aws

개발자 가이드

Amazon Managed Blockchain Query

Amazon Managed Blockchain Query: 개발자 가이드

Table of Contents

Amazon Managed Blockchain(AMB) 쿼리란 무엇입니까? 1
AMB 쿼리를 처음 사용하시나요? 1
주요 개념 2
Amazon Managed Blockchain(AMB) 쿼리 사용에 대한 고려 사항 및 제한 사항 2
설정 5
필수 조건 및 고려 사항 5
에 가입 AWS 5
적절한 권한을 가진 IAM 사용자 생성 5
설치 및 구성 AWS Command Line Interface 6
AWS Management Console 를 사용하여 AMB 쿼리를 사용하여 블록체인 쿼리 6
시작하기 7
IAM 정책 생성 7
Go를 사용하는 예제 8
Node.js 사용 예제 14
Python을 사용하는 예제 18
사용 예제 AWS Management Console 20
AMB 쿼리 사용 사례 21
현재 및 과거 토큰 밸런스 쿼리 21
기록 트랜잭션 데이터 검색 21
지정된 주소에 대한 모든 토큰 밸런스 가져오기 21
트랜잭션에 대해 내보낸 이벤트 나열 22
계약에서 빼낸 모든 토큰 가져오기 22
계약 나열 및 계약 정보 가져오기 22
AMB 쿼리 API 참조 23
보안 24
데이터 암호화 24
전송 중 암호화 25
ID 및 액세스 관리 25
대상 25
ID를 통한 인증 25
정책을 사용하여 액세스 관리 27
Amazon Managed Blockchain(AMB) 쿼리가 IAM과 작동하는 방식 28
ID 기반 정책 예시 33
문제 해결 37

API 사용량 지표 39
Amazon CloudWatch의 API 사용량 지표 39
문서 이력 41
..... xliii

Amazon Managed Blockchain(AMB) 쿼리란 무엇입니까?

Amazon Managed Blockchain(AMB)은 퍼블릭 및 프라이빗 블록체인 모두에서 복원력이 뛰어난 Web3 애플리케이션을 구축할 수 있도록 설계된 완전 관리형 서비스입니다. AMB Access를 사용하여 여러 블록체인에 대한 즉각적인 서버리스 액세스를 제공합니다. 특수 블록체인 인프라를 배포하고 블록체인 네트워크에 연결할 필요 없이 Web3-ready 애플리케이션을 구축합니다. AMB 쿼리를 사용하면 개발자 친화적인 API 작업을 사용하여 여러 블록체인의 실시간 및 기록 데이터에 액세스할 수 있습니다. 표준화된 블록체인 데이터는 특수 블록체인 인프라 또는 ETL(추출, 변환 및 로드) 없이도 AWS 서비스와 통합할 수 있습니다. 모든 AMB 기능은 기관 등급 및 메인스트림 소비자 애플리케이션 빌드에 맞게 안전하게 확장됩니다.

Amazon Managed Blockchain(AMB) 쿼리는 개발자 친화적인 API 작업을 통해 표준화된 다중 블록체인 데이터 세트에 대한 서버리스 액세스를 제공합니다. AMB 쿼리를 사용하면 블록체인 데이터를 구문 분석하고, 계약을 추적하고, 특수 인덱싱 인프라를 유지 관리하는 데 필요한 오버헤드 없이 하나 이상의 퍼블릭 블록체인에서 데이터가 필요한 애플리케이션을 신속하게 배송할 수 있습니다. 과거 토큰 밸런스에서 펑블 토큰 또는 펑블 토큰(NFTs)을 분석하거나, 지정된 Wallet 주소에 대한 트랜잭션 기록을 보거나, Ether와 같은 네이티브 암호화폐의 배포에 대한 데이터 분석을 수행하든 AMB 쿼리를 사용하면 블록체인 데이터에 액세스할 수 있습니다.

AMB 쿼리를 처음 사용하시나요?

AMB 쿼리를 처음 사용하는 경우 먼저 다음 섹션을 읽는 것이 좋습니다.

- 주요 개념: Amazon Managed Blockchain(AMB) 쿼리
- Amazon Managed Blockchain(AMB) 쿼리 설정
- Amazon Managed Blockchain(AMB) 쿼리 시작하기
- Amazon Managed Blockchain(AMB) 쿼리 사용 사례

주요 개념: Amazon Managed Blockchain(AMB) 쿼리

> **Note**
>
> 이 안내서에서는 필수 블록체인 개념에 익숙하다고 가정합니다. 이러한 개념에는 분산화, 토큰, 계약, 트랜잭션, proof-of-work, 지갑, 퍼블릭 및 프라이빗 키, 스테이킹, 채굴, 절반 등이 포함됩니다.

Amazon Managed Blockchain(AMB) 쿼리를 사용하면 다중 블록체인 네트워크 데이터에 편리하게 액세스할 수 있으므로 블록체인 활동과 관련된 컨텍스트 데이터를 더 쉽게 추출할 수 있습니다. AMB 쿼리를 사용하여 Bitcoin Mainnet 및 Ethereum Mainnet과 같은 퍼블릭 블록체인 네트워크에서 데이터를 읽을 수 있습니다. 주소의 현재 및 과거 잔액과 같은 정보를 가져오거나 지정된 기간 동안의 블록체인 트랜잭션 목록을 가져올 수도 있습니다. 또한 애플리케이션의 비즈니스 로직에서 추가로 분석하거나 사용할 수 있는 트랜잭션 이벤트와 같은 특정 트랜잭션의 세부 정보를 얻을 수 있습니다.

Amazon Managed Blockchain(AMB) 쿼리 사용에 대한 고려 사항 및 제한 사항

AMB 쿼리를 사용할 때는 다음 사항을 고려하세요.

- 사용 가능한 리전

  AMB 쿼리는 미국 동부(버지니아 북부) `us-east-1` 리전에서 지원됩니다.

- Service endpoints

  AMB 쿼리는 다음 엔드포인트를 사용하여 액세스할 수 있습니다.

  `https://managedblockchain-query.us-east-1.amazonaws.com`.

- 지원되는 블록체인 네트워크

  AMB 쿼리는 다음과 같은 퍼블릭 블록체인 네트워크를 지원합니다.

  - 비트코인 메인넷 proof-of-work 합의에 의해 보호되고 비트코인(BTC) 암호화폐가 발급 및 처리되는 퍼블릭 비트코인 블록체인 네트워크입니다. Mainnet의 트랜잭션은 실제 값(즉, 실제 비용이 발생함)을 가지며 퍼블릭 블록체인에 기록됩니다.

  - Bitcoin Testnet - Bitcoin Mainnet의 테스트넷입니다. 이 네트워크의 비트코인(BTC)은 Mainnet BTC와 별개이며 일반적으로 값이 없습니다.
  - 이더리움 메인넷 proof-of-stake 메인 네트워크입니다. Mainnet의 트랜잭션은 실제 값(즉, 실제 비용이 발생함)을 가지며 분산 원장에 기록됩니다.
  - Sepolia Testnet - Ethereum Mainnet의 테스트넷입니다. 이 네트워크의 Ether(ETH)는 Mainnet ETH와 별개이며 일반적으로 값이 없습니다.

- 지원되는 블록체인 토큰 및 계약

  AMB 쿼리는 다음과 같은 기본 및 표준 Ethereum 계약 토큰을 지원합니다.

  - 퍼블릭 블록체인 네이티브 토큰
    - 비트코인(BTC) - 비트코인 관련 블록체인의 기본 토큰입니다.
    - Ether(ETH) - Ethereum 관련 블록체인의 기본 토큰입니다.

  - 이더리움 계약 표준
    - ERC-20 토큰 표준 - ERC-20은 가용성 토큰의 표준입니다. 각 ERC-20 토큰을 다른 ERC-20 토큰과 정확히 동일하게 만드는(유형 및 값) 속성이 있습니다. 즉, 하나의 토큰은 이고 항상 다른 모든 토큰과 동일합니다. 자세한 내용은 Ethereum.org ERC-20 토큰 표준을 참조하세요.
    - ERC-721 식용 불가능한 토큰 표준 - ERC-721은 식용 불가능한 토큰(NFTs. 이 유형의 토큰은 고유하며 기간, 희귀성 또는 기타 속성으로 인해 동일한 계약의 다른 토큰과 다른 값을 가질 수 있습니다. 자세한 내용은 Ethereum.org ERC-721 토큰 표준을 참조하세요.

      ERC-1155 다중 토큰 표준 - ERC-1155는 원하는 수의 유효 및 비유효 토큰 유형을 표현하고 제어할 수 있는 계약 인터페이스를 생성하는 표준입니다. 이러한 방식으로 ERC-1155 토큰은 ERC-20 및 ERC-721 토큰과 동일하게 작동할 수 있으며, 동시에 둘 다와 같이 작동할 수도 있습니다. ERC-1155 토큰은 ERC-20 및 ERC-721 표준의 기능을 모두 개선하여 구현 오류를 수정하는 동시에 효율성을 높입니다. 자세한 내용은 Ethereum.org ERC-1155 토큰 표준을 참조하세요.

- 최종성

  블록체인에서 최종성은 유효한 트랜잭션이 반전될 가능성이 거의 없음을 의미합니다. Bitcoin Mainnet의 경우 AMB 쿼리는 6개 블록 후 트랜잭션 최종본을 고려합니다. Bitcoin Testnet의 경우 6 블록 또는 60분 중 먼저 도래하는 시점 이후에 트랜잭션 최종본을 고려합니다. 지원되는 Ethereum 네트워크의 경우 AMB 쿼리는 64개 블록 후 트랜잭션 최종본을 고려합니다.

AMB 쿼리의 토큰 밸런스 및 계약 API 작업은 최종 상태에 도달한 데이터만 반환합니다. 그러나 AMB 쿼리의 트랜잭션 및 트랜잭션 이벤트 API 작업은 아직 최종 상태에 도달하지 않았더라도 블록체인 네트워크에서 확인된 트랜잭션에 대한 데이터를 반환할 수 있습니다.

- NULL 주소가 지원되지 않음

  AMB 쿼리는 NULL (`0x0000000000000000000000000000000000000000`) 주소를 지원하지 않습니다.

- API 호출의 서명 버전 4 서명

  AMB 쿼리 APIs를 호출할 때 서명 버전 4 서명 프로세스를 사용하여 인증된 HTTPS 연결을 통해 호출할 수 있습니다. 즉, 계정의 AWS 승인된 IAM 보안 주체만 AMB 쿼리 API를 호출할 수 있습니다. 이렇게 하려면 호출과 함께 AWS 자격 증명(액세스 키 ID 및 보안 액세스 키)을 제공해야 합니다.

  > **Important**
  >
  > 사용자 대면 애플리케이션에 클라이언트 자격 증명을 포함시키지 마십시오.

- AMB 쿼리에서 Bitcoin 트랜잭션 식별자 및 트랜잭션 해시 지원

  Bitcoin 네트워크의 경우 AMB 쿼리 API 작업은 트랜잭션 식별자(`transactionId`)와 트랜잭션 해시()를 모두 지원합니다`transactionHash`. 는 감시 데이터를 포함하지 않는 트랜잭션의 이중 SHA 해시`transactionId`입니다. `transactionHash`는 감시 데이터(관찰 트랜잭션 ID라고도 함)를 포함한 트랜잭션의 이중 SHA 해시입니다.

  Bitcoin 네트워크에 대해 `GetTransaction` 또는 `ListTransactionEvents` API 작업을 호출할 때 `transactionId` 또는를 지정할 수 있습니다`transactionHash`. 또한 `transactionId` 또는를 반환하는 Bitcoin 네트워크의 모든 AMB 쿼리 작업`transactionHash`에는 응답의 일부로 두 값이 모두 포함됩니다.

Amazon Managed Blockchain(AMB) 쿼리 설정

Amazon Managed Blockchain(AMB) 쿼리를 처음 사용하기 전에이 섹션의 단계에 따라 AWS 계정을 생성합니다. 다음 섹션에서는 AMB 쿼리 사용을 시작하는 방법을 설명합니다.

필수 조건 및 고려 사항

Amazon Web Services를 처음 사용하기 전에 AWS 계정이 있어야 합니다.

에 가입 AWS

Amazon Web Services(AWS)에 가입하면 Amazon Managed Blockchain(AMB) 쿼리를 AWS 서비스포함한 모든에 AWS 계정이 자동으로 등록됩니다. 사용한 서비스에 대해서만 청구됩니다.

가 AWS 계정 이미 있는 경우 다음 단계로 이동합니다. AWS 계정이 없는 경우에는 다음 절차에 따라 계정을 만드세요.

AWS 계정을 생성하려면

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

적절한 권한을 가진 IAM 사용자 생성

AMB 쿼리를 생성하고 사용하려면 필요한 관리형 블록체인 작업을 허용하는 권한이 있는 AWS Identity and Access Management (IAM) 보안 주체(사용자 또는 그룹)를 생성해야 합니다.

IAM 보안 주체만 AMB 쿼리 API 요청을 할 수 있습니다. AMB 쿼리 APIs를 호출할 때 서명 버전 4 서명 프로세스를 사용하여 인증된 HTTPS 연결을 통해 호출할 수 있습니다. 즉, 계정의 AWS 승인된 IAM 보

안 주체만 AMB 쿼리 API를 호출할 수 있습니다. 이렇게 하려면 호출과 함께 AWS 자격 증명(액세스 키 ID 및 보안 액세스 키)을 제공해야 합니다.

IAM 사용자를 생성하는 방법에 대한 자세한 내용은 계정에서 IAM 사용자 생성을 참조하세요 AWS. 사용자에게 권한 정책을 연결하는 방법에 대한 자세한 내용은 IAM 사용자의 권한 변경을 참조하세요. 사용자에게 AMB 쿼리 작업 권한을 부여하는 데 사용할 수 있는 권한 정책의 예는 섹션을 참조하세요Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제.

설치 및 구성 AWS Command Line Interface

아직 설치하지 않은 경우 터미널의 AWS 리소스로 작업할 최신 AWS 명령줄 인터페이스(CLI)를 설치합니다. 자세한 내용은 최신 버전의 AWS CLI설치 또는 업데이트를 참조하세요.

> **Note**
>
> CLI 액세스를 위해서는 액세스 키 ID 및 비밀 액세스 키가 필요합니다. 가능하다면 장기 액세스 키 대신 임시 보안 인증 정보를 사용하세요. 임시 보안 인증도 액세스 키 ID와 비밀 액세스 키로 구성되지만 보안 인증이 만료되는 시간을 나타내는 보안 토큰이 포함되어 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용을 참조하세요.

AWS Management Console 를 사용하여 Amazon Managed Blockchain(AMB) 쿼리를 사용하여 블록체인 쿼리

를 사용하여 Amazon Managed Blockchain(AMB) 쿼리에 액세스하고 지원되는 블록체인 네트워크에서 쿼리를 수행할 수 있습니다 AWS Management Console. 다음 단계에서는이 작업을 수행하는 방법을 보여줍니다.

1. https://console.aws.amazon.com/managedblockchain/ Amazon Managed Blockchain 콘솔을 엽니다.
2. 쿼리 섹션에서 쿼리 편집기를 선택합니다.
3. 지원되는 블록체인 네트워크 중 하나를 선택합니다.
4. 실행할 쿼리 유형을 선택합니다.
5. 선택한 쿼리 유형 및 쿼리 실행에 대한 관련 파라미터를 입력합니다.

AMB 쿼리는 쿼리를 실행하며 쿼리 결과 창에 결과가 표시됩니다.

Amazon Managed Blockchain(AMB) 쿼리 시작하기

이 섹션의 step-by-step 자습서를 사용하여 Amazon Managed Blockchain(AMB) 쿼리를 사용하여 작업을 수행하는 방법을 알아봅니다. 이러한 절차에는 몇 가지 사전 조건이 필요합니다. AMB 쿼리를 처음 사용하는 경우이 가이드의 설정 섹션을 검토할 수 있습니다. 자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 설정 단원을 참조하십시오.

> **Note**
>
> 이 예제의 일부 변수는 의도적으로 난독화되었습니다. 이 예제를 실행하기 전에 해당 예제를 유효한 것으로 바꿉니다.

주제

- AMB 쿼리 API 작업에 액세스하기 위한 IAM 정책 생성
- Go를 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성
- Node.js를 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성
- Python을 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성
- 에서 Amazon Managed Blockchain(AMB) 쿼리 AWS Management Console 를 사용하여 GetTokenBalance 작업 실행

AMB 쿼리 API 작업에 액세스하기 위한 IAM 정책 생성

AMB 쿼리 API 요청을 하려면 Amazon Managed Blockchain(AMB) 쿼리에 대한 적절한 IAM 권한이 있는 사용자 자격 증명(AWS_ACCESS_KEY_ID 및 AWS_SECRET_ACCESS_KEY)을 사용해야 합니다. 이 AWS CLI 설치된 터미널에서 다음 명령을 실행하여 AMB 쿼리 API 작업에 액세스하는 IAM 정책을 생성합니다.

```
cat <<EOT > ~/amb-query-access-policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid" : "AMBQueryAccessPolicy",
            "Effect": "Allow",
```

```
            "Action": [
                "managedblockchain-query:*"
            ],
            "Resource": "*"
        }
    ]
}
EOT
aws iam create-policy --policy-name AmazonManagedBlockchainQueryAccess --policy-
document file://$HOME/amb-query-access-policy.json
```

정책을 생성한 후 해당 정책을 IAM 사용자의 역할에 연결하면 정책이 적용됩니다. 에서 IAM 서비스로 AWS Management Console이동하여 서비스를 사용할 IAM 사용자에게 할당된 `AmazonManagedBlockchainQueryAccess` 역할에 정책을 연결합니다. 자세한 내용은 역할 생성 및 IAM 사용자에게 할당을 참조하세요.

> **Note**
>
> AWS 에서는 와일드카드를 사용하는 대신 특정 API 작업에 대한 액세스 권한을 부여할 것을 권장합니다*. 자세한 내용은 특정 Amazon Managed Blockchain(AMB) 쿼리 API 작업에 액세스 단원을 참조하십시오.

Go를 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성

Amazon Managed Blockchain(AMB) 쿼리를 사용하면 블록체인에서 확정된 블록체인 데이터에 대한 즉각적인 액세스에 의존하는 애플리케이션을 빌드할 수 있습니다. 이는 아직 완료되지 않은 경우에도 마찬가지입니다. AMB 쿼리를 사용하면 Wallet의 트랜잭션 기록을 채우거나, 트랜잭션 해시를 기반으로 트랜잭션에 대한 컨텍스트 정보를 제공하거나, 네이티브 토큰과 ERC-721, ERC-1155 및 ERC-20 토큰의 균형을 확보하는 등 여러 사용 사례를 사용할 수 있습니다.

다음 예제는 Go 언어로 생성되며 AMB 쿼리 API 작업을 사용합니다. Go에 대한 자세한 내용은 Go 설명서를 참조하세요. AMB 쿼리 API에 대한 자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 API 참조 설명서를 참조하세요.

다음 예제에서는 `ListTransactions` 및 `GetTransaction` API 작업을 사용하여 먼저 Ethereum Mainnet의 지정된 외부 소유 주소(EOA)에 대한 모든 트랜잭션 목록을 가져온 다음, 다음 예제에서는 목록에서 단일 트랜잭션에 대한 트랜잭션 세부 정보를 검색합니다.

Example- Go를 사용하여 ListTransactions API 작업 수행

다음 코드를 ListTransactions 디렉터리`listTransactions.go`의 파일에 복사합니다.

```
package main

import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
    "time"
)

func main() {

    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // Inputs for ListTransactions API
    ownerAddress := "0x00000bf26964af9d7eed9e03e53415d********"
    network := managedblockchainquery.QueryNetworkEthereumMainnet
    sortOrder := managedblockchainquery.SortOrderAscending
    fromTime := time.Date(1971, 1, 1, 1, 1, 1, 1, time.UTC)
    toTime := time.Now()
    nonFinal := "NONFINAL"
    // Call ListTransactions API. Transactions that have reached finality are always
 returned
    listTransactionRequest, listTransactionResponse :=
 client.ListTransactionsRequest(&managedblockchainquery.ListTransactionsInput{
        Address: &ownerAddress,
        Network: &network,
        Sort: &managedblockchainquery.ListTransactionsSort{
            SortOrder: &sortOrder,
        },
        FromBlockchainInstant: &managedblockchainquery.BlockchainInstant{
            Time: &fromTime,
        },
        ToBlockchainInstant: &managedblockchainquery.BlockchainInstant{
```

```
            Time: &toTime,
        },

        ConfirmationStatusFilter: &managedblockchainquery.ConfirmationStatusFilter{
            Include: []*string{&nonFinal},
          },
    })
    errors := listTransactionRequest.Send()

    if errors == nil {
        // handle API response
        fmt.Println(listTransactionResponse)
    } else {
        // handle API errors
        fmt.Println(errors)
    }
}
```

파일을 저장한 후 ListTransactions 디렉터리 내에서 다음 명령을 사용하여 코드를 실행합니다go run listTransactions.go.

다음 출력은 다음과 유사합니다.

```
{
  Transactions: [
    {
      ConfirmationStatus: "FINAL",
      Network: "ETHEREUM_MAINNET",
      TransactionHash:
 "0x12345ea404b45323c0cf458ac755ecc45985fbf2b18e2996af3c8e8693354321",
      TransactionTimestamp: 2020-06-01 01:59:11 +0000 UTC
    },
    {
      ConfirmationStatus: "FINAL",
      Network: "ETHEREUM_MAINNET",
      TransactionHash:
 "0x1234547c65675d867ebd2935bb7ebe0996e9ec8e432a579a4516c7113bf54321",
      TransactionTimestamp: 2021-09-01 20:06:59 +0000 UTC
    },
     {
      ConfirmationStatus: "NONFINAL",
      Network: "ETHEREUM_MAINNET",
```

```
      TransactionHash:
 "0x123459df7c1cd42336cd1c444cae0eb660ccf13ef3a159f05061232a24954321",
      TransactionTimestamp: 2024-01-23 17:10:11 +0000 UTC
    }
  ]
}
```

Example- Go를 사용하여 GetTransaction API 작업 수행

이 예제에서는 이전 출력의 트랜잭션 해시를 사용합니다. 다음 코드를 GetTransaction 디렉터리GetTransaction.go의 파일에 복사합니다.

```
package main

import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
)

func main() {

    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // inputs for GetTransaction API
    transactionHash :=
 "0x123452695a82868950d9db8f64dfb2f6f0ad79284a6c461d115ede8930754321"
    network := managedblockchainquery.QueryNetworkEthereumMainnet

    // Call GetTransaction API. This operation will return transaction details for all
    // transactions that are con#rmed on the blockchain, even if they have not
    // reached #nality.
    getTransactionRequest, getTransactionResponse :=
 client.GetTransactionRequest(&managedblockchainquery.GetTransactionInput{
        Network:         &network,
        TransactionHash: &transactionHash,
```

```
    })

    errors := getTransactionRequest.Send()
    if errors == nil {
        // handle API response
        fmt.Println(getTransactionResponse)
    } else {
        // handle API errors
        fmt.Println(errors)
    }
}
```

파일을 저장한 후 GetTransaction 디렉터리 내에서 다음 명령을 사용하여 코드를 실행합니다`go run GetTransaction.go`.

다음 출력은 다음과 유사합니다.

```
{
  Transaction: {
    BlockHash: "0x000005c6a71d1afbc005a652b6ceca71cd516d97b0fc514c2a1d0f2ca3912345",
    BlockNumber: "11111111",
    CumulativeGasUsed: "5555555",
    EffectiveGasPrice: "44444444444",
    From: "0x9157f4de39ab4c657ad22b9f19997536********",
    GasUsed: "22222",
    Network: "ETHEREUM_MAINNET",
    NumberOfTransactions: 111,
    SignatureR: "0x99999894fd2df2d039b3555dab80df66753f84be475069dfaf6c6103********",
    SignatureS: "0x77777a101e7f37dd2dd0bf878b39080d5ecf3bf082c9bd4f40de783e********",
    SignatureV: 0,
    ConfirmationStatus: "FINAL",
    ExecutionStatus: "SUCCEEDED",
    To: "0x5555564f282bf135d62168c1e513280d********",
    TransactionHash:
 "0x123452695a82868950d9db8f64dfb2f6f0ad79284a6c461d115ede8930754321",
    TransactionIndex: 11,
    TransactionTimestamp: 2022-02-02 01:01:59 +0000 UTC
  }
}
```

GetTokenBalance API는 특정 시점에 외부 소유 계정(EOA)의 현재 밸런스를 가져오는 데 사용할 수 있는 네이티브 토큰(ETH 및 BTC)의 밸런스를 얻을 수 있는 방법을 제공합니다.

Example- GetTokenBalance API 작업을 사용하여 Go에서 네이티브 토큰의 밸런스 가져오기

다음 예제에서는 `GetTokenBalance` API를 사용하여 Ethereum Mainnet에서 주소 Ether(ETH) 밸런스를 가져옵니다. 다음 코드를 GetTokenBalance 디렉터리`GetTokenBalanceEth.go`의 파일에 복사합니다.

```
package main

import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
)

func main() {
    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // inputs for GetTokenBalance API
    ownerAddress := "0xBeE510AF9804F3B459C0419826b6f225********"
    network := managedblockchainquery.QueryNetworkEthereumMainnet
    nativeTokenId  := "eth" //Ether on Ethereum mainnet

    // call GetTokenBalance API
    getTokenBalanceRequest, getTokenBalanceResponse :=
 client.GetTokenBalanceRequest(&managedblockchainquery.GetTokenBalanceInput{
        TokenIdentifier: &managedblockchainquery.TokenIdentifier{
            Network:         &network,
            TokenId: &nativeTokenId,
        },
        OwnerIdentifier: &managedblockchainquery.OwnerIdentifier{
            Address: &ownerAddress,
        },
    })
    errors := getTokenBalanceRequest.Send()

    if errors == nil {
```

```
        // process API response
        fmt.Println(getTokenBalanceResponse)
    } else {
        // process API errors
        fmt.Println(errors)
    }
}
```

파일을 저장한 후 GetTokenBalance 디렉터리 내에서 다음 명령을 사용하여 코드를 실행합니다go run GetTokenBalanceEth.go.

다음 출력은 다음과 유사합니다.

```
{
  AtBlockchainInstant: {
    Time: 2020-12-05 11:51:01 +0000 UTC
  },
  Balance: "4343260710",
  LastTransactionHash:
 "0x00000ce94398e56641888f94a7d586d51664eb9271bf2b3c48297a50a0711111",
  LastTransactionTime: 2023-03-14 18:33:59 +0000 UTC,
  OwnerIdentifier: {
    Address: "0x12345d31750D727E6A3a7B534255BADd********"
  },
  TokenIdentifier: {
    Network: "ETHEREUM_MAINNET",
    TokenId: "eth"
  }
}
```

Node.js를 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성

이러한 노드 예제를 실행하려면 다음 사전 조건이 적용됩니다.

1. 시스템에 노드 버전 관리자(nvm) 및 Node.js가 설치되어 있어야 합니다. OS에 대한 설치 지침은 여기에서 확인할 수 있습니다.
2. node --version 명령을 사용하여 노드 버전 14 이상을 사용하고 있는지 확인합니다. 필요한 경우 nvm install 14 명령을 사용한 다음 nvm use 14 명령을 사용하여 버전 14를 설치할 수 있습니다.

3. 환경 변수 AWS_ACCESS_KEY_ID 및 에는 계정과 연결된 자격 증명이 포함되어야 AWS_SECRET_ACCESS_KEY 합니다.

   다음 명령을 사용하여 이러한 변수를 클라이언트에서 문자열로 내보냅니다. 다음에서 강조 표시된 값을 IAM 사용자 계정의 적절한 값으로 바꿉니다.

```
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
```

> **Note**
>
> - 모든 사전 조건을 완료한 후 HTTPS를 통해 서명된 요청을 제출하여 Amazon Managed Blockchain(AMB) 쿼리 API 작업에 액세스하고 Node.js의 기본 https 모듈을 사용하여 요청하거나 AXIOS와 같은 타사 라이브러리를 사용하여 AMB 쿼리에서 데이터를 검색할 수 있습니다.
> - 이 예제에서는 Node.js용 타사 HTTP 클라이언트를 사용하지만 AWS JavaScript SDK를 사용하여 AMB 쿼리를 요청할 수도 있습니다.
> - 다음 예제에서는 Axios 및 AWS SigV4용 SDK 모듈을 사용하여 AMB 쿼리 API 요청을 수행하는 방법을 보여줍니다.

다음 package.json 파일을 로컬 환경의 작업 디렉터리에 복사합니다.

```
{
  "name": "amb-query-examples",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@aws-crypto/sha256-js": "^4.0.0",
    "@aws-sdk/credential-provider-node": "^3.360.0",
    "@aws-sdk/protocol-http": "^3.357.0",
    "@aws-sdk/signature-v4": "^3.357.0",
```

```
    "axios": "^1.4.0"
  }
}
```

Example- AMB 쿼리 GetTokenBalance API를 사용하여 특정 외부 소유 주소(EOA)에서 과거 토큰 밸런스를 검색합니다.

GetTokenBalance API를 사용하여 다양한 토큰(예: ERC20, ERC721 및 ERC1155)과 네이티브 코인(예: ETH 및 BTC)의 밸런스를 가져올 수 있습니다. 이를 사용하여 기록timestamp(Unix 타임스탬프 - 초)을 기반으로 외부 소유 계정(EOA)의 현재 밸런스를 가져올 수 있습니다. 이 예제에서는 GetTokenBalance API를 사용하여 Ethereum Mainnet에서 ERC20 토큰 USDC의 주소 밸런스를 가져옵니다.

GetTokenBalance API를 테스트하려면 다음 코드를 라는 파일에 복사token-balance.js하고 파일을 동일한 작업 디렉터리에 저장합니다.

```
const axios = require('axios').default;
const SHA256 = require('@aws-crypto/sha256-js').Sha256
const defaultProvider = require('@aws-sdk/credential-provider-node').defaultProvider
const HttpRequest = require('@aws-sdk/protocol-http').HttpRequest
const SignatureV4 = require('@aws-sdk/signature-v4').SignatureV4

// define a signer object with AWS service name, credentials, and region
const signer = new SignatureV4({
  credentials: defaultProvider(),
  service: 'managedblockchain-query',
  region: 'us-east-1',
  sha256: SHA256,
});

const queryRequest = async (path, data) => {
  //query endpoint
  let queryEndpoint = `https://managedblockchain-query.us-east-1.amazonaws.com/
${path}`;

  // parse the URL into its component parts (e.g. host, path)
  const url = new URL(queryEndpoint);

  // create an HTTP Request object
  const req = new HttpRequest({
    hostname: url.hostname.toString(),
    path: url.pathname.toString(),
```

```
    body: JSON.stringify(data),
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Accept-Encoding': 'gzip',
      host: url.hostname,
    }
  });


  // use AWS SignatureV4 utility to sign the request, extract headers and body
  const signedRequest = await signer.sign(req, { signingDate: new Date() });

  try {
    //make the request using axios
    const response = await axios({...signedRequest, url: queryEndpoint, data: data})

    console.log(response.data)
  } catch (error) {
    console.error('Something went wrong: ', error)
    throw error
  }


}


let methodArg = 'get-token-balance';

let dataArg = {
  " atBlockchainInstant": {
    "time": 1688071493
  },
  "ownerIdentifier": {
      "address": "0xf3B0073E3a7F747C7A38B36B805247B2********" // externally owned
 address
  },
  "tokenIdentifier": {
      "contractAddress":"0xA0b86991c6218b36c1d19D4a2e9Eb0cE********", //USDC contract
 address
      "network":"ETHEREUM_MAINNET"
  }
}
```

```
//Run the query request.
queryRequest(methodArg, dataArg);
```

코드를 실행하려면 파일과 동일한 디렉터리에서 터미널을 열고 다음 명령을 실행합니다.

```
npm i
node token-balance.js
```

이 명령은 스크립트를 실행하여 코드에 정의된 인수를 전달하여 Ethereum Mainnet에 나열된 EOA의 ERC20 USDC 밸런스를 요청합니다. 응답은 다음과 유사합니다.

```
{
 atBlockchainInstant: { time: 1688076218 },
 balance: '140386693440144',
 lastUpdatedTime: { time: 1688074727 },
 ownerIdentifier: { address: '0xf3b0073e3a7f747c7a38b36b805247b2********' },
 tokenIdentifier: {
   contractAddress: '0xa0b86991c6218b36c1d19d4a2e9eb0ce********',
   network: 'ETHEREUM_MAINNET'
 }
```

Python을 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청 생성

이러한 Python 예제를 실행하려면 다음 사전 조건이 적용됩니다.

1. 시스템에 Python이 설치되어 있어야 합니다. OS에 대한 설치 지침은 여기에서 확인할 수 있습니다.
2. Python용 AWS SDK(Boto3)를 설치합니다.
3. AWS 명령줄 인터페이스를 설치하고 명령을 실행`aws configure`하여 `Access Key ID`, `Secret Access Key`및에 대한 변수를 설정합니다`Region`.

모든 사전 조건을 완료한 후 HTTPS를 AWS 통한 Python용 SDK를 사용하여 Amazon Managed Blockchain(AMB) 쿼리 API 요청을 할 수 있습니다.

다음 Python 예제에서는 boto3의 모듈을 사용하여 필요한 SigV4 헤더가 첨부된 요청을 AMB 쿼리 `ListTransactionEvents` API 작업으로 보냅니다. 이 예제에서는 Ethereum Mainnet의 지정된 트랜잭션에서 내보낸 이벤트 목록을 검색합니다.

다음 `list-transaction-events.py` 파일을 로컬 환경의 작업 디렉터리에 복사합니다.

```
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.session import Session
from botocore.httpsession import URLLib3Session

def signed_request(url, method,  params, service, region):

    session = Session()
    sigv4 = SigV4Auth(session.get_credentials(), service, region)
    data = json.dumps(params)
    request = AWSRequest(method, url, data=data)
    sigv4.add_auth(request)
    http_session = URLLib3Session()
    response = http_session.send(request.prepare())

    return(response)

url = 'https://managedblockchain-query.us-east-1.amazonaws.com/list-transaction-events'
method = 'POST'
params = {
'network': 'ETHEREUM_MAINNET',
'transactionHash': '0x125714bb4db48757007fff2671b37637bbfd6d47b3a4757ebbd0c5222984f905'
}
service = 'managedblockchain-query'
region = 'us-east-1'

# Call the listTransactionEvents operation. This operation will return transaction
 details for
# all transactions that are con#rmed on the blockchain, even if they have not reached
# #nality.
listTransactionEvents = signed_request(url, method, params, service, region)

print(json.loads(listTransactionEvents.content.decode('utf-8')))
```

샘플 코드를에 실행하려면 작업 디렉터리에 파일을 ListTransactionEvents저장한 다음 명령을 실행합니다python3 list-transaction-events.py. 이 명령은 스크립트를 실행하여 코드에 정의된 인수를 전달하여 Ethereum Mainnet에서 지정된 트랜잭션 해시와 연결된 이벤트를 요청합니다. 응답은 다음과 유사합니다.

```
{
 'events':
```

```
 [
  {
      'contractAddress': '0x95ad61b0a150d79219dcf64e1e6cc01f********',
      'eventType': 'ERC20_TRANSFER',
      'from': '0xab5801a7d398351b8be11c439e05c5b3********',
      'network': 'ETHEREUM_MAINNET',
      'to': '0xdead000000000000000420694206942********',
      'transactionHash':
 '0x125714bb4db48757007fff2671b37637bbfd6d47b3a4757ebbd0c522********',
      'value': '4102419967718718947718261747554 64'
  }
 ]
}
```

에서 Amazon Managed Blockchain(AMB) 쿼리 AWS Management Console 를 사용하여 GetTokenBalance 작업 실행

다음 예제는에서 Amazon Managed Blockchain(AMB) 쿼리를 사용하여 Ethereum Mainnet에서 토큰의 밸런스를 가져오는 방법을 보여줍니다. AWS Management Console

Example

1. https://console.aws.amazon.com/managedblockchain/ Amazon Managed Blockchain 콘솔을 엽니다.
2. 쿼리 섹션에서 쿼리 편집기를 선택합니다.
3. ETHEREUM_MAINNET을 블록체인 네트워크로 선택합니다.
4. 쿼리 유형으로 GetTokenBalance를 선택합니다.
5. 토큰의 블록체인 주소를 입력합니다.
6. 토큰의 계약 주소를 입력합니다.
7. 토큰의 선택적 토큰 ID를 입력합니다.
8. 토큰 밸런스의 날짜를 선택합니다.
9. 토큰 밸런스에 대한 선택적 시를 입력합니다.
10. 쿼리 실행을 선택합니다.

AMB 쿼리는 쿼리를 실행하며 쿼리 결과 창에 결과가 표시됩니다.

Amazon Managed Blockchain(AMB) 쿼리 사용 사례

이 주제에서는 AMB 쿼리 사용 사례 목록을 제공합니다.

주제

- 현재 및 과거 토큰 밸런스 쿼리
- 기록 트랜잭션 데이터 검색
- 지정된 주소에 대한 모든 토큰 밸런스 가져오기
- 트랜잭션에 대해 내보낸 이벤트 나열
- 계약에서 빼낸 모든 토큰 가져오기
- 계약 나열 및 계약 정보 가져오기

현재 및 과거 토큰 밸런스 쿼리

`GetTokenBalance` API는 지원되는 토큰(ERC20, ERC721, ERC1155)과 네이티브 코인(ETH, BTC)의 밸런스를 가져와 외부 소유 계정(EOAs)의 범용 타임스탬프(Unix 타임스탬프, 초 단위)를 사용하여 현재 또는 과거 밸런스를 가져옵니다. 예를 들어 `GetTokenBalance` API 작업을 사용하여 Ethereum Mainnet에서 ERC20 토큰 USDC의 주소 밸런스를 가져올 수 있습니다. `BatchGetTokenBalance` API 작업을 사용하여 토큰과 네이티브 코인의 밸런스를 일괄 검색할 수도 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

기록 트랜잭션 데이터 검색

Amazon Managed Blockchain(AMB) 쿼리를 사용하면 Ethereum 및 Bitcoin과 같은 퍼블릭 블록체인에서 기록 데이터를 검색할 수 있습니다. 이 기능을 사용하면 블록체인 지갑에서 트랜잭션 기록을 검색하거나 트랜잭션 해시를 기반으로 트랜잭션에 대한 컨텍스트 정보를 제공하는 등 여러 사용 사례를 사용할 수 있습니다. `ListTransactions` API 작업을 사용하여 Ethereum Mainnet에서 지정된 외부 소유 주소(EOA)에 대한 트랜잭션 목록을 가져온 다음 `GetTransaction` API 작업을 사용하여 목록에서 단일 트랜잭션에 대한 트랜잭션 세부 정보를 검색할 수 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

지정된 주소에 대한 모든 토큰 밸런스 가져오기

ListTokenBalances API 작업을 사용하여 Wallets, 사용자 인터페이스, web3 유틸리티 등에 대한 밸런스를 얻을 수 있습니다. 이 API 작업은 단일 API 작업을 사용하여 지정된 퍼블릭 블록체인의 토큰(ERC20, ERC721, ERC1155) 및 네이티브 코인(ETH, BTC) 전반의 주소에 대한 모든 잔액 목록을 반환합니다. 예를 들어 외부 소유 주소(EOA)와 네트워크(Ethereum Mainnet)를 제공할 수 있으며, 응답에서 토큰 및 네이티브 코인 밸런스 목록을 받을 수 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

트랜잭션에 대해 내보낸 이벤트 나열

ListTransactionEvents API 작업을 사용하여 해시(트랜잭션 식별자)로 식별되는 지정된 트랜잭션의 결과로 생성되는 계약 이벤트 목록을 검색할 수 있습니다. 예를 들어 ListTransactionEvents를 사용하여 전송 이벤트 또는 ERC20 계약에서 철회 이벤트와 같이 이더리움 블록체인에서 ERC20 토큰 계약의 함수를 호출하는 트랜잭션의 결과 이벤트를 검색할 수 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

계약에서 빼낸 모든 토큰 가져오기

ListTokenBalances API 작업을 사용하여 계약 주소를 입력으로 전달할 때 계약에 의해 제거된 지원되는 모든 토큰(ERC20, ERC721, ERC1155) 목록을 반환할 수 있습니다. 예를 들어 API ListTokenBalances 작업을 사용하여 Ethereum 블록체인의 ERC721 계약 표준에 따라 민팅된 고식적 토큰(NFTs)과 관련된 정보를 검색할 수 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

계약 나열 및 계약 정보 가져오기

ListAssetContracts API 작업을 사용하여 지정된 주소로 배포된 ERC-721, ERC-1155 또는 ERC-20 계약을 나열할 수 있습니다. 또한 계약 주소가 있는 경우 GetAssetContract API 작업을 사용하여 계약 유형 배포자 주소 및 관련 토큰 메타데이터와 같은 계약의 속성을 검색할 수 있습니다.

자세한 내용은 Amazon Managed Blockchain(AMB) 쿼리 참조 안내서를 참조하세요.

Amazon Managed Blockchain(AMB) 쿼리 API 참조

Amazon Managed Blockchain(AMB) 쿼리는 지원되는 블록체인을 쿼리하기 위한 API 작업을 제공합니다. 여기에는 토큰, 트랜잭션 및 계약을 쿼리하기 위한 APIs가 포함됩니다. 자세한 내용은 AMB 쿼리 API 참조를 참조하세요.

Amazon Managed Blockchain(AMB) 쿼리의 보안

의 클라우드 보안 AWS 이 가장 우선합니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이를 클라우드의 보안과 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon Managed Blockchain(AMB) 쿼리에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 AWS 준수 프로그램 제공 범위 내 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 사용자는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

Amazon Managed Blockchain은 데이터 보호, 인증 및 액세스 제어를 제공하기 위해 Managed Blockchain에서 실행되는 오픈 소스 프레임워크의 기능과 기능을 사용합니다 AWS .

이 설명서는 AMB 쿼리를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 AMB 쿼리를 구성하는 방법을 보여줍니다. 또한 AMB 쿼리 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 배울 수 있습니다.

주제

- 데이터 암호화
- Amazon Managed Blockchain(AMB) 쿼리의 ID 및 액세스 관리

데이터 암호화

데이터 암호화는 권한이 없는 사용자가 블록체인 네트워크 및 연결된 데이터 스토리지 시스템에서 데이터를 읽는 것을 방지하는 데 도움이 됩니다. 여기에는 네트워크를 이동할 때 가로채질 수 있는 전송 중 데이터라고 하는 데이터가 포함됩니다.

전송 중 암호화

기본적으로 Managed Blockchain은 HTTPS/TLS 연결을 사용하여 AWS CLI 클라이언트에서 AWS 서비스 엔드포인트로 전송되는 모든 데이터를 암호화합니다.

Amazon Managed Blockchain(AMB) 쿼리의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 AMB 쿼리 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon Managed Blockchain(AMB) 쿼리가 IAM과 작동하는 방식
- Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제
- Amazon Managed Blockchain(AMB) 쿼리 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조Amazon Managed Blockchain(AMB) 쿼리 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(Amazon Managed Blockchain(AMB) 쿼리가 IAM과 작동하는 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS Sign-In 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한

정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon Managed Blockchain(AMB) 쿼리가 IAM과 작동하는 방식

IAM을 사용하여 AMB 쿼리에 대한 액세스를 관리하기 전에 AMB 쿼리와 함께 사용할 수 있는 IAM 기능을 알아봅니다.

Amazon Managed Blockchain(AMB) 쿼리와 함께 사용할 수 있는 IAM 기능

IAM 특성	AMB 쿼리 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요

IAM 특성	AMB 쿼리 지원
정책 작업	예
정책 리소스	아니요
정책 조건 키	아니요
ACL	아니요
ABAC(정책 내 태그)	아니요
임시 보안 인증	예
엔터티 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

AMB 쿼리 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

AMB 쿼리에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

AMB 쿼리에 대한 자격 증명 기반 정책 예제

AMB 쿼리 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제.

AMB 쿼리 내 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

AMB 쿼리에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

AMB 쿼리 작업 목록을 보려면 서비스 승인 참조의 Amazon Managed Blockchain(AMB) 쿼리에서 정의한 작업을 참조하세요.

AMB 쿼리의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
managedblockchain-query:
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
      "managedblockchain-query::ListTransaction",
      "managedblockchain-query::GetTransaction"
         ]
```

AMB 쿼리 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제.

AMB 쿼리에 대한 정책 리소스

정책 리소스 지원: 아니요

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

AMB 쿼리 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의 Amazon Managed Blockchain(AMB) 쿼리에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Managed Blockchain(AMB) 쿼리에서 정의한 작업을 참조하세요.

AMB 쿼리 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제.

AMB 쿼리에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 아니요

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

AMB 쿼리 조건 키 목록을 보려면 서비스 승인 참조의 Amazon Managed Blockchain(AMB) 쿼리에 대한 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 Amazon Managed Blockchain(AMB) 쿼리에서 정의한 작업을 참조하세요.

AMB 쿼리 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제.

ACLs

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

AMB 쿼리가 포함된 ABAC

ABAC 지원(정책의 태그): 아니요

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

AMB 쿼리에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 전환 역할을 사용할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요.

AMB 쿼리에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 대한 요청 AWS 서비스 과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

AMB 쿼리에 대한 서비스 역할

서비스 역할 지원: 아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 AMB 쿼리 기능이 중단될 수 있습니다. AMB 쿼리가 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

AMB 쿼리에 대한 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

Amazon Managed Blockchain(AMB) 쿼리에 대한 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 AMB 쿼리 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 AMB 쿼리에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon Managed Blockchain(AMB) 쿼리에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 특정 Amazon Managed Blockchain(AMB) 쿼리 API 작업에 액세스

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 AMB 쿼리 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

특정 Amazon Managed Blockchain(AMB) 쿼리 API 작업에 액세스

> **Note**
>
> AMB 쿼리에 액세스하여 API를 호출하려면 AMB 쿼리에 대한 적절한 IAM 권한이 있는 사용자 자격 증명(`AWS_ACCESS_KEY_ID` 및 `AWS_SECRET_ACCESS_KEY`)이 필요합니다.

Example모든 Amazon Managed Blockchain(AMB) 쿼리 APIs에 액세스하기 위한 IAM 정책

이 예제에서는의 IAM 사용자에게 모든 AMB 쿼리 APIs에 대한 AWS 계정 액세스 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAllAMBQueryAPIs",
            "Effect": "Allow",
            "Action": [
                "managedblockchain-query:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Amazon Managed Blockchain(AMB) 쿼리 **`ListTransactions`** 및 **`GetTransaction`** APIs에 액세스하기 위한 IAM 정책

이 예제에서는의 IAM 사용자에게 AMB 쿼리 `ListTransaction` 및 `GetTransaction` API에 대한 AWS 계정 액세스 권한을 부여합니다. APIs

> **Note**
>
> 예제의 APIs를 다른 APIs로 바꾸거나 추가하여 다른 API에 대한 액세스 권한을 부여할 수 APIs. AMB 쿼리 APIs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAMBQueryAPIs",
            "Effect": "Allow",
            "Action": [
                "managedblockchain-query:ListTransactions",
                "managedblockchain-query:GetTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Managed Blockchain(AMB) 쿼리 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 AMB 쿼리 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- AMB 쿼리에서 작업을 수행할 권한이 없음

AMB 쿼리에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `managedblockchain-query:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 managedblockchain-query:GetWidget on resource: my-example-widget
```

이 경우, `managedblockchain-query::GetWidget` 작업을 사용하여 `my-example-widget` 리소스에 액세스할 수 있도록 `mateojackson` 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

Amazon CloudWatch의 Amazon Managed Blockchain(AMB) 쿼리 API 사용량 지표

Amazon CloudWatch의 API 사용량 지표

CloudWatch에 게시된 API 사용량 지표는 Amazon Managed Blockchain(AMB) 쿼리 서비스 할당량에 해당합니다. 사용량이 서비스 할당량에 가까워지면 알림을 보내도록 경보를 구성할 수 있습니다. Service Quotas와 CloudWatch의 통합에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 AWS 사용량 지표를 참조하세요.

AMB 쿼리는 `Amazon Managed Blockchain Query` 서비스 이름과 함께 네`AWS/Usage`임스페이스에 다음 API 지표를 게시합니다.

지표	설명
`CallCount`	AMB 쿼리에서 API에 대한 총 호출 수입니다. SUM은 지정된 기간 동안 API에 대한 총 직접 호출 수를 나타냅니다.

Amazon Managed Blockchain(AMB) 쿼리는 다음 차원을 사용하여 사용량 지표를 `AWS/Usage` 네임스페이스에 게시합니다.

차원	설명
Service	리소스가 포함된 AWS 서비스의 이름입니다. `Amazon Managed Blockchain Query` 는 항상이 차원의 값입니다.
Type	보고되는 개체의 유형입니다. `API`는 항상이 차원의 값입니다.
Resource	보고되는 리소스의 유형입니다. 사용되는 AMB 쿼리 API 작업의 이름은이 차원의 값이 됩니다.

차원	설명
Class	보고되는 리소스의 클래스입니다. None는 항상 이 차원의 값이 됩니다.

AMB 쿼리 사용 설명서의 문서 기록

다음 표에서는 AMB 쿼리에 대한 설명서 릴리스를 설명합니다.

변경 사항	설명	날짜
AMB 쿼리에서 Bitcoin 트랜잭션 식별자 및 트랜잭션 해시 지원	Bitcoin 네트워크의 경우 AMB 쿼리 API 작업은 트랜잭션 식별자(`transactionId`)와 트랜잭션 해시()를 모두 지원합니다`transactionHash` .	2024년 3월 21일
Amazon CloudWatch에서 API 사용 지표 지원	AMB 쿼리에 CloudWatch의 API 사용량 지표에 대한 지원이 추가되었습니다. 이러한 사용량 지표는 AMB 쿼리 서비스 할당량에 해당합니다.	2024년 2월 8일
최종 상태에 도달하지 않은 트랜잭션에 대한 지원	AMB 쿼리에 최종에 도달하지 않은 트랜잭션에 대한 지원이 추가되었습니다. 또한 `GetTransaction` 작업의 응답에서 `status` 속성에 대한 지원을 제거합니다. 대신 `confirmationStatus` 및 `executionStatus` 속성을 사용하여 트랜잭션의 상태를 결정합니다.	2024년 2월 1일
트랜잭션 데이터 유형의 `status` 속성 사용 중단	Amazon Managed Blockchain(AMB) 쿼리는 트랜잭션 데이터 유형의 `status` 속성을 더 이상 사용하지 않습니다. `confirmationStatus` 및 `executionStatus` 필드를 사용하여 트랜잭션`status`의	2023년 12월 20일

	가 FINAL 또는 인지 확인해야 합니다FAILED.	
Sepolia Testnet 지원	Amazon Managed Blockchain(AMB) 쿼리는 이제 Ethereum Sepolia Testnet에서 쿼리를 지원합니다.	2023년 10월 19일
자산 계약 지원	ListAssetContracts API 작업을 사용하여 지정된 주소별로 배포된 목록을 나열할 수 있습니다. 또한 계약 주소가 있는 경우 GetAssetContract API 작업을 사용하여 계약 세부 정보를 검색할 수 있습니다.	2023년 10월 16일
Bitcoin Testnet 지원	Amazon Managed Blockchain(AMB) 쿼리는 이제 Bitcoin Testnet에서 쿼리를 지원합니다.	2023년 10월 16일
최초 릴리스	AMB 쿼리 서비스의 최초 릴리스입니다.	2023년 7월 27일

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