모듈에 대한 단일 로드 밸런서)에 대한 참조를 단순화하려면 리소스의 이름을 지정`main`하거나 명확성을 `this` 위해를 지정합니다.
- 리소스의 목적과 컨텍스트를 설명하고 유사한 리소스(예: 기본 데이터베이스의 경우 , 데이터베이스의 읽기 전용 복제본`primary`의 `read_replica` 경우)를 구분하는 데 도움이 되는 의미 있는 이름을 사용합니다.
- 복수 이름이 아닌 단수를 사용합니다.
- 리소스 이름에 리소스 유형을 반복하지 마십시오.

변수 이름 지정 지침을 따릅니다.

- 디스크 크기 또는 RAM 크기(예: 기가바이트 단위의 `ram_size_gb` RAM 크기)와 같은 숫자 값을 나타내는 입력, 로컬 변수 및 출력의 이름에 단위를 추가합니다. 이 방법을 사용하면 구성 유지 관리에 대한 예상 입력 단위가 명확해집니다.
- 스토리지 크기에는 MiB 및 GiB와 같은 이진 단위를 사용하고 다른 지표에는 MB 또는 GB와 같은 십진 단위를 사용합니다.
- 와 같은 부울 변수의 양수 이름을 지정합니다`enable_external_access`.

첨부 파일 리소스 사용

일부 리소스에는 가상 리소스가 속성으로 포함되어 있습니다. 가능하면 이러한 임베디드 리소스 속성을 사용하지 말고 고유한 리소스를 사용하여 대신 해당 의사 리소스를 연결해야 합니다. 이러한 리소스 관계는 각 리소스에 고유한 cause-and-effect 문제를 일으킬 수 있습니다.

임베디드 속성 사용(이 패턴은 피함):

```
resource "aws_security_group" "allow_tls" {
  ...
  ingress {
    description      = "TLS from VPC"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = [aws_vpc.main.cidr_block]
    ipv6_cidr_blocks = [aws_vpc.main.ipv6_cidr_block]
  }

  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
    ipv6_cidr_blocks = ["::/0"]
  }
}
```

첨부 파일 리소스 사용(선호):

```
resource "aws_security_group" "allow_tls" {
```

```
  ...
}

resource "aws_security_group_rule" "example" {
  type              = "ingress"
  description      = "TLS from VPC"
  from_port        = 443
  to_port          = 443
  protocol         = "tcp"
  cidr_blocks      = [aws_vpc.main.cidr_block]
  ipv6_cidr_blocks = [aws_vpc.main.ipv6_cidr_block]
  security_group_id = aws_security_group.allow_tls.id
}
```

기본 태그 사용

태그를 수락할 수 있는 모든 리소스에 태그를 할당합니다. Terraform AWS Provider에는 루트 모듈 내에서 사용해야 하는 aws_default_tags 데이터 소스가 있습니다.

Terraform 모듈에서 생성한 모든 리소스에 필요한 태그를 추가하는 것이 좋습니다. 다음은 연결할 수 있는 태그 목록입니다.

- 이름: 사람이 읽을 수 있는 리소스 이름
- AppId: 리소스를 사용하는 애플리케이션의 ID입니다.
- AppRole: 리소스의 기술 함수. 예: "webserver" 또는 "database"
- AppPurpose: 리소스의 비즈니스 목적. 예: “frontend ui” 또는 “결제 프로세서”
- 환경: 개발, 테스트 또는 생산과 같은 소프트웨어 환경
- 프로젝트: 리소스를 사용하는 프로젝트
- CostCenter: 리소스 사용에 대한 청구 대상

Terraform 레지스트리 요구 사항 충족

모듈 리포지토리는 Terraform 레지스트리에 게시할 수 있도록 다음 요구 사항을 모두 충족해야 합니다.

모듈을 단기적으로 레지스트리에 게시할 계획이 없더라도 항상 이러한 요구 사항을 따라야 합니다. 이렇게 하면 리포지토리의 구성 및 구조를 변경할 필요 없이 나중에 모듈을 레지스트리에 게시할 수 있습니다.

- 리포지토리 이름: 모듈 리포지토리의 경우 세 부분으로 구성된 이름을 사용합니다. `terraform-aws-<NAME>`여기서는 모듈이 관리하는 인프라 유형을 <NAME> 반영합니다. <NAME> 세그먼트에는 추가 하이픈(예: `terraform-aws-iam-terraform-roles`)이 포함될 수 있습니다.
- 표준 모듈 구조: 모듈은 표준 리포지토리 구조를 준수해야 합니다. 이렇게 하면 레지스트리가 모듈을 검사하고 설명서를 생성하고 리소스 사용량을 추적하는 등의 작업을 수행할 수 있습니다.
  - Git 리포지토리를 생성한 후 모듈 파일을 리포지토리의 루트에 복사합니다. 재사용 가능한 각 모듈을 자체 리포지토리의 루트에 배치하는 것이 좋지만 하위 디렉터리의 모듈을 참조할 수도 있습니다.
  - HCP Terraform을 사용하는 경우 조직 레지스트리에 공유하려는 모듈을 게시합니다. 레지스트리는 HCP Terraform API 토큰을 사용한 다운로드를 처리하고 액세스를 제어하므로 소비자는 명령줄에서 Terraform을 실행하는 경우에도 모듈의 소스 리포지토리에 액세스할 필요가 없습니다.
- 위치 및 권한: 리포지토리는 구성된 버전 관리 시스템(VCS) 공급자 중 하나에 있어야 하며, HCP Terraform VCS 사용자 계정에는 리포지토리에 대한 관리자 액세스 권한이 있어야 합니다. 레지스트리는 새 모듈 버전을 가져오기 위해 웹후크를 생성하려면 관리자 액세스 권한이 필요합니다.
- 릴리스용 x.y.z 태그: 모듈을 게시하려면 릴리스 태그가 하나 이상 있어야 합니다. 레지스트리는 릴리스 태그를 사용하여 모듈 버전을 식별합니다. 릴리스 태그 이름은 시맨틱 버전 관리를 사용해야 하며, 선택적으로 접두사v(예: `v1.1.0` 및)를 사용할 수 있습니다`1.1.0`. 레지스트리는 버전 번호처럼 보이지 않는 태그를 무시합니다. 모듈 게시에 대한 자세한 내용은 Terraform 설명서를 참조하세요.

자세한 내용은 Terraform 설명서의 모듈 리포지토리 준비를 참조하세요.

권장 모듈 소스 사용

Terraform은 모듈 블록의 `source` 인수를 사용하여 하위 모듈의 소스 코드를 찾아 다운로드합니다.

반복되는 코드 요소를 고려하고 기본 Terraform 모듈 레지스트리 또는 여러 구성에서 공유하도록 의도된 모듈에 대한 VCS 공급자를 사용하는 것을 기본 목적으로 하는 밀접하게 관련된 모듈에는 로컬 경로를 사용하는 것이 좋습니다.

다음 예제에서는 모듈 공유를 위한 가장 일반적이고 권장되는 소스 유형을 보여줍니다. 레지스트리 모듈은 버전 관리를 지원합니다. 다음 예제와 같이 항상 특정 버전을 제공해야 합니다.

레지스트리

Terraform 레지스트리:

```
module "lambda" {
```

```
  source = "github.com/terraform-aws-modules/terraform-aws-lambda.git?
ref=e78cdf1f82944897ca6e30d6489f43cf24539374" #--> v4.18.0

  ...

}
```

커밋 해시를 고정하면 공급망 공격에 취약한 퍼블릭 레지스트리의 드리프트를 방지할 수 있습니다.

HCP Terraform:

```
module "eks_karpenter" {
  source = "app.terraform.io/my-org/eks/aws"
  version = "1.1.0"

  ...

  enable_karpenter = true
}
```

Terraform Enterprise:

```
module "eks_karpenter" {
  source = "terraform.mydomain.com/my-org/eks/aws"
  version = "1.1.0"

  ...

  enable_karpenter = true
}
```

VCS 공급자

VCS 공급자는 다음 예제와 같이 특정 개정을 선택하기 위한 `ref` 인수를 지원합니다.

GitHub(HTTPS):

```
module "eks_karpenter" {
  source = "github.com/my-org/terraform-aws-eks.git?ref=v1.1.0"

  ...
```

```
  enable_karpenter = true
}
```

일반 Git 리포지토리(HTTPS):

```
module "eks_karpenter" {
  source = "git::https://example.com/terraform-aws-eks.git?ref=v1.1.0"

  ...

  enable_karpenter = true
}
```

일반 Git 리포지토리(SSH):

> **Warning**
>
> 프라이빗 리포지토리에 액세스하려면 자격 증명을 구성해야 합니다.

```
module "eks_karpenter" {
  source = "git::ssh://username@example.com/terraform-aws-eks.git?ref=v1.1.0"

  ...

  enable_karpenter = true
}
```

코딩 표준 준수

모든 구성 파일에 일관된 Terraform 형식 지정 규칙 및 스타일을 적용합니다. CI/CD 파이프라인에서 자동 스타일 검사를 사용하여 표준을 적용합니다. 코딩 모범 사례를 팀 워크플로에 포함하면 사용량이 조직 전체에 광범위하게 분산됨에 따라 구성이 읽기 쉽고 유지 관리 가능하며 협업적으로 유지됩니다.

스타일 지침 준수

- HashiCorp 스타일 표준에 맞게 모든 Terraform 파일(`.tf` 파일)을 terraform fmt 명령으로 포맷합니다.
- terraform validate 명령을 사용하여 구성의 구문과 구조를 확인합니다.

- TFLint를 사용하여 코드 품질을 통계적으로 분석합니다. 이 린터는 형식을 지정하는 것 이상의 Terraform 모범 사례를 확인하고 오류가 발생하면 빌드에 실패합니다.

커밋 전 후크 구성

커밋을 허용하기 전에 `terraform fmt`, `tflint`checkov, 및 기타 코드 스캔 및 스타일 검사를 실행하는 클라이언트 측 커밋 전 후크를 구성합니다. 이 방법은 개발자 워크플로 초기에 표준 적합성을 검증하는 데 도움이 됩니다.

사전 커밋과 같은 사전 커밋 프레임워크를 사용하여 Terraform 린팅, 서식 지정 및 코드 스캔을 로컬 시스템의 후크로 추가합니다. 후크는 각 Git 커밋에서 실행되며 검사가 통과되지 않으면 커밋에 실패합니다.

스타일 및 품질 검사를 로컬 커밋 전 후크로 이동하면 변경 사항이 도입되기 전에 개발자에게 신속한 피드백을 제공할 수 있습니다. 표준은 코딩 워크플로의 일부가 됩니다.

AWS 프로바이더 버전 관리 모범 사례

AWS 제공자 및 관련 Terraform 모듈의 버전을 신중하게 관리하는 것은 안정성을 위해 매우 중요합니다. 이 섹션에서는 버전 제한 및 업그레이드에 대한 모범 사례를 간략하게 설명합니다.

모범 사례:

- 자동 버전 검사 추가
- 새 릴리스를 모니터링하세요.
- 제공업체에 기여하세요

자동 버전 검사 추가

CI/CD 파이프라인에 Terraform 공급자에 대한 버전 검사를 추가하여 버전 고정을 검증하고 버전이 정의되지 않은 경우 빌드에 실패합니다.

- CI/CD 파이프라인에 TFlint 검사를 추가하여 고정된 메이저/마이너 버전 제약이 정의되지 않은 제공자 버전을 스캔하세요. 발생 가능한 오류를 탐지하기 위한 규칙을 제공하고 리소스에 대한 모범 사례를 확인하는 Terraform AWS Provider용 TFlint 규칙 세트 플러그인을 사용하세요. AWS
- 고정되지 않은 제공자 버전을 탐지하여 암시적 업그레이드가 프로덕션에 도달하지 못하도록 하는 실패 CI 실행

새 릴리스를 모니터링하세요.

- 제공업체 릴리스 노트와 변경 로그 피드를 모니터링하세요. 새 메이저 릴리즈/마이너 릴리스에 대한 알림을 받으세요.
- 잠재적 주요 변경 사항이 있는지 릴리스 노트를 평가하고 기존 인프라에 미치는 영향을 평가하십시오.
- 프로덕션 환경을 업데이트하기 전에 먼저 비프로덕션 환경에서 마이너 버전을 업그레이드하여 유효성을 검사하십시오.

파이프라인의 버전 검사를 자동화하고 새 릴리스를 모니터링하면 지원되지 않는 업그레이드를 조기에 발견하고 팀이 프로덕션 환경을 업데이트하기 전에 새로운 메이저/마이너 릴리스의 영향을 평가할 시간을 확보할 수 있습니다.

제공업체에 기여하세요

결함을 보고하거나 GitHub 문제의 기능을 요청하여 HashiCorp AWS 공급업체에 적극적으로 기여하세요.

- AWS Provider 리포지토리에서 잘 문서화된 문제를 열어 발생한 버그 또는 누락된 기능에 대해 자세히 설명하십시오. 재현 가능한 단계를 제공하세요.
- 새로운 서비스를 관리할 수 있는 AWS 공급자의 역량을 확대하기 위한 개선 사항을 요청하고 투표하십시오.
- 제공업체 결함 또는 개선 사항에 대해 제안된 수정 사항을 제공할 때 발행된 풀 요청을 참조하세요. 관련 문제 링크.
- 코딩 규칙, 테스트 표준 및 문서에 대한 리포지토리의 기여 가이드라인을 따르십시오.

사용 중인 제공업체에 환원하면 해당 제공업체의 로드맵에 직접 내용을 입력하고 모든 사용자의 품질과 기능을 개선하는 데 도움을 줄 수 있습니다.

커뮤니티 모듈 모범 사례

모듈을 효과적으로 사용하는 것은 복잡한 Terraform 구성을 관리하고 재사용을 촉진하는 데 중요합니다. 이 섹션에서는 커뮤니티 모듈, 종속성, 소스, 추상화 및 기여에 대한 모범 사례를 제공합니다.

베스트 프랙티스:

- 커뮤니티 모듈 살펴보기
- 종속성 이해
- 신뢰할 수 있는 출처 사용
- 커뮤니티 모듈에 기여

커뮤니티 모듈 살펴보기

새 모듈을 빌드하기 전에 Terraform 레지스트리 및 기타 소스에서 사용 사례를 해결할 수 있는 기존 AWS 모듈을 검색하세요. GitHub 최신 업데이트가 적용되고 활발하게 유지 관리되고 있는 인기 옵션을 찾아보세요.

변수를 사용하여 사용자 지정하십시오.

커뮤니티 모듈을 사용할 때는 소스 코드를 분기하거나 직접 수정하는 대신 변수를 통해 입력을 전달하세요. 필요한 경우 모듈 내부를 변경하는 대신 기본값을 재정의하십시오.

포크는 더 넓은 커뮤니티에 도움이 되도록 원본 모듈에 수정 사항이나 기능을 제공하는 것으로 제한해야 합니다.

종속성 이해

모듈을 사용하기 전에 소스 코드와 설명서를 검토하여 종속성을 파악하세요.

- 필수 제공자: 모듈에 필요한 AWS, Kubernetes 또는 기타 제공자의 버전을 기록해 두십시오.
- 중첩 모듈: 내부적으로 사용되는 다른 모듈이 계단식 종속성을 유발하는지 확인하세요.
- 외부 데이터 소스: 모듈이 사용하는 API, 사용자 지정 플러그인 또는 인프라 종속성을 기록해 두십시오.

직접 및 간접 종속성의 전체 트리를 매핑하면 모듈을 사용할 때 예상치 못한 문제를 피할 수 있습니다.

신뢰할 수 있는 출처 사용

확인되지 않았거나 알려지지 않은 게시자로부터 Terraform 모듈을 소싱하는 것은 심각한 위험을 초래합니다. 신뢰할 수 있는 출처의 모듈만 사용하십시오.

- AWS 또는 HashiCorp 파트너와 같은 검증된 제작자가 게시한 Terraform 레지스트리의 인증 모듈을 선호하십시오.
- 사용자 지정 모듈의 경우 해당 모듈이 자체 조직의 모듈이더라도 게시자 기록, 지원 수준 및 사용 평판을 검토하세요.

출처가 알려지지 않았거나 알려지지 않은 모듈을 허용하지 않으면 코드에 취약성이나 유지 관리 문제가 발생할 위험을 줄일 수 있습니다.

알림 구독

신뢰할 수 있는 게시자의 새 모듈 출시 알림을 구독하세요.

- GitHub 모듈 리포지토리에서 모듈의 새 버전에 대한 알림을 받으세요.
- 게시자 블로그와 변경 로그를 모니터링하여 업데이트를 확인하세요.
- 업데이트를 암시적으로 가져오는 대신 검증되고 높은 평가를 받은 출처로부터 새 버전에 대한 사전 알림을 받으세요.

신뢰할 수 있는 출처의 모듈만 사용하고 변경 사항을 모니터링하면 안정성과 보안이 보장됩니다. 검증된 모듈은 공급망 위험을 최소화하면서 생산성을 향상시킵니다.

커뮤니티 모듈에 기여

다음에서 GitHub 호스팅되는 커뮤니티 모듈에 대한 수정 및 개선 사항을 제출하십시오.

- 모듈에 대한 풀 리퀘스트를 열어 사용 중에 발생하는 결함이나 제한 사항을 해결하세요.
- 이슈를 생성하여 기존 OSS 모듈에 새 모범 사례 구성을 추가하도록 요청하세요.

커뮤니티 모듈에 기여하면 모든 Terraform 실무자가 재사용 가능하고 코드화된 패턴을 개선할 수 있습니다.

FAQ

Q. 공급자에 초점을 맞추는 이유는 무엇입니까? AWS

A. AWS 프로바이더는 Terraform에서 인프라 프로비저닝에 가장 널리 사용되고 복잡한 프로바이더 중 하나입니다. 이러한 모범 사례를 따르면 사용자는 환경에 맞게 공급자 사용을 최적화하는 데 도움이 됩니다. AWS

Q. 저는 Terraform을 처음 사용합니다. 이 가이드를 사용할 수 있나요?

A. 이 가이드는 Terraform을 처음 접하는 사람과 기술 수준을 높이려는 고급 실무자를 위한 것입니다. 이러한 관행은 학습의 모든 단계에서 사용자의 워크플로를 개선합니다.

Q: 다루는 몇 가지 주요 모범 사례는 무엇입니까?

A. 주요 모범 사례에는 액세스 키보다 IAM 역할 사용, 버전 고정, 자동 테스트 통합, 원격 상태 잠금, 자격 증명 교체, 제공자에 대한 지원, 논리적인 코드 기반 구성 등이 포함됩니다.

Q. Terraform에 대한 자세한 내용은 어디에서 확인할 수 있습니까?

A. 리소스 섹션에는 공식 HashiCorp Terraform 설명서 및 커뮤니티 포럼으로 연결되는 링크가 포함되어 있습니다. 링크를 사용하여 고급 Terraform 워크플로에 대해 자세히 알아보세요.

다음 단계

이 가이드를 읽은 후 다음 단계로 넘어갈 수 있는 몇 가지 단계는 다음과 같습니다.

- 기존 Terraform 코드베이스가 있는 경우 구성을 검토하고 이 가이드에 제공된 권장 사항을 기반으로 개선할 수 있는 영역을 식별하세요. 예를 들어 원격 백엔드 구현, 코드를 모듈로 분리, 버전 고정 사용 등에 대한 모범 사례를 검토하고 구성에서 이를 검증하세요.
- 기존 Terraform 코드베이스가 없는 경우 새 구성을 구성할 때 이 모범 사례를 사용하세요. 상태 관리, 인증, 코드 구조 등에 대한 조언을 처음부터 따르십시오.
- 이 가이드에서 참조하는 몇 가지 HashiCorp 커뮤니티 모듈을 사용하여 아키텍처 패턴을 단순화하는지 확인해 보세요. 모듈은 더 높은 수준의 추상화를 허용하므로 공통 리소스를 다시 작성할 필요가 없습니다.
- 린팅, 보안 스캔, 정책 검사, 자동화된 테스트 도구를 사용하여 보안, 규정 준수, 코드 품질과 관련된 몇 가지 모범 사례를 강화하세요. TFlint, tfsec, Checkov와 같은 도구가 도움이 될 수 있습니다.
- 최신 AWS 공급자 설명서를 검토하여 Terraform 사용을 최적화하는 데 도움이 될 수 있는 새로운 리소스 또는 기능이 있는지 확인하세요. 공급자의 AWS 새 버전에 대한 최신 정보를 확인하세요.
- 추가 지침은 웹 사이트의 Terraform 설명서, 모범 사례 가이드 및 스타일 가이드를 참조하십시오. HashiCorp

리소스

참조

다음 링크는 Terraform AWS 제공자 및 IaC용 Terraform 사용에 대한 추가 읽기 자료를 제공합니다. AWS

- 테라폼 제공자 (설명서) AWS HashiCorp
- AWS 서비스용 테라폼 모듈 (테라폼 레지스트리)
- AWS 및 HashiCorp 파트너십 (블로그 게시물) HashiCorp
- AWS 공급자를 통한 동적 자격 증명 (HCP Terraform 설명서)
- DynamoDB 상태 잠금 (테라폼 설명서)
- Sentinel을 통한 정책 적용 (테라폼 문서)

도구

다음 도구는 이 모범 사례 가이드에서 AWS권장하는 대로 Terraform 구성의 코드 품질과 자동화를 개선하는 데 도움이 됩니다.

코드 품질:

- Checkov: 배포 전에 Terraform 코드를 스캔하여 구성 오류를 식별합니다.
- TFlint: 발생 가능한 오류, 더 이상 사용되지 않는 구문, 사용하지 않는 선언을 식별합니다. 또한 이 린터는 모범 사례와 명명 규칙을 적용할 수 있습니다. AWS
- terraform-docs: Terraform 모듈에서 다양한 출력 형식으로 문서를 생성합니다.

자동화 도구:

- HCP Terraform: 팀이 정책 검사 및 승인 게이트를 통해 Terraform 워크플로를 버전 지정, 협업 및 구축할 수 있도록 지원합니다.
- Atlantis: 코드 변경 검증을 위한 오픈 소스 Terraform 풀 리퀘스트 자동화 도구입니다.
- Terraform용 CDK: HashiCorp 구성 언어 (HCL) 대신 TypeScript Python, Java, C#, Go와 같은 친숙한 언어를 사용하여 Terraform 인프라를 코드로 정의, 프로비저닝 및 테스트할 수 있는 프레임워크입니다.

문서 기록

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

변경 사항	설명	날짜
원격 상태 잠금 업데이트	이제 권장되는 접근 방식인 Amazon S3 네이티브 상태 잠금을 반영하도록 백엔드 모범 사례 섹션을 업데이트했습니다.	2025년 8월 26일
최초 게시	—	2024년 5월 28일

AWS 권장 가이드 용어집

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

숫자

7가지 전략

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

- 리팩터링/리아키텍트 - 클라우드 네이티브 기능을 최대한 활용하여 애플리케이션을 이동하고 해당 아키텍처를 수정함으로써 민첩성, 성능 및 확장성을 개선합니다. 여기에는 일반적으로 운영 체제와 데이터베이스 이식이 포함됩니다. 예: 온프레미스 Oracle 데이터베이스를 Amazon Aurora PostgreSQL 호환 에디션으로 마이그레이션합니다.
- 리플랫포밍(리프트 앤드 리셰이프) - 애플리케이션을 클라우드로 이동하고 일정 수준의 최적화를 도입하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드의 Amazon Relational Database Service(Amazon RDS) for Oracle로 마이그레이션합니다.
- 재구매(드롭 앤드 숍) - 일반적으로 기존 라이선스에서 SaaS 모델로 전환하여 다른 제품으로 전환합니다. 예: 고객 관계 관리(CRM) 시스템을 Salesforce.com으로 마이그레이션합니다.
- 리호스팅(리프트 앤드 시프트) - 애플리케이션을 변경하지 않고 클라우드로 이동하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드클라우드의 EC2 인스턴스에 있는 Oracle로 마이그레이션합니다.
- 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 로 마이그레이션합니다 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 참조하세요.

추상화된 서비스

관리형 서비스를 참조하세요.

ACID

원자성, 일관성, 격리성, 내구성을 참조하세요.

능동-능동 마이그레이션

양방향 복제 도구 또는 이중 쓰기 작업을 사용하여 소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되고, 두 데이터베이스 모두 마이그레이션 중 연결 애플리케이션의 트랜잭션을 처리하는 데이터베이스 마이그레이션 방법입니다. 이 방법은 일회성 전환이 필요한 대신 소규모의 제어된 배치로 마이그레이션을 지원합니다. 더 유연하지만 액티브 패시브 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 작동하고 그룹에 대한 단일 반환 값을 계산하는 SQL 함수입니다. 집계 함수의 예로 `SUM` 및 `MAX`가 있습니다.

AI

인공 지능을 참조하세요.

AIOps

인공 지능 운영을 참조하세요.

익명화

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

안티 패턴

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

애플리케이션 제어

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

애플리케이션 포트폴리오

애플리케이션 구축 및 유지 관리 비용과 애플리케이션의 비즈니스 가치를 비롯하여 조직에서 사용하는 각 애플리케이션에 대한 세부 정보 모음입니다. 이 정보는 포트폴리오 탐색 및 분석 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 용 ABAC AWS를 참조하세요.

신뢰할 수 있는 데이터 소스

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

가용 영역

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

AWS 클라우드 채택 프레임워크(AWS CAF)

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

AWS 워크로드 검증 프레임워크(AWS WQF)

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

B

악성 봇

개인 또는 조직을 방해하거나 해를 입히기 위한 봇입니다.

BCP

비즈니스 연속성 계획을 참조하세요.

동작 그래프

리소스 동작과 시간 경과에 따른 상호 작용에 대한 통합된 대화형 뷰입니다. Amazon Detective에서 동작 그래프를 사용하여 실패한 로그온 시도, 의심스러운 API 직접 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 참조하세요.

바이너리 분류

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

블룸 필터

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

블루/그린(Blue/Green) 배포

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

bot

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

봇넷

맬웨어에 감염되고 봇 허더 또는 봇 운영자와 같은 단일 당사자가 제어하는 봇 네트워크입니다. 봇넷은 봇의 규모와 봇의 영향 범위를 확대하는 가장 잘 알려진 메커니즘입니다.

브랜치

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

긴급 액세스 권한

예외적인 상황에서 승인된 프로세스를 통해 사용자가 일반적으로 액세스할 권한이 없는에 액세스할 수 AWS 계정 있는 빠른 방법입니다. 자세한 내용은 AWS Well-Architected 지침의 Implement break-glass procedures 지표를 참조하세요.

브라운필드 전략

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

버퍼 캐시

가장 자주 액세스하는 데이터가 저장되는 메모리 영역입니다.

사업 역량

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

비즈니스 연속성 계획(BCP)

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

C

CAF

AWS Cloud Adoption Framework를 참조하세요.

카나리 배포

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

CCoE

클라우드 혁신 센터를 참조하세요.

CDC

데이터 캡처 변경을 참조하세요.

변경 데이터 캡처(CDC)

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

카오스 엔지니어링

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

CI/CD

지속적 통합 및 지속적 전송을 참조하세요.

분류

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

클라이언트측 암호화

대상이 데이터를 AWS 서비스 수신하기 전에 로컬에서 데이터를 암호화합니다.

클라우드 혁신 센터(CCoE)

클라우드 모범 사례 개발, 리소스 동원, 마이그레이션 타임라인 설정, 대규모 혁신을 통한 조직 선도 등 조직 전체에서 클라우드 채택 노력을 추진하는 다분야 팀입니다. 자세한 내용은 AWS 클라우드 엔터프라이즈 전략 블로그의 CCoE 게시물을 참조하세요.

클라우드 컴퓨팅

원격 데이터 스토리지와 IoT 디바이스 관리에 일반적으로 사용되는 클라우드 기술 클라우드 컴퓨팅은 일반적으로 엣지 컴퓨팅 기술에 연결되어 있습니다.

클라우드 운영 모델

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

클라우드 채택 단계

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

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

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

CMDB

구성 관리 데이터베이스를 참조하세요.

코드 리포지토리

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

콜드 캐시

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

콜드 데이터

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

컴퓨터 비전(CV)

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

구성 드리프트

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

구성 관리 데이터베이스(CMDB)

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

규정 준수 팩

규정 준수 및 보안 검사를 사용자 지정하기 위해 조합할 수 있는 AWS Config 규칙 및 수정 작업 모음입니다. YAML 템플릿을 사용하여 적합성 팩을 AWS 계정 및 리전 또는 조직 전체에 단일 엔터티로 배포할 수 있습니다. 자세한 내용은 AWS Config 설명서의 적합성 팩을 참조하세요.

지속적 통합 및 지속적 전달(CI/CD)

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

CV

컴퓨터 비전을 참조하세요.

D

저장 데이터

스토리지에 있는 데이터와 같이 네트워크에 고정되어 있는 데이터입니다.

데이터 분류

중요도와 민감도를 기준으로 네트워크의 데이터를 식별하고 분류하는 프로세스입니다. 이 프로세스는 데이터에 대한 적절한 보호 및 보존 제어를 결정하는 데 도움이 되므로 사이버 보안 위험 관리 전략의 중요한 구성 요소입니다. 데이터 분류는 AWS Well-Architected Framework의 보안 원칙 구성 요소입니다. 자세한 내용은 데이터 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 거버넌스를 통해 분산되고 탈중앙화된 데이터 소유권을 제공하는 아키텍처 프레임워크입니다.

데이터 최소화

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

데이터 경계

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

데이터 사전 처리

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

데이터 출처

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

데이터 주체

데이터를 수집 및 처리하는 개인입니다.

데이터 웨어하우스

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

데이터 정의 언어(DDL)

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

데이터베이스 조작 언어(DML)

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

DDL

데이터 정의 언어를 참조하세요.

딥 앙상블

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

딥 러닝

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

심층 방어

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

위임된 관리자

에서 AWS Organizations호환되는 서비스는 AWS 멤버 계정을 등록하여 조직의 계정을 관리하고 해당 서비스에 대한 권한을 관리할 수 있습니다. 이러한 계정을 해당 서비스의 위임된 관리자라고 합니다. 자세한 내용과 호환되는 서비스 목록은 AWS Organizations 설명서의 AWS 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 참조하세요.

탐지 제어

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

개발 가치 흐름 매핑 (DVSM)

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

디지털 트윈

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

차원 테이블

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

재해

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

재해 복구(DR)

재해로 인한 가동 중지 시간 및 데이터 손실을 최소화하기 위해 사용하는 전략 및 프로세스입니다. 자세한 내용은 AWS Well-Architected Framework의 Disaster Recovery of Workloads on AWS: Recovery in the Cloud를 참조하세요.

DML

데이터베이스 조작 언어를 참조하세요.

도메인 기반 설계

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

DR

재해 복구를 참조하세요.

드리프트 감지

기준이 되는 구성과의 편차 추적을 말합니다. 예를 들어 AWS CloudFormation 를 사용하여 시스템 리소스의 드리프트를 감지하거나를 사용하여 AWS Control Tower 거버넌스 요구 사항 준수에 영향을 미칠 수 있는 랜딩 존의 변경 사항을 감지할 수 있습니다.

DVSM

개발 가치 흐름 매핑을 참조하세요.

E

EDA

탐색 데이터 분석을 참조하세요.

EDI

전자 데이터 교환을 참조하세요.

엣지 컴퓨팅

IoT 네트워크의 엣지에서 스마트 디바이스의 컴퓨팅 성능을 개선하는 기술 엣지 컴퓨팅은 클라우드 컴퓨팅에 비해 보다 통신 지연 시간을 줄이고 응답 시간을 개선할 수 있습니다.

전자 데이터 교환(EDI)

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

암호화

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

암호화 키

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

엔디안

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

엔드포인트

서비스 엔드포인트를 참조하세요.

엔드포인트 서비스

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

엔터프라이즈 리소스 계획(ERP)

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

봉투 암호화

암호화 키를 다른 암호화 키로 암호화하는 프로세스입니다. 자세한 내용은 AWS Key Management Service (AWS KMS) 설명서의 봉투 암호화를 참조하세요.

환경

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

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

에픽

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

ERP

엔터프라이즈 리소스 계획을 참조하세요.

탐색 데이터 분석(EDA)

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

F

팩트 테이블

스타 스키마의 중앙 테이블입니다. 비즈니스 운영에 대한 정량적 데이터를 저장합니다. 일반적으로 팩트 테이블은 측정값이 있는 열 및 차원 테이블에 대한 외래 키가 있는 열과 같이 두 가지 열 유형을 포함합니다.

빠른 실패

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

장애 격리 경계

에서 장애의 영향을 제한하고 워크로드의 복원력을 개선하는 데 도움이 되는 가용 영역, AWS 리전 컨트롤 플레인 또는 데이터 플레인과 같은 AWS 클라우드경계입니다. 자세한 내용은 AWS 장애 격리 경계를 참조하세요.

기능 브랜치

브랜치를 참조하세요.

기능

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

기능 중요도

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

기능 변환

추가 소스로 데이터를 보강하거나, 값을 조정하거나, 단일 데이터 필드에서 여러 정보 세트를 추출하는 등 ML 프로세스를 위해 데이터를 최적화하는 것입니다. 이를 통해 ML 모델이 데이터를 활용

할 수 있습니다. 예를 들어, 날짜 '2021-05-27 00:15:37'을 '2021년', '5월', '목', '15일'로 분류하면 학습 알고리즘이 다양한 데이터 구성 요소와 관련된 미묘한 패턴을 학습하는 데 도움이 됩니다.

퓨샷 프롬프팅

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

FGAC

세분화된 액세스 제어를 참조하세요.

세분화된 액세스 제어(FGAC)

여러 조건을 사용하여 액세스 요청을 허용하거나 거부합니다.

플래시컷 마이그레이션

단계적 접근 방식을 사용하는 대신 변경 데이터 캡처를 통해 지속적 데이터 복제를 사용하여 최단 시간에 데이터를 마이그레이션하는 데이터베이스 마이그레이션 방법입니다. 목표는 가동 중지 시간을 최소화하는 것입니다.

FM

파운데이션 모델을 참조하세요.

파운데이션 모델(FM)

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

G

생성형 AI

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

지리적 차단

지리적 제한을 참조하세요.

지리적 제한(지리적 차단)

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

Gitflow 워크플로

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

골든 이미지

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

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, 이는 AWS Config,, Amazon GuardDuty AWS Security Hub CSPM, , AWS Trusted Advisor Amazon Inspector 및 사용자 지정 AWS Lambda 검사를 사용하여 구현됩니다.

H

HA

고가용성을 참조하세요.

이기종 데이터베이스 마이그레이션

다른 데이터베이스 엔진을 사용하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Oracle에서 Amazon Aurora로) 이기종 마이그레이션은 일반적으로 리아키텍트 작업의 일부이며 스

키마를 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.

홀드아웃 데이터

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

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. 핫픽스는 긴급하기 때문에 일반적인 DevOps 릴리스 워크플로 외부에서 실행됩니다.

하이퍼케어 기간

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

I

IaC

코드형 인프라를 참조하세요.

자격 증명 기반 정책

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

유휴 애플리케이션

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

IIoT

산업용 사물 인터넷을 참조하세요.

변경 불가능한 인프라

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

인바운드(수신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 외부에서 네트워크 연결을 수락, 검사 및 라우팅하는 VPC입니다. AWS Security Reference 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 것이 제대로 작동하는지 확인한 후에는 레거시 시스템을 폐기할 수 있을 때까지 추가 마이크로서비스 또는 사용자를 점진적으로 이동할 수 있습니다. 이 전략을 사용하면 대규모 마이그레이션과 관련된 위험을 줄일 수 있습니다.

Industry 4.0

연결성, 실시간 데이터, 자동화, 분석 및 AI/ML의 발전을 통해 제조 프로세스의 현대화를 나타내기 위해 2016년에 Klaus Schwab에서 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 아키텍처에서는 VPC(동일하거나 다른 AWS 리전), 인터넷 및 온프레미스 네트워크 간의 네트워크 트래픽 검사를 관리하는 중앙 집중식 VPCs입니다. 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 기계 학습 모델 해석 가능성을 참조하세요 AWS.

IoT

사물 인터넷을 참조하세요.

IT 정보 라이브러리(ITIL)

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

IT 서비스 관리(ITSM)

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

ITIL

IT 정보 라이브러리를 참조하세요.

ITSM

IT 서비스 관리를 참조하세요.

L

레이블 기반 액세스 제어(LBAC)

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

랜딩 존

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

대규모 언어 모델(LLM)

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

대규모 마이그레이션

300대 이상의 서버 마이그레이션입니다.

LBAC

레이블 기반 액세스 제어를 참조하세요.

최소 권한

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

리프트 앤드 시프트

7R을 참조하세요.

리틀 엔디안 시스템

가장 덜 중요한 바이트를 먼저 저장하는 시스템입니다. 엔디안도 참조하세요.

LLM

대규모 언어 모델을 참조하세요.

하위 환경

환경을 참조하세요.

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

브랜치를 참조하세요.

맬웨어

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

관리형 서비스

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

제조 실행 시스템(MES)

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

MAP

Migration Acceleration Program을 참조하세요.

메커니즘

도구를 생성하고 도구 채택을 유도한 다음 조정을 위해 결과를 검사하는 전체 프로세스입니다. 메커니즘은 작동 시 자체적으로 강화하고 개선하는 주기입니다. 자세한 내용은 AWS Well-Architected Framework의 메커니즘 구축을 참조하세요.

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 참조하세요.

메시지 큐 원격 분석 전송(MQTT)

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

마이크로서비스

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

마이크로서비스 아키텍처

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

Migration Acceleration Program(MAP)

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

대규모 마이그레이션

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

마이그레이션 팩토리

자동화되고 민첩한 접근 방식을 통해 워크로드 마이그레이션을 간소화하는 다기능 팀입니다. 마이그레이션 팩토리 팀에는 일반적으로 스프린트에서 일하는 운영, 비즈니스 분석가 및 소유자, 마이그레이션 엔지니어, 개발자, DevOps 전문가가 포함됩니다. 엔터프라이즈 애플리케이션 포트폴리오의 20~50%는 공장 접근 방식으로 최적화할 수 있는 반복되는 패턴으로 구성되어 있습니다. 자세한 내용은 이 콘텐츠 세트의 클라우드 마이그레이션 팩토리 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 및 AWS 계정이 있습니다.

마이그레이션 패턴

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

Migration Portfolio Assessment(MPA)

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

마이그레이션 준비 상태 평가(MRA)

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

마이그레이션 전략

워크로드를 AWS 클라우드로 마이그레이션하는 데 사용되는 접근 방식입니다. 자세한 내용은 이 용어집의 7R 항목과 조직을 동원하여 대규모 마이그레이션 가속화를 참조하세요.

ML

기계 학습을 참조하세요.

현대화

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

현대화 준비 상태 평가

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

모놀리식 애플리케이션(모놀리식 유형)

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

MPA

Migration Portfolio Assessment를 참조하세요.

MQTT

메시지 큐 원격 분석 전송을 참조하세요.

멀티클래스 분류

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

변경 가능한 인프라

프로덕션 워크로드에 대한 기존 인프라를 업데이트하고 수정하는 모델입니다. 일관성, 신뢰성 및 예측 가능성을 높이기 위해 AWS Well-Architected Framework에서는 변경 불가능한 인프라를 모범 사례로 사용할 것을 권장합니다.

O

OAC

오리진 액세스 제어를 참조하세요.

OAI

오리진 액세스 ID를 참조하세요.

OCM

조직 변경 관리를 참조하세요.

오프라인 마이그레이션

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

OI

운영 통합을 참조하세요.

OLA

운영 수준 계약을 참조하세요.

온라인 마이그레이션

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

OPC-UA

Open Process Communications - Unified Architecture(OPC-UA)를 참조하세요.

Open Process Communications - Unified Architecture(OPC-UA)

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

운영 수준 협약(OLA)

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

운영 준비 상태 검토(ORR)

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

운영 기술(OT)

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

운영 통합(OI)

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

조직 트레일

조직 AWS 계정 내 모든에 대한 모든 이벤트를 로깅 AWS CloudTrail 하는에서 생성된 추적입니다 AWS Organizations. 이 트레일은 조직에 속한 각 AWS 계정 에 생성되고 각 계정의 활동을 추적합니다. 자세한 내용은 CloudTrail 설명서의 Creating a trail for an organization을 참조하십시오.

조직 변경 관리(OCM)

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

오리진 액세스 제어(OAC)

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

오리진 액세스 ID(OAI)

CloudFront에서 Amazon S3 콘텐츠를 보호하기 위해 액세스를 제한하는 옵션입니다. OAI를 사용하면 CloudFront는 Amazon S3가 인증할 수 있는 보안 주체를 생성합니다. 인증된 보안 주체는 특

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

ORR

운영 준비 상태 검토를 참조하세요.

OT

운영 기술을 참조하세요.

아웃바운드(송신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 내에서 시작된 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 참조하세요.

플레이북

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

PLC

프로그래밍 가능 로직 컨트롤러를 참조하세요.

PLM

제품 수명 주기 관리를 참조하세요.

정책

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

다국어 지속성

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

포트폴리오 평가

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

조건자

보통 `WHERE` 절에 있는 `true` 또는 `false`를 반환하는 쿼리 조건입니다.

푸시다운 조건자

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

예방적 제어

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

보안 주체

작업을 수행하고 리소스에 액세스할 수 AWS 있는의 엔터티입니다. 이 엔터티는 일반적으로 , AWS 계정 IAM 역할 또는 사용자의 루트 사용자입니다. 자세한 내용은 IAM 설명서의 역할 용어 및 개념의 보안 주체를 참조하십시오.

개인 정보 보호 중심 설계

전체 개발 프로세스에서 개인 정보를 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

프라이빗 호스팅 영역

Amazon Route 53에서 하나 이상의 VPC 내 도메인과 하위 도메인에 대한 DNS 쿼리에 응답하는 방법에 대한 정보가 담긴 컨테이너입니다. 자세한 내용은 Route 53 설명서의 프라이빗 호스팅 영역 작업을 참조하십시오.

선제적 제어

규정 미준수 리소스의 배포를 방지하도록 설계된 보안 제어입니다. 이러한 제어는 리소스를 프로비저닝하기 전에 리소스를 스캔합니다. 리소스가 제어를 준수하지 않으면 프로비저닝되지 않습니다. 자세한 내용은 AWS Control Tower 설명서의 제어 참조 가이드를 참조하고 보안 제어 구현의 사전 예방적 제어를 참조하세요. AWS

제품 수명 주기 관리(PLM)

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

프로덕션 환경

환경을 참조하세요.

프로그래밍 가능 로직 컨트롤러(PLC)

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

프롬프트 체이닝

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

가명화

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

게시/구독(pub/sub)

여러 마이크로서비스에서 비동기 통신을 지원하여 확장성과 응답성을 개선하는 패턴입니다. 예를 들어 마이크로서비스 기반 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 명령어와 같은 일련의 단계입니다.

쿼리 계획 회귀

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

R

RACI 매트릭스

Responsible, Accountable, Consulted, Informed(RACI)를 참조하세요.

RAG

검색 증강 생성을 참조하세요.

랜섬웨어

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

RASCI 매트릭스

Responsible, Accountable, Consulted, Informed(RACI)를 참조하세요.

RCAC

행 및 열 액세스 제어를 참조하세요.

읽기 전용 복제본

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

리아키텍팅

7R을 참조하세요.

Recovery Point Objective(RPO)

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

Recovery Time Objective(RTO)

서비스 중단과 서비스 복원 사이의 허용 가능한 지연 시간입니다.

리팩터링

7R을 참조하세요.

리전

지리적 영역의 AWS 리소스 모음입니다. 각 AWS 리전 는 내결함성, 안정성 및 복원력을 제공하기 위해 서로 격리되고 독립적입니다. 자세한 내용은 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

회귀

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

리호스팅

7R을 참조하세요.

릴리스

배포 프로세스에서 변경 사항을 프로덕션 환경으로 승격시키는 행위입니다.

재배치

7R을 참조하세요.

리플랫포밍

7R을 참조하세요.

재구매

7R을 참조하세요.

복원력

중단에 저항하거나 중단을 복구할 수 있는 애플리케이션의 기능입니다. 고가용성 및 재해 복구는 AWS 클라우드에서 복원력을 계획할 때 일반적인 고려 사항입니다. 자세한 내용은 AWS 클라우드 복원력을 참조하세요.

리소스 기반 정책

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

RACI(Responsible, Accountable, Consulted, Informed) 매트릭스

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

대응 제어

보안 기준에서 벗어나거나 부정적인 이벤트를 해결하도록 설계된 보안 제어입니다. 자세한 내용은 AWS에서 보안 제어 구현의 대응 제어를 참조하세요.

retain

7R을 참조하세요.

사용 중지

7R을 참조하세요.

검색 증강 세대(RAG)

응답을 생성하기 전에 LLM이 훈련 데이터 소스 외부에 있는 신뢰할 수 있는 데이터 소스를 참조하는 생성형 AI 기술입니다. 예를 들어 RAG 모델은 조직의 지식 기반 또는 사용자 지정 데이터에 대한 시맨틱 검색을 수행할 수 있습니다. 자세한 내용은 검색 증강 생성(RAG)이란 무엇인가요?를 참조하세요.

교체

공격자가 자격 증명에 액세스하는 것을 더욱 어렵게 만들기 위해 보안 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

목표 복구 시점(RPO)을 참조하세요.

RTO

목표 복구 시간(RTO)을 참조하세요.

런북

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

S

SAML 2.0

많은 ID 제공업체(idP)에서 사용하는 개방형 표준입니다. 이 기능을 사용하면 연동 SSO(Single Sign-On)를 AWS Management Console 사용할 수 있으므로 사용자는 조직의 모든 사용자에 대해 IAM에서 사용자를 생성하지 않고도에 로그인하거나 AWS 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 획득을 참조하세요.

SCP

서비스 제어 정책을 참조하세요.

보안 암호

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

보안 중심 설계

전체 개발 프로세스에서 보안을 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

보안 제어

위협 행위자가 보안 취약성을 악용하는 능력을 방지, 탐지 또는 감소시키는 기술적 또는 관리적 가드레일입니다. 보안 제어는 예방, 감지, 대응, 선제적과 같은 기본적인 네 가지 보안 제어 유형으로 구분됩니다.

보안 강화

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

보안 정보 및 이벤트 관리(SIEM) 시스템

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

보안 응답 자동화

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

서버 측 암호화

대상에서 데이터를 수신하는 AWS 서비스 에 의한 데이터 암호화.

서비스 제어 정책(SCP)

AWS Organizations에 속한 조직의 모든 계정에 대한 권한을 중앙 집중식으로 제어하는 정책입니다. SCP는 관리자가 사용자 또는 역할에 위임할 수 있는 작업에 대해 제한을 설정하거나 가드레일을 정의합니다. SCP를 허용 목록 또는 거부 목록으로 사용하여 허용하거나 금지할 서비스 또는 작업을 지정할 수 있습니다. 자세한 내용은 AWS Organizations 설명서의 서비스 제어 정책을 참조하세요.

서비스 엔드포인트

에 대한 진입점의 URL입니다 AWS 서비스. 엔드포인트를 사용하여 대상 서비스에 프로그래밍 방식으로 연결할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

서비스 수준에 관한 계약(SLA)

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

서비스 수준 지표(SLI)

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

서비스 수준 목표(SLO)

서비스 수준 지표로 측정되는 서비스의 상태를 나타내는 목표 지표입니다.

공동 책임 모델

클라우드 보안 및 규정 준수를 AWS 위해와 공유하는 책임을 설명하는 모델입니다. AWS 는 클라우드의 보안을 담당하는 반면, 사용자는 클라우드의 보안을 담당합니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

SIEM

보안 정보 및 이벤트 관리 시스템을 참조하세요.

단일 장애점(SPOF)

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

SLA

서비스 수준 계약을 참조하세요.

SLI

서비스 수준 지표를 참조하세요.

SLO

서비스 수준 목표를 참조하세요.

분할 앤 시드 모델

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

SPOF

단일 장애점을 참조하세요.

스타 스키마

하나의 큰 팩트 테이블을 사용하여 트랜잭션 또는 측정된 데이터를 저장하고 하나 이상의 더 작은 차원 테이블을 사용하여 데이터 속성을 저장하는 데이터베이스 조직 구조입니다. 이 구조는 데이터 웨어하우스에서 또는 비즈니스 인텔리전스 목적으로 사용하도록 설계되었습니다.

Strangler Fig 패턴

레거시 시스템을 폐기할 수 있을 때까지 시스템 기능을 점진적으로 다시 작성하고 교체하여 모놀리식 시스템을 현대화하기 위한 접근 방식. 이 패턴은 무화과 덩굴이 나무로 자라 결국 숙주를 압도하고 대체하는 것과 비슷합니다. Martin 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 제어 및 데이터 획득(SCADA)

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

대칭 암호화

동일한 키를 사용하여 데이터를 암호화하고 복호화하는 암호화 알고리즘입니다.

합성 테스트

사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 감지하거나 성능을 모니터링하는 방식으로 진행되는 시스템 테스트입니다. Amazon CloudWatch Synthetics를 사용하여 이러한 테스트를 생성할 수 있습니다.

시스템 프롬프트

LLM에 컨텍스트, 명령 또는 지침을 제공하여 동작을 지시하는 기법입니다. 시스템 프롬프트는 컨텍스트를 설정하고 사용자와의 상호 작용을 위한 규칙을 설정하는 데 도움이 됩니다.

T

tags

AWS 리소스를 구성하기 위한 메타데이터 역할을 하는 키-값 페어입니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색, 필터링할 수 있습니다. 자세한 내용은 AWS 리소스에 태그 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 참조하세요.

훈련

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

Transit Gateway

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

트렁크 기반 워크플로

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

신뢰할 수 있는 액세스

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

튜닝

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

피자 두 판 팀

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

U

불확실성

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

차별화되지 않은 작업

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

상위 환경

환경을 참조하세요.

V

정리

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

버전 제어

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

VPC 피어링

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

취약성

시스템 보안을 손상시키는 소프트웨어 또는 하드웨어 결함입니다.

W

웜 캐시

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

웜 데이터

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

창 함수

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

워크로드

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

워크스트림

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

WORM

Write Once, Read Many(WORM)를 참조하세요.

WQF

AWS Workload Qualification Framework를 참조하세요.

Write Once Read Many(WORM)

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

Z

제로데이 익스플로잇

제로데이 취약성을 악용하는 공격(일반적으로 맬웨어)입니다.

제로데이 취약성

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

제로샷 프롬프팅

태스크를 수행하기 위해 LLM에 명령을 제공하지만 안내에 도움이 되는 예제(샷)는 제공하지 않습니다. LLM은 사전 훈련된 지식을 사용하여 태스크를 처리해야 합니다. 제로샷 프롬프팅의 효과는 태스크의 복잡성과 프롬프트의 품질에 따라 달라집니다. 퓨샷 프롬프팅도 참조하세요.

좀비 애플리케이션

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

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