AWS Secrets Manager 및 HashiCorp Terraform을 사용하여 민감한 데이터 보호

AWS 권장 가이드

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

AWS 권장 가이드: AWS Secrets Manager 및 HashiCorp Terraform을 사용하여 민감한 데이터 보호

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

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

Table of Contents

소개 1
대상 독자 1
목표 1
Secrets Manager 및 사용 Terraform 3
AWS Secrets Manager 3
Terraform 3
모범 사례 4
무작위 암호 생성 4
보안 암호 교체 6
액세스 제한 8
이름 지정 규칙 사용 9
민감한 데이터 관리 10
Terraform 상태 파일에서 민감한 데이터 보호 10
Amazon EKS의 보안 암호 관리 10
알려진 네트워크에 민감한 데이터 유지 13
리소스 15
AWS 설명서 15
AWS 블로그 게시물 15
기타 리소스 15
기여자 16
작성 16
검토 16
기술 작성 16
문서 기록 17
..... xviii

AWS Secrets Manager 및를 사용하여 민감한 데이터 보호 HashiCorp Terraform

Amazon Web Services(기여자)

2025년 5월(문서 기록)

자격 증명, 비밀 문자열 및 암호를 포함한 민감한 데이터의 관리는 인프라 관리 및 애플리케이션 개발 및 배포의 인식되는 원칙입니다. 조직을 보호하려면 클라우드에서 민감한 데이터를 관리하기 위한 모범 사례를 채택하십시오. 민감한 데이터를 보호하는 것은 보안 및 규정 준수를 위한 사전 조건입니다. 는 환경에서 민감한 데이터를 보안 암호로 보호하는 데 도움이 될 AWS Secrets Manager 수 있습니다.

이 가이드에서는 Secrets Manager에서 보안 암호를 가져오는 방법,를 사용하여 민감한 데이터의 보안 암호를 AWS Lambda 자동으로 교체하는 방법 등 보안 암호에 대한 모범 사례를 검토합니다. 또한 계층적 이름을 사용하여 보안 암호를 관리하고 관리하는 방법에 대한 권장 사항을 제공합니다. 마지막으로 중앙 집중화, Terraform 통합 및 네트워킹 고려 사항과 같은 보안 암호의 사용 및 액세스를 관리하는 데 도움이 됩니다.

HashiCorp Terraform는 업계에서 코드형 인프라(IaC) 솔루션으로 광범위하게 채택되었습니다. 그러나 는 민감한 데이터를 상태 파일에 일반 텍스트로 Terraform 표시합니다. 이 가이드에는를 사용하여 민감한 데이터를 Terraform 관리하고 Secrets Manager 보안 암호를 생성 및 사용하는 모범 사례가 포함되어 있습니다.

대상 독자

이 가이드는를 IaC 솔루션으로 사용하려는 조직을 대상으로 Terraform 합니다. 이 가이드의 모범 사례는 데이터베이스 아키텍트, 인프라 팀 및 애플리케이션 개발자를 돕기 위해 설계되었습니다. 이 가이드의 사전 조건은에 대한 지식Terraform입니다.

목표

다음은이 가이드의 권장 사항을 구현한 후 달성할 것으로 예상되는 비즈니스 성과입니다.

- 보안 암호 관리 프로세스를 자동화하여 더 빠르게 혁신합니다.
- 에서 조직의 보안 태세를 개선합니다 AWS 클라우드.

다음은이 가이드의 권장 사항을 구현한 후 달성할 것으로 예상되는 기술적 성과입니다.

- Secrets Manager를 사용하면 Terraform 상태 파일에서 민감한 데이터가 노출되는 것을 방지할 수 있습니다.
- 거버넌스를 개선하고 규정 준수를 달성하기 위해 보안 암호 및 민감한 데이터의 관리를 중앙 집중화합니다.
- 조직의 클라우드 인프라 배포 프로세스에 보안 모범 사례를 적용합니다.

Secrets Manager 및 사용 Terraform

AWS Secrets Manager

AWS Secrets Manager는 데이터베이스 및 기타 서비스에 대한 보안 인증을 안전하게 암호화, 저장 및 교체하기 위한 서비스입니다. 암호를 포함하여 코드의 하드 코딩된 자격 증명을 프로그래밍 방식으로 보안 암호를 검색하는 API 호출로 대체하는 데 도움이 됩니다. Secrets Manager에서 보안 암호는 자격 증명 정보(보안 암호 값)와 메타데이터로 구성됩니다. 보안 암호 값은 바이너리, 단일 문자열 또는 여러 문자열일 수 있습니다. 자세한 내용은 보안 암호를 참조하세요.

Secrets Manager는 256비트 Advanced Encryption Standard(AES) 대칭 데이터 키를 사용하여 보안 암호 값을 암호화합니다. 자세한 내용은 AWS Secrets Manager에서 시크릿 암호화 및 복호화를 참조하세요.

다음 방법 중 하나를 사용하여 Secrets Manager에 액세스하고 작업할 수 있습니다.

- Secrets Manager 콘솔
- 명령줄 도구
- AWS SDKs
- Secrets Manager API라고도 하는 HTTPS 쿼리 API
- AWS Secrets Manager 엔드포인트

Terraform

Terraform는 클라우드 및 온프레미스 리소스를 생성하고 관리하는 데 도움이 되는 HashiCorp의 IaC 도구입니다. HashiCorp Terraform를 사용하여에 리소스와 인프라를 배포할 수 있습니다 AWS 클라우드.

Terraform는 관리형 AWS 인프라 및 해당 구성에 대한 정보를 저장합니다. 이 정보를 상태라고 합니다. 기본적으로 상태는 라는 로컬 파일에 저장됩니다`Terraform.tfstate`. 이 파일은 JSON 형식이며이 상태 파일에 민감한 데이터를 일반 텍스트로 저장할 Terraform 수 있습니다. 이렇게 하면 상태 파일에 액세스할 수 있는 모든 사용자가 민감한 데이터에 액세스할 수 있기 때문에 민감한 데이터에 위험이 있습니다.

이 가이드에서는를 사용하여 AWS 리소스를 관리할 때 민감한 데이터를 보호하는 Terraform 데 도움이 되는 모범 사례와 권장 사항을 제공합니다.

보안 암호 생성, 교체 및 사용에 대한 모범 사례

이 섹션에서는를 통해 배포할 애플리케이션 및 인프라를 코딩할 때 따를 수 있는 모범 사례와 표준에 대해 설명합니다Terraform. 이는 다음을 포함합니다.

- 를 사용하여 무작위 암호 생성 AWS Secrets Manager - 조직의 규정 준수 정책에 따라 지정된 복잡성의 무작위 보안 암호를 생성합니다.
- AWS Lambda 를 사용하여 보안 암호 교체 - 보안 암호 교체를 위해 자동으로 실행되도록 예약할 AWS Lambda 수 있습니다.
- 보안 암호에 대한 액세스 제한 - 보안 암호에 액세스할 수 있는 사용자와 보안 암호에 액세스하는 방법을 제한합니다.
- 보안 암호에 대한 계층적 이름 지정 규칙 사용 - 보안 암호를 대규모로 관리하는 데 도움이 되는 이름 지정 규칙을 정의하고 사용합니다.

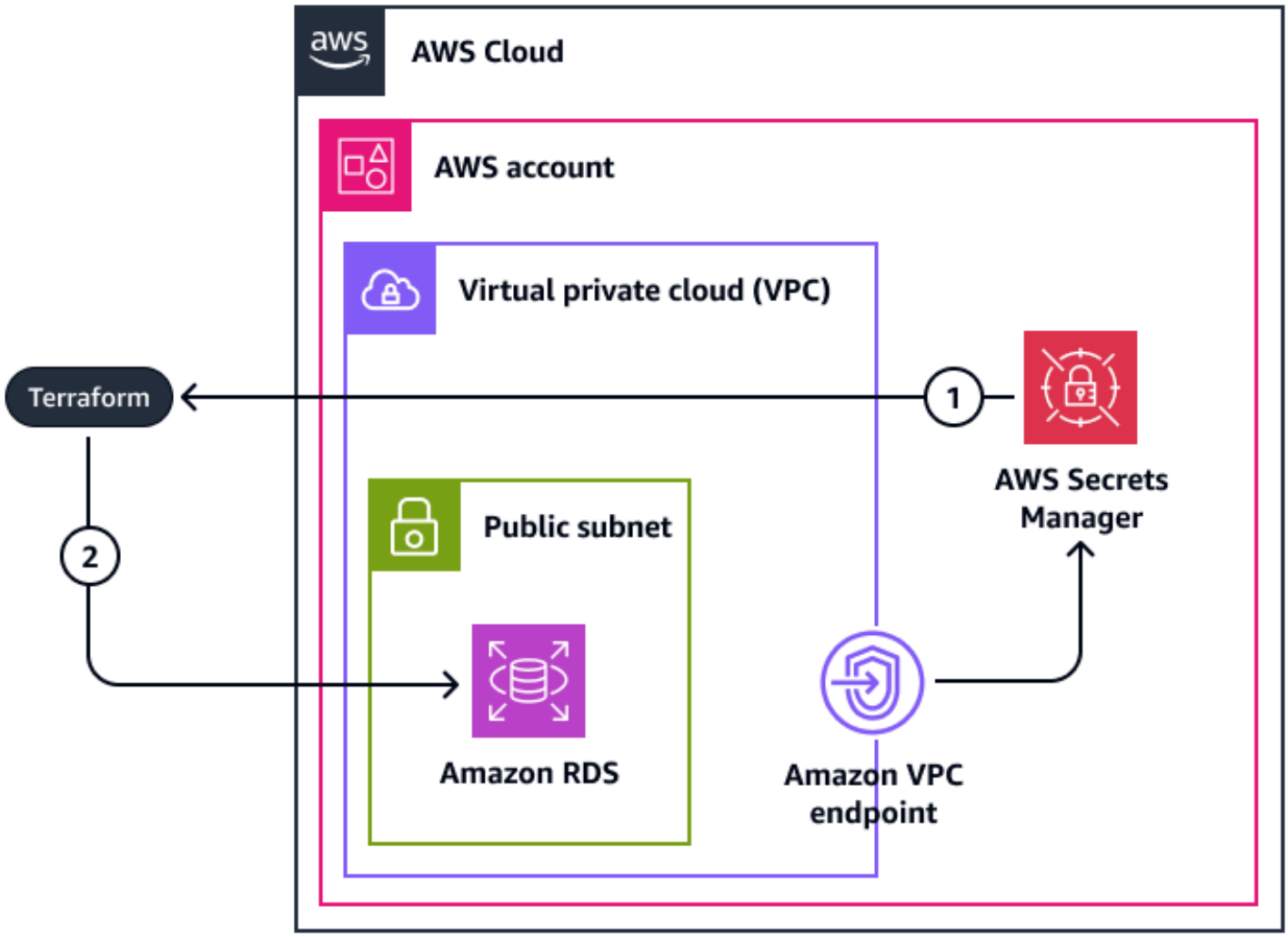
를 사용하여 무작위 암호 생성 AWS Secrets Manager

AWS Well-Architected Framework에서는 보안 암호를 안전하게 저장하고 사용할 것을 권장합니다. AWS Secrets Manager API를 사용하여 임의의 암호를 생성하고 암호 복잡성 요구 사항을 사용자 지정할 수 있습니다. `GetRandomPassword` 작업은 1~4,096자의 암호 문자열 길이를 지원합니다. 자세한 내용은 API 참조의 GetRandomPassword를 참조하세요. AWS Secrets Manager 사용자가 보안 암호를 수동으로 정의할 수 있도록 허용하는 대신이 접근 방식을 사용하여 보안 암호를 생성하는 것이 좋습니다.

다음 코드 샘플은 20자 길이이고 숫자, 구두점 문자 제외 및 공백 제외를 포함하는 무작위 암호를 생성하는 방법을 보여줍니다. 조직의 암호 보안 요구 사항을 충족하도록이 코드 예제를 수정할 수 있습니다.

```
data "aws_secretsmanager_random_password" "test" {
password_length = 20
exclude_numbers = false
exclude_punctuation = true
include_space = false
}
```

IaC를 배포할 때 무작위 보안 암호 생성을 사용하면 0시간이라고 하는 민감한 데이터를 처음부터 보호할 수 있습니다. 배포 단계부터 민감한 데이터는 아무에게도 알려지지 않습니다.



1. 를 통해 AWS Secrets Manager 를 Terraform사용하여 무작위 암호 암호를 생성합니다.
2. Terraform는에 저장된이 무작위 암호 암호를 AWS Secrets Manager사용하여 데이터베이스에 액세스합니다.

> **Important**
>
> 를 데이터 소스Terraform로 사용하면 보안 암호가 상태 파일에 저장되지 않습니다. 그러나 데이터베이스 또는 서비스에서 해당 보안 암호를 사용하면 상태 파일에 저장됩니다. 보안 암호를 즉시 교체하거나 상태 파일에 액세스할 수 있는 매우 제한적인 권한을 생성하는 것이 좋습니다. 자세한 내용은 이 안내서의 Terraform 상태 파일에서 민감한 데이터 보호 섹션을 참조하세요.

AWS Lambda 를 사용하여 보안 암호 교체

AWS Well-Architected Framework에서는 보안 암호를 안전하게 저장하고 사용할 것을 권장합니다. 이 모범 사례에서는 자격 증명 교체를 정기적으로 자동화할 것을 권장합니다. 교체는 공격자가 자격 증명에 액세스하는 것을 더욱 어렵게 만들기 위해 보안 암호를 주기적으로 업데이트하는 프로세스입니다. 또한 많은 규정 준수 프레임워크 및 규정에 따라 보안 암호를 교체해야 합니다.

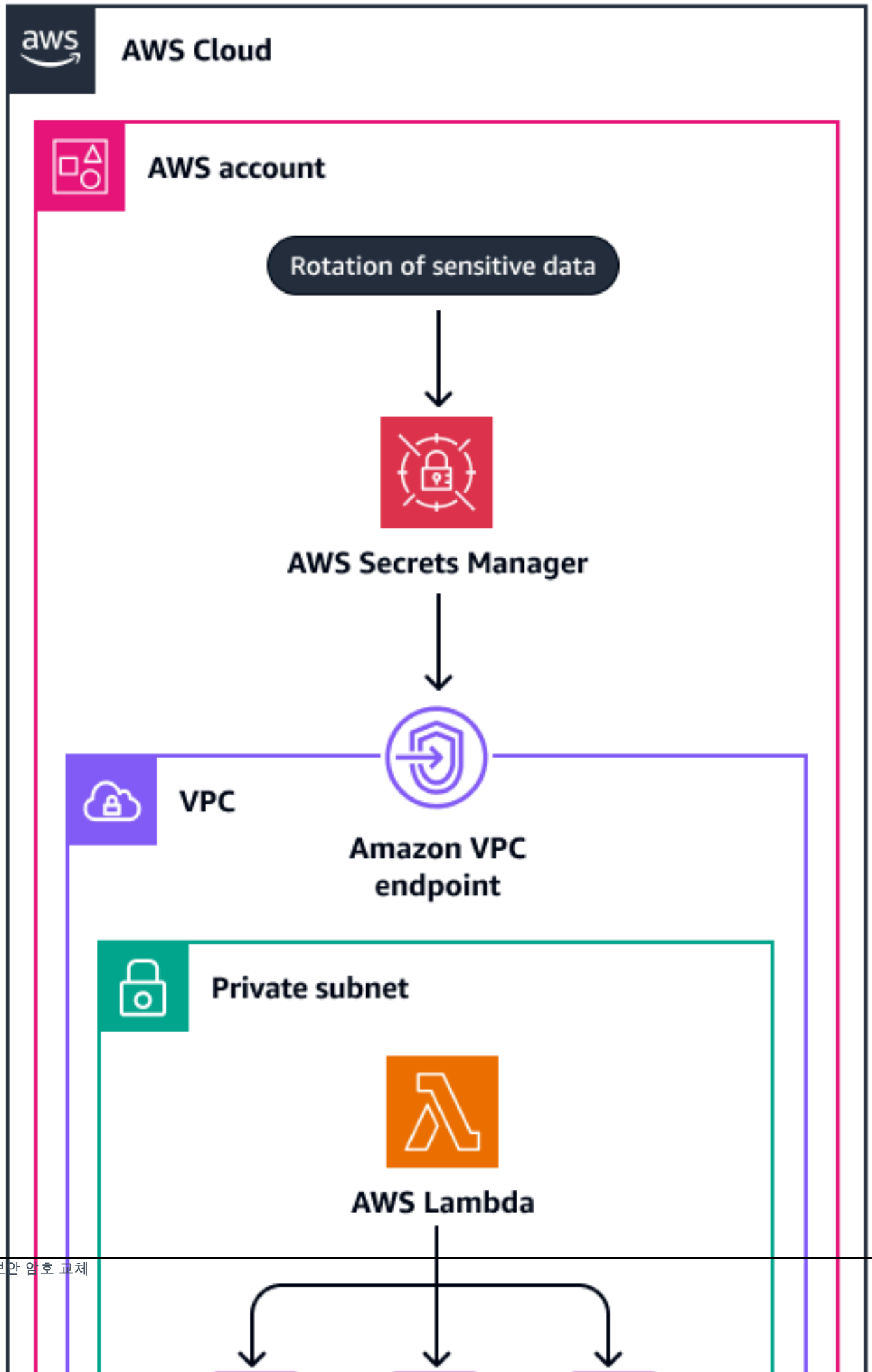
Terraform IaC의 경우 AWS Secrets Manager 및를 사용하여 자동 교체AWS Lambda를 설정할 수 있습니다. Secrets Manager에서 보안 암호에 대한 자동 교체를 설정할 수 있습니다. Secrets Manager가 보안 암호를 교체하면 보안 암호 및 데이터베이스 또는 서비스 모두에서 자격 증명이 업데이트됩니다.

데이터베이스의 경우 Secrets Manager에서 기본 자격 증명을 관리하고 정기적으로 보안 암호를 교체하는 것이 좋습니다. Secrets Manager는 여러 유형의 데이터베이스 자격 증명에 대해 Lambda용 교체 함수 템플릿을 제공합니다. 자세한 내용은 Secrets Manager 설명서의 AWS Secrets Manager 교체 함수 템플릿과 GitHub의 코드 샘플을 참조하세요. 다음은 보안 암호 또는 민감한 데이터를 교체하는 데 사용할 수 있는 Terraform IaC의 예입니다.

```
resource "aws_secretsmanager_secret_rotation" "createrotation" {
  count               = var.needrotation == true ? 1 : 0
  secret_id           = aws_secretsmanager_secret.initiatesecret.id
  rotation_lambda_arn = aws_lambda_function.rotationlambda.arn

  rotation_rules {
    automatically_after_days = 1
  }
}
```

다음 아키텍처 다이어그램은 Secrets Manager, Amazon VPC 엔드포인트 및 Lambda 함수를 사용하여에서 민감한 데이터를 교체하는 방법을 보여줍니다 AWS 계정.


AWS Cloud
AWS account
Rotation of sensitive data
AWS Secrets Manager
VPC
Amazon VPC endpoint
Private subnet
AWS Lambda

보안 암호에 대한 액세스 제한

보안이 최우선 순위입니다 AWS. AWS Well-Architected Framework에서 보안 암호 저장 및 사용 모범 사례에서는 보안 암호에 대한 액세스를 제한하고, 보안 암호에 대한 액세스를 로깅 및 감사하고, 사용자에게 액세스 권한을 부여하는 대신 API를 통해 액세스할 것을 권장합니다. 이는이 가이드에서 앞서 설명한 를 사용하여 무작위 암호를 생성하는 AWS Secrets Manager 것과 관련이 있습니다.

과거에는 데이터베이스, 타사 APIs, 토큰 및 기타 보안 암호에 인증하는 데 사용되는 자격 증명이 소스 코드 또는 환경 파일에 포함되었을 수 있습니다. 많은 조직에서 데이터베이스 관리자는 여전히 모든 데이터베이스 사용자 자격 증명에 액세스할 수 있습니다. 하지만 그렇지 않으면 더 안전합니다. 다음을 수행하는 것이 좋습니다.

- AWS 서비스 역할 또는 사용자의 IAM 역할에 연결된 AWS Identity and Access Management (IAM) 정책을 통해 Secrets Manager에 대한 액세스를 제한합니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-user.html
- Secrets Manager에 대한 API 호출을 통해서만 Secrets Manager의 보안 암호에 대한 프로그래밍 방식 액세스를 확장합니다.
- 하드 코딩된 보안 암호를 사용하지 마십시오.
- 변수에 자격 증명 값을 전달하지 마십시오.
- 보안 암호가 인간 사용자가 아닌 애플리케이션 또는 인프라에서 직접 사용되는지 확인합니다.

다음 코드 샘플은 데이터베이스를 생성할 때 민감한 데이터를 보호하는 데 사용할 수 있는 접근 방식을 보여줍니다. 데이터베이스를 생성할 때 사용자 입력을 암호로 요구한 다음 암호를 교체하거나 처음부터 임의의 암호를 사용할 수 있습니다. 처음부터 임의의 암호를 생성하면 자격 증명을 보다 안전하게 처리하고 관리하는 데 도움이 될 수 있습니다.

다음 코드 조각은 데이터베이스 리소스를 생성하고 로컬 변수에서 암호를 `password = local.passwddata` 가져옵니다.

```
resource "aws_db_instance" "default" {
  count                  = var.createdb == true ? 1 : 0
  allocated_storage      = 20
  identifier             = var.identifier
  db_name                = var.databasename
  engine                 = "mysql"
  engine_version         = "8.0.32"
  instance_class         = "db.t3.micro"
  username               = "admin"
```

```
  password               = local.passwddata
  skip_final_snapshot    = true
  publicly_accessible    = false
  vpc_security_group_ids = [aws_security_group.db.id]
  db_subnet_group_name   = aws_db_subnet_group.db.0.name
}
```

다음 코드 조각은 임의의 암호를 생성하거나 사용자가 제공한 암호를 수락합니다.는이 암호를 로컬 변수로 Terraform 저장하므로 나중에 코드에서이 암호 변수를 사용할 수 있습니다.

```
locals {
  passwddata = var.passwddata != null ? var.passwddata :
 data.aws_secretsmanager_random_password.test.random_secrets
}
```

보안 암호에 계층적 이름 지정 규칙 사용

암호를 수집하고 저장할 때는 정의된 이름 지정 규칙을 AWS Secrets Manager사용합니다. 예를 들어 명명 규칙은 일 수 있습니다`<client name>/<dev or prod>/<project>/<version>`. 조직에 적합한 이름 지정 규칙을 선택합니다. 대부분의 일반 정보에서 가장 구체적인 정보로 진행되는 계층 구조를 사용합니다.

이렇게 하면 특히 AWS 환경 전체에서 중앙 집중식 계정을 여러 보안 암호에 사용하는 경우 보안 암호를 관리하는 데 도움이 됩니다. 계층적 이름 지정 규칙은 다음과 같은 방법으로 도움이 될 수 있습니다.

- 다른 보안 암호에 대한 액세스를 분리하는 데 도움이 됩니다. 예를 들어 어떤 프로젝트 또는 환경에서 어떤 보안 암호에 액세스할 수 있는지 정의할 수 있습니다.
- 이를 통해 보안 암호의 이름을 기반으로 보안 암호에 대한 세분화된 액세스 제어를 설정할 수 있습니다.

다음 코드 샘플은 보안 암호를 대규모로 관리하는 데 도움이 되는 명명 규칙의 예를 보여줍니다.

```
resource "aws_secretsmanager_secret" "initiatesecret" {
  name                    = "org-name/dev-env/project-name"
  kms_key_id              = var.kmskeyarn
  recovery_window_in_days = var.recoverywindow
}
```

사용 시 민감한 데이터 관리 Terraform

민감한 데이터는 잘 설계되어야 합니다. 애플리케이션과 인프라가 확장됨에 따라 민감한 데이터를 신중하게 추적하고 처리하는 것이 점점 더 중요해지고 있습니다. Terraform IaC를 배포할 AWS 계정 때 다음 접근 방식을 사용하여에서 민감한 데이터를 보호할 수 있습니다.

- Terraform 상태 파일에서 민감한 데이터 보호 - 민감한 데이터가 처음 수집되는 순간부터 보호할 수 있습니다 AWS Secrets Manager. 예를 들어 보안 암호를 즉시 교체하여 보안을 유지할 수 있습니다.
- Amazon EKS의 보안 암호 액세스 및 관리 - Secrets Manager에서 Amazon Elastic Kubernetes Service(Amazon EKS)의 모든 보안 암호를 관리합니다.
- VPC 엔드포인트를 사용하여 알려진 네트워크에서 민감한 데이터 유지 - 민감한 데이터의 트래픽은 프라이빗 네트워크를 벗어나서는 안 됩니다. 이를 통해 공격 및 데이터 유출을 방지할 수 있습니다.

Terraform 상태 파일에서 민감한 데이터 보호

이 섹션에서는 라는 Terraform 상태 파일의 민감한 데이터를 처리하기 위한 보안 암호 및 포인터의 난독화에 대해 설명합니다`tfstate`. 일반적으로 배포에 대한 데이터가 포함된 일반 텍스트 파일이며 Terraform 배포된 인프라에 대한 민감한 데이터와 민감하지 않은 데이터가 모두 포함됩니다. 민감한 데이터는 Terraform 상태 파일의 일반 텍스트로 표시됩니다. 민감한 데이터를 보호하려면 다음을 수행합니다.

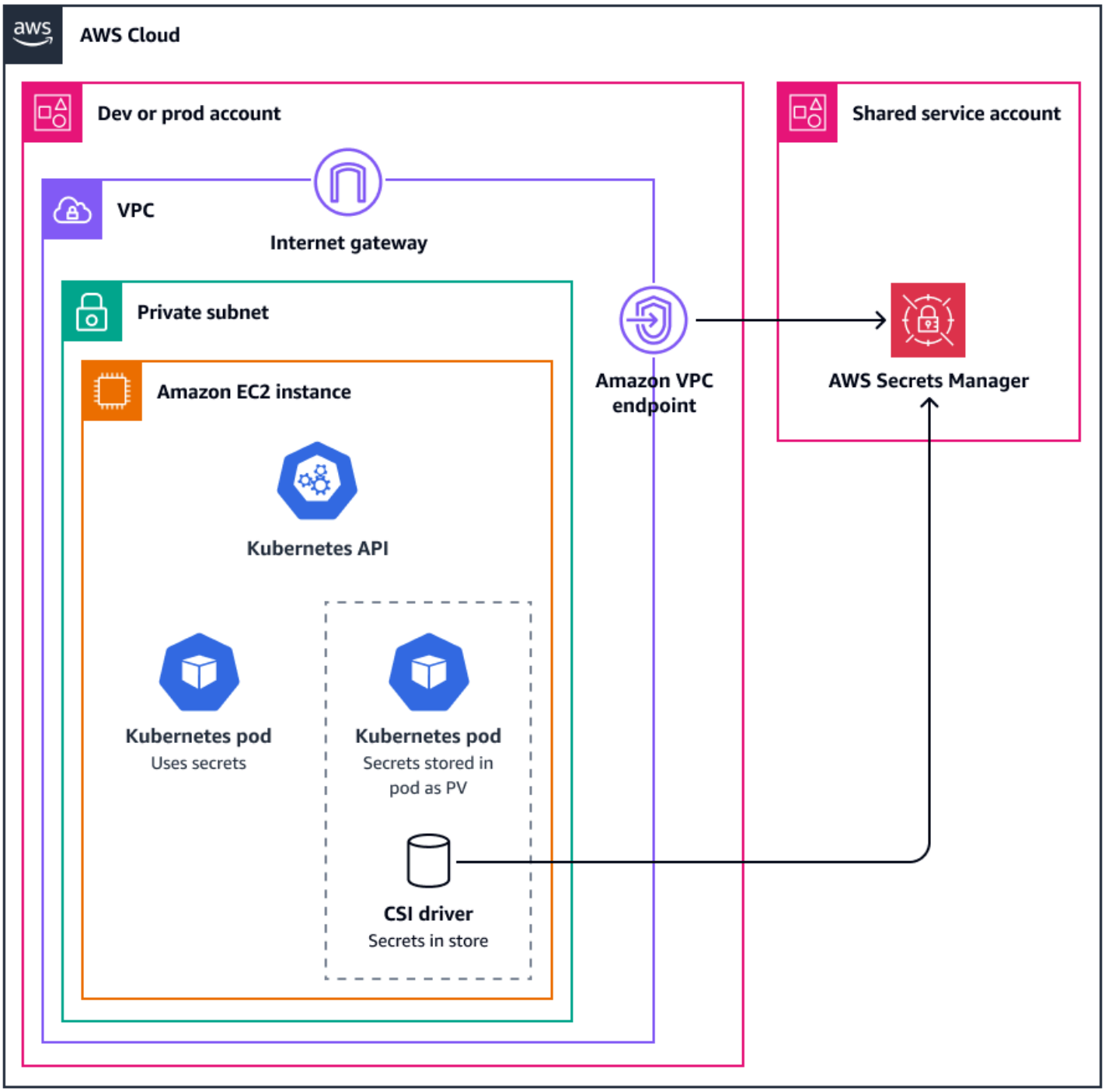
- 보안 암호를 수집할 때 보안 암호를 즉시 교체하도록 선택합니다. 자세한 내용은 Secrets Manager 설명서의 즉시 AWS Secrets Manager 보안 암호 교체를 참조하세요.
- Secrets Manager를 운영하는 중앙 AWS 계정 집중식에 Terraform 상태 파일을 저장합니다. Amazon Simple Storage Service(Amazon S3) 버킷에 파일을 저장하고 이에 대한 액세스를 제한하는 정책을 구성합니다. 자세한 내용은 Amazon S3 설명서의 버킷 정책 및 사용자 정책을 참조하세요.
- 손상을 방지하기 위해 Terraform 상태를 잠글 수 있습니다. 상태 잠금 및 상태 파일 보호에 대한 자세한 내용은 Terraform 설명서의 Amazon S3 백엔드를 참조하세요.

Amazon EKS의 보안 암호 액세스 및 관리

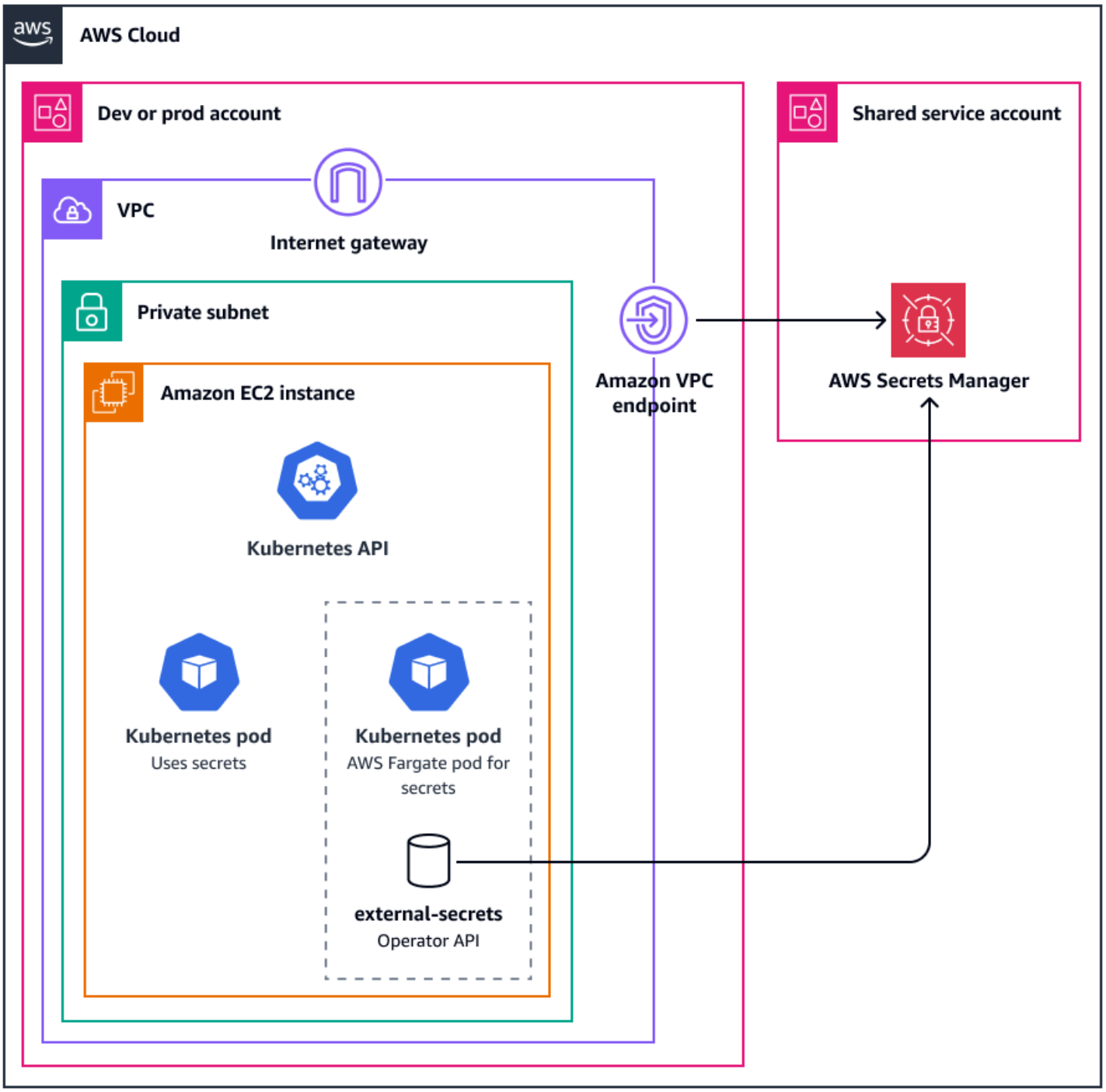
Amazon Elastic Kubernetes Service(Amazon EKS)를 사용하면 자체 Kubernetes 컨트롤 플레인 또는 노드를 설치하거나 유지 관리할 필요 AWS 없이에서 Kubernetes를 실행할 수 있습니다. Amazon EKS는 Base64(Wikipedia) 인코딩을 사용하여 민감한 데이터를 보호합니다. 인코딩은 시스템 간 전송 중

데이터 수정을 방지하도록 설계되었으며 암호화는 데이터에 대한 무단 액세스를 방지하도록 설계되었습니다. Base64 인코딩은 무단 액세스로부터 데이터를 보호하는 데 충분하지 않습니다. AWS Secrets Manager 를 사용하여 Amazon EKS에서 민감한 데이터를 보호합니다.

다음 이미지는 Kubernetes 작업자 노드 역할을 하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 배포된 Amazon EKS를 보여줍니다. 컨테이너 스토리지 인터페이스(CSI) 드라이버를 사용하여 Secrets Manager에서 보안 암호를 검색할 수 있습니다. 자세한 내용은 AWS 보안 블로그의 Kubernetes AWS Secrets Store CSI 드라이버와 함께 보안 암호 및 구성 공급자를 사용하는 방법을 참조하세요.



다음 이미지는에 배포된 Amazon EKS를 보여줍니다AWS Fargate. 오픈 소스 외부 보안 암호 운영자 API(GitHub)를 사용하여 Secrets Manager에서 보안 암호를 검색할 수 있습니다. 자세한 내용은 AWS 컨테이너 블로그의 외부 AWS 보안 암호 연산자를 사용하여 EKS Fargate의 보안 암호 스토어 활용을 참조하세요.



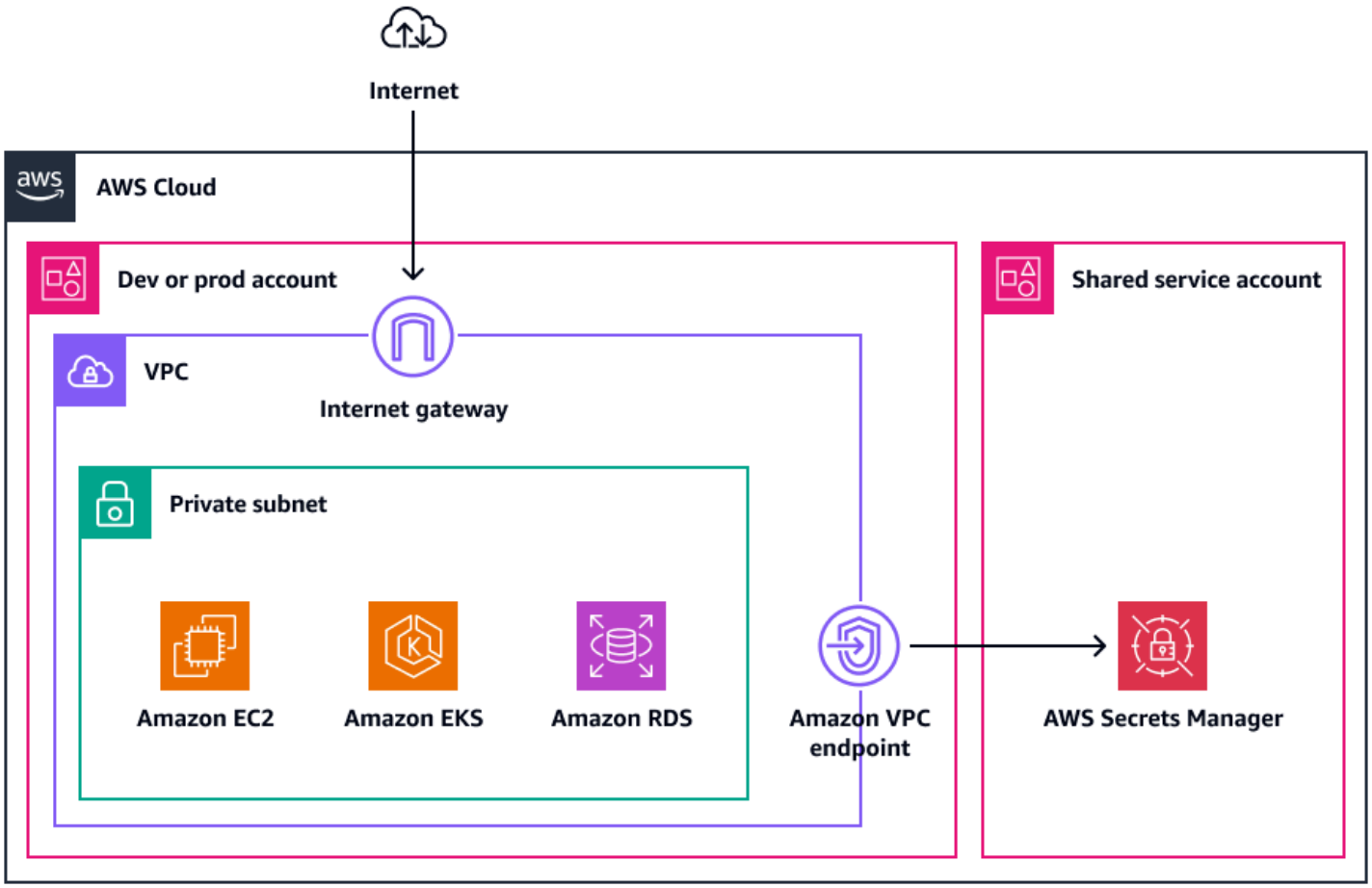
VPC 엔드포인트를 사용하여 알려진 네트워크에서 민감한 데이터 유지

보안 암호는 인터넷을 통해 액세스할 수 없어야 합니다.는 알려진 및 프라이빗 네트워크 경로를 통해 트래픽을 라우팅할 때 개인 정보 보호를 유지하기 위한 옵션을 AWS 제공합니다.

AWS Secrets Manager 와 온프레미스 클라이언트 및 애플리케이션 간의 트래픽을 구성할 때 다음 방법 중 하나를 사용할 수 있습니다.

- AWS Site-to-Site VPN VPN 연결
- AWS Direct Connect 연결

Secrets Manager와 API 클라이언트 간의 트래픽을 동일한 로 보호하려면를 AWS 리전사용하여 인터페이스 VPC 엔드포인트AWS PrivateLink를 생성합니다. 이 옵션을 사용하면 보안 암호에 대한 모든 트래픽을 프라이빗 네트워크 내에 유지할 수 있습니다. 자세한 내용을 알아보려면 AWS Secrets Manager VPC 엔드포인트 사용을 참조하세요.



리소스

AWS 설명서

- 비밀 AWS Secrets Manager 로테이트
- AWS Secrets Manager 보안 암호를 즉시 교체
- AWS Secrets Manager 교체 함수 템플릿
- 의 데이터 보호 AWS Secrets Manager
- AWS Secrets Manager VPC 엔드포인트 사용
- S3 버킷 정책 및 사용자 정책
- 보안 암호를 안전하게 저장하고 사용(AWS Well-Architected Framework)
- 주기적으로 자격 증명 감사 및 교체(AWS Well-Architected Framework)

AWS 블로그 게시물

- 리소스 기반 정책을 연결하여에서 보안 암호 AWS 계정 에 액세스하는 방법
- Kubernetes AWS Secrets Store CSI 드라이버와 함께 보안 암호 및 구성 공급자를 사용하는 방법
- 외부 AWS 보안 암호 연산자를 사용하여 EKS Fargate의 보안 암호 스토어 활용

기타 리소스

- Amazon S3 백엔드(Terraform 설명서)

기여자

다음 개인이이 가이드에 기여했습니다.

작성

- Chintamani Aphale, 수석 컨설턴트, AWS
- Aarti Rajput, 수석 컨설턴트, AWS
- T.V.R.L. Phani Kumar Dadi, 컨설턴트, AWS
- Pradip Kumar Pandey, 수석 컨설턴트, AWS
- Mayuri Shinde, DevOps Consultant, AWS
- Pratap Kumar Nanda, 수석 컨설턴트, AWS

검토

- Janardhan Malyala, 수석 보안 컨설턴트, AWS
- Anand Krishna Varanasi, 수석 컨설턴트, AWS
- Sandeep Gawande, 수석 수석 컨설턴트, AWS
- Mike Zazon, 보안 주체 클라우드 인프라 아키텍트 AWS

기술 작성

- GxP AbouHarb, Senior Technical Writer, AWS

문서 기록

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

변경 사항	설명	날짜
상태 파일 잠금	Terraform 상태 파일 섹션에서 민감한 데이터 보호 섹션을 업데이트했습니다. Amazon DynamoDB 테이블을 사용하여 상태 파일을 잠근 상태로 유지하는 것은 더 이상 지원되지 않습니다.	2025년 5월 20일
최초 게시	—	2024년 1월 5일

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