조직의 코드형 인프라 도구 선택

AWS 권장 가이드

AWS 권장 가이드: 조직의 코드형 인프라 도구 선택

Table of Contents

조직의 코드형 인프라 도구 선택

Amazon Web Services(기여자)

2026년 2월(문서 기록)

코드형 인프라(IaC)는 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. 버전 관리, 지속적 통합 및 지속적 배포와 같은 애자일 및 DevOps 관행의 주요 구성 요소입니다.

코드형 인프라(IaC) 도구를 선택하는 것은 조직의 전략적 결정으로 간주됩니다. 이 결정은 회사의 인프라, 애플리케이션 및 서비스를 구축하는 모든 팀에 영향을 미칩니다. 각 도구에는 장단점이 있으므로 one-size-fits-all 모델이 없습니다.

과거에는 인프라 관리 및 프로비저닝이 오류가 발생한 수동 프로세스였습니다. IaC는 코드를 통해 이러한 작업을 간소화하며 인프라를 배포하기 위한 신뢰할 수 있는 솔루션이 되었습니다. IaC 도구는 개발자가 프로그래밍 언어를 사용하여 인프라를 정의하고 배포할 수 있도록 지원합니다. 이렇게 하면 비즈니스 민첩성이 향상될 뿐만 아니라 성장과 혁신 속도가 가속화됩니다. 또한 IaC를 사용하면 조직이 배포 전에 코드를 스캔하여 인프라가 안정적이고 안전한지 확인할 수 있으므로 IaC는 보안을 크게 개선합니다. 궁극적으로 올바른 IaC 도구는 기술적 결정일 뿐만 아니라 비즈니스의 전반적인 성공에 직접적인 영향을 미치는 전략적 결정입니다.

이 가이드에서는 리소스를 프로비저닝 AWS 하는 데 사용할 수 있는 다섯 가지 IaC 도구, 즉 AWS CloudFormation, AWS Serverless Application Model (AWS SAM), AWS Cloud Development Kit (AWS CDK), HashiCorp Terraform 및 Pulumi를 살펴봅니다. 이러한 도구를 비교하고 팀, 조직 및 클라우드 인재의 요구 사항을 충족하는 도구를 선택하는 프로세스를 안내합니다. 핵심은 선택한 IaC 도구를 조직의 목표 및 개발자의 기술 역량에 맞추는 것입니다. 예를 들어 팀이 JavaScript에 능숙하다면 개발 워크플로를 최적화하기 때문에 TypeScript AWS CDK 를 기본 IaC 도구로 선택할 수 있습니다.

IaC 구현의 이점

조직에 적합한 IaC 도구를 선택하면 다음과 같은 이점을 얻을 수 있습니다.

- 속도 — IaC의 목표 중 하나는 수동 프로세스를 제거하여 작업 속도를 높이는 것입니다. 코드 기반 접근 방식을 사용하면 더 짧은 시간에 더 많은 작업을 더 쉽게 수행할 수 있습니다. 각 IT 환경을 수동으로 설정하는 것은 비용이 많이 들고 하드웨어와 소프트웨어를 설정하는 전담 엔지니어와 설계자가 필요합니다. 속도를 높이면 조직은 변화하는 고객 요구와 시장 상황에 더 빠르게 적응할 수 있습니다. 또한 IaC 코드를 한 번 작성하여 수백 개의 환경에 동일한 코드를 배포할 수 있어 수동으로 생성하는 데 걸리는 시간보다 훨씬 짧은 시간이 소요됩니다.
- 확장성 — IaC를 사용하면 기존 인프라에 리소스를 더 쉽게 추가할 수 있습니다. 업그레이드가 빠르게 제공되므로 사용량이 가장 많은 시간대에 빠르게 확장할 수 있습니다. 예를 들어 온라인 서비스를 운영하는 조직은 블랙 프라이데이와 같이 수요가 많은 시기에 사용자 수요에 맞춰 규모를 확장할 수 있습니다.
- 보안 강화 — IaC는 조직의 보안 및 규정 준수를 지원하는 데 탁월합니다. 조직은 IaC에 대해 자동화된 테스트를 실행하여 기본 보안 정책 및 구성을 설정하고 파이프라인을 통해 이를 적용할 수 있습니다. IaC가 규정 준수 규칙을 위반하면 파이프라인이 실패하고 개발자에게 자동으로 피드백을 제공합니다. 이렇게 하면 안전하지 않은 인프라가 생성되는 것을 방지할 수 있습니다.
- 일관성 — 일관성은 IaC의 또 다른 중요한 이점입니다. 여러 엔지니어가 구성을 수동으로 배포하는 경우 불일치는 불가피합니다. 시간이 지나면 동일한 환경을 추적하고 재현하기가 어려워집니다. 이러한 불일치로 인해 개발, QA 및 프로덕션 환경 간에 중대한 차이가 발생하는 경우가 많습니다.
- 효율성 — IaC는 개발 라이프사이클 전반에 걸쳐 효율성과 생산성을 개선합니다. 프로그래머는 샌드박스 환경을 만들어 따로 개발합니다. 운영팀은 보안 테스트를 위한 인프라를 신속하게 프로비저닝할 수 있습니다. QA 엔지니어는 테스트 중에 프로덕션 환경의 정확한 사본을 보유합니다. 배포할 준비가 되면 개발자는 인프라와 코드를 한 번에 프로덕션 환경에 적용할 수 있습니다. 또한 더 높은 수준의 협업과 팀워크를 가능하게 할 수 있습니다. 팀에서 응용 프로그램의 보안 패턴을 만든 다음 해당 템플릿이나 구성을 다른 팀과 공유하여 중복 작업을 줄일 수 있습니다.
- 비용 절감 — IaC는 소프트웨어 개발 비용을 줄여줍니다. 환경을 수동으로 설정하는 데 리소스를 소비할 필요가 없습니다. 현재 사용 중인 리소스에 대해서만 비용을 지불하므로 불필요한 오버헤드가 발생하지 않습니다.
- 안정성 향상 — 인프라가 크면 리소스를 잘못 구성하거나 서비스를 잘못된 순서로 프로비저닝하기 쉽습니다. IaC를 사용하면 리소스가 항상 선언된 대로 정확하게 프로비저닝되고 구성됩니다. 리소스를 수동으로 생성하면 오류가 발생하기 쉬우므로 IaC를 사용하면 인프라가 의도한 대로 생성될 것이라는 확신을 가질 수 있습니다.

- 구성 편차 탐지 — 환경을 프로비저닝한 구성이 더 이상 실제 환경과 일치하지 않을 때 구성 드리프트가 발생합니다. 많은 IaC 도구를 사용하여 드리프트를 감지하고 문제를 해결할 수 있습니다. 예를 들어, 누군가 리소스를 수동으로 잘못 수정한 경우 IaC 도구를 사용하여 변경된 내용을 감지하고 의도한 상태로 복원할 수 있습니다.
- 실험 — IaC를 사용하면 새 인프라를 훨씬 빠르고 쉽게 프로비저닝할 수 있으므로 많은 시간과 리소스를 투자하지 않고도 실험적 변경 사항을 적용하고 테스트할 수 있습니다. 결과가 마음에 들면 프로덕션을 위해 새 인프라를 빠르게 확장할 수 있습니다.

다음을 위한 IaC 도구 검토 AWS 클라우드

이 가이드에서는 리소스를 프로비저닝하는 데 사용할 수 있는 다섯 가지 IaC 도구를 살펴봅니다. AWS 이러한 도구를 비교하고 팀, 조직 및 클라우드 인재의 요구 사항에 맞는 도구를 선택하는 과정을 안내합니다. 핵심은 선택한 IaC 도구를 조직의 목표 및 개발자의 기술 수준에 맞추는 것입니다.

이 섹션에서는 각 도구의 작동 방식과 장단점에 대해 자세히 알아봅니다. 예를 들어 일부 도구는 에서만 사용할 수 AWS 클라우드있으며 멀티 클라우드 또는 하이브리드 클라우드 배포에는 적합하지 않습니다. 특수 프로그래밍 언어에 대한 지식이 필요한 곳도 있고 일반 프로그래밍 언어를 지원하는 곳도 있습니다. 각 도구의 장단점을 평가하고 조직에 가장 적합한 것이 무엇인지 생각해 보세요.

이 섹션에서는 다음과 같은 IaC 도구에 대해 설명합니다.

- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)
- AWS Cloud Development Kit (AWS CDK)
- HashiCorp 테라폼
- 풀루미

IaC AWS CloudFormation 도구로 사용

AWS CloudFormation템플릿 파일을 사용하여 리소스 프로비저닝을 자동화하는 도구입니다. AWS 서비스 AWS 배포하려는 모든 AWS 리소스를 설명하는 템플릿을 만들고 해당 리소스를 자동으로 CloudFormation 프로비저닝 및 구성합니다.

CloudFormation 템플릿은 JSON 또는 YAML을 사용하여 작성됩니다. CloudFormation스택은 템플릿에 정의된 리소스의 구현입니다. 를 통해 AWS Management Console, CloudFormation SDK를 통한 프로그래밍 방식 또는 () 를 통해 CloudFormation 스택을 관리할 수 있습니다. AWS Command Line Interface AWS CLI CloudFormation 작동 방식에 대한 자세한 내용은 AWS CloudFormation 설명서의 개념 및 AWS CloudFormation 작동 방식을 참조하십시오. CloudFormation

사용의 이점 CloudFormation:

- CloudFormation 변경 세트를 사용하면 해당 변경 내용을 배포하기 전에 실행 중인 스택의 변경 내용을 미리 볼 수 있습니다. 변경 세트는 기존 스택에서 실행 중인 리소스에 제안된 변경 사항을 요약합니다. 이를 통해 배포 전에 충돌이나 의도하지 않은 결과를 식별할 수 있습니다. 예를 들어 Amazon

관계형 데이터베이스 서비스 (Amazon RDS) 데이터베이스 CloudFormation 인스턴스의 이름을 변경하면 에서 새 데이터베이스를 생성하고 이전 데이터베이스를 삭제합니다. 이미 백업하지 않은 이상 이전 데이터베이스의 데이터는 손실됩니다. 변경 세트를 생성하면 변경으로 인해 데이터베이스가 교체되는 것을 확인할 수 있으며 스택을 업데이트하기 전에 그에 따라 계획을 세울 수 있습니다.

- 변경 세트를 배포하는 동안 오류가 발생하는 경우 마지막으로 알려진 작업 상태로 자동 CloudFormation 롤백됩니다.
- CloudFormation 스택 세트를 사용하여 여러 랜드에 리소스를 배포할 수 AWS 계정 있습니다 AWS 리전.
- AWS: :*, Alexa: :*, Custom: :* 및 Custom: :*의 네임스페이스에서 리소스 공급자와 CloudFormation 함께 사용하는 경우 추가 요금이 부과되지 않습니다. 이러한 경우에는 수동으로 프로비저닝한 것처럼 프로비저닝한 AWS 리소스에 대한 비용만 지불하면 됩니다.
- CloudFormation 상태를 자동으로 관리합니다. 즉 CloudFormation , CloudFormation 템플릿에 정의된 대로 리소스를 프로비저닝하고 AWS 구성하기 위해 기본 서비스를 호출합니다.
- CloudFormation 구성 편차를 감지하고 해결하는 도구를 제공합니다. 자세한 내용은 설명서의 스택 및 리소스에 대한 관리되지 않는 구성 변경 감지를 참조하십시오. CloudFormation
- 를 CloudFormation 사용하여 사용자 지정 리소스를 만들 수 있습니다. 스택을 생성, 업데이트 또는 삭제할 때마다 CloudFormation 실행되는 사용자 지정 프로비저닝 로직을 템플릿에 작성할 수 있습니다.
- CloudFormation 레지스트리를 통해 타사 애플리케이션 리소스의 모델링, 프로비저닝 및 관리를 지원합니다. CloudFormation
- CloudFormation 기존 리소스를 경영진으로 CloudFormation 가져올 수 있습니다.

사용의 CloudFormation 단점:

- JSON 또는 YAML 구문에 익숙하지 않은 경우 익숙해지는 데 시간이 걸릴 수 있습니다. JSON은 사람이 읽을 수 있도록 설계되지 않았으며 인라인 주석을 작성할 수 없습니다. YAML을 사용하면 댓글을 달 수 있고 읽기도 더 쉽습니다. 하지만 구문은 탭과 공백을 기반으로 하므로 들여쓰기 실수를 하기 쉽습니다.
- CloudFormation 멀티 클라우드 배포는 지원하지 않습니다.
- 재사용 가능한 구문 및 기타 모듈화된 코드를 만들려면 와 같은 상위 수준 구현을 사용해야 합니다. AWS Cloud Development Kit (AWS CDK)

AWS Serverless Application Model (AWS SAM) 를 IaC 도구로 사용

AWS Serverless Application Model (AWS SAM) 는 확장 가능한 툴킷입니다. AWS CloudFormation여기에는 서버리스 애플리케이션을 더 빠르게 만들 수 있도록 설계된 추가 기능이 포함되어 있습니다. AWS SAM 템플릿을 배포하면 정의된 리소스를 생성하기 위해 템플릿이 변환됩니다. CloudFormation AWS SAM AWS SAM 템플릿 사양과AWS SAM CLI (AWS SAM 명령줄 인터페이스) 의 두 부분으로 구성되어 있습니다. AWS SAM 템플릿에서 직접 CloudFormation 구문을 사용할 수도 있지만, AWS SAM 은 특히 서버리스 개발 속도를 높이는 데 초점을 맞춘 고유한 구문을 제공합니다. 이 간단한 구문을 사용하면 Amazon API Gateway와 같은 서버리스 리소스 및 리소스에 대한 IaC를 최적화할 수 있습니다. AWS Lambda AWS Step Functions AWS SAM CLI는 로컬에서 AWS Lambda 함수를 테스트하고, 지속적 통합 및 지속적 전달 (CI/CD) 파이프라인을 생성하고, 명령을 실행하여 서버리스 애플리케이션을 배포하는 데 도움이 되는 기능을 포함하는 개발자 도구입니다.

사용의 이점: AWS SAM

- AWS SAM 와 같은 장점이 있습니다 CloudFormation.
- 에 비해 CloudFormation, 에서 지원하는 Amazon API Gateway와 같은 서버리스 애플리케이션 및 리소스를 만드는 AWS SAM 데 더 쉽게 사용할 수 있습니다. AWS Lambda
- AWS SAM CLI를 사용하면 AWS Lambda 함수를 로컬에서 테스트할 수 있습니다. 디버그 모드에서 Lambda 함수를 로컬로 호출하면 디버거를 해당 함수에 연결할 수 있습니다. 디버거를 사용하면 코드를 한 줄씩 단계별로 실행하고, 다양한 변수의 값을 확인하고, 다른 애플리케이션과 동일한 방식으로 문제를 해결할 수 있습니다.

사용의 단점: AWS SAM

- AWS SAM 와 같은 단점이 있습니다 CloudFormation.
- AWS SAM 외부에서는 사용할 수 없습니다 AWS.

IaC AWS CDK 도구로 사용

친숙한 프로그래밍 언어를 사용하여 클라우드 애플리케이션 리소스를 정의할 수 있는 오픈 소스 소프트웨어 개발 AWS Cloud Development Kit (AWS CDK)프레임워크입니다. AWS CDK 는 JavaScript, Python TypeScript, 자바, C#, Go를 지원합니다. 를 통해 안전하고 반복 가능한 방식으로 리소스를 AWS CDK 프로비저닝합니다. AWS CloudFormation AWS CDK 코드를 합성하면 결과가 템플릿이 됩니다. CloudFormation 는 리소스 정의 프로세스를 단순화하는 높은 수준의 추상화를 AWS CDK 제공합니다. AWS

는 구문의 개념을 AWS CDK 사용합니다. 구문은 Amazon Simple Storage Service (Amazon S3) 버킷과 같은 하나 이상의 CloudFormation 리소스와 해당 구성을 나타내는 애플리케이션 내의 구성 요소입니다. 구문을 구성하고 사용자 지정하여 더 복잡한 인프라를 만들 수 있습니다. 자세한 내용은 AWS CDK 설명서의 구성 수준을 참조하십시오. 는 개발자가 작성한 코드를 기반으로 CloudFormation 템플릿을 AWS CDK 생성합니다. 따라서 CloudFormation 템플릿을 수동으로 만들 필요가 없습니다. 많은 조직에서 다른 소프트웨어 라이브러리와 마찬가지로 커뮤니티 내에서 구성을 사용자 지정, 공유 및 재사용합니다. 구문을 공유하면 개발자가 코드를 더 빠르게 작성하고 기본적으로 모범 사례를 통합할 수 있습니다.

AWS CDK 측면은 조직이 지정된 범위 내의 모든 구문에 표준을 적용하는 데 도움이 될 수 있습니다. 애스펙트는 태그를 추가하는 등의 방법으로 구성을 수정할 수 있습니다. 또는 구문의 상태에 대해 무언가를 확인할 수도 있습니다.

AWS CDK 이를 통해 개발자는 기존 프로그래밍 기술과 지식을 사용하여 클라우드 인프라를 정의할 수 있습니다. 친숙한 프로그래밍 언어를 사용하면 개발자가 전문 지식을 적용하여 AWS 리소스를 설명할 수 있으므로 애플리케이션 개발에서 인프라 프로비저닝으로 쉽게 전환할 수 있습니다. 또한 인프라 구축을 가속화할 AWS CDK 수 있습니다. AWS 이렇게 하면 템플릿을 수동으로 작성하는 것보다 개발 라이프사이클이 빨라집니다. CloudFormation

사용의 AWS CDK이점:

- 잘 알려진 프로그래밍 언어를 AWS CDK 지원합니다.
- 범용 언어를 사용하면 for 루프, 객체, 강력한 형식 및 기타 프로그래밍 기법과 같은 논리적 구조를 사용할 수 있습니다. 이를 통해 개발자는 간결하고 오류 없는 방식으로 인프라를 선언할 수 있습니다. 또한 이 접근 방식을 사용하면 통합 개발 환경 (IDE) 및 관련 도구를 사용하여 대규모 리소스 선언과 관련된 복잡성을 관리할 수 있습니다.
- AWS CDK 구조는 공유할 수 있으며 거버넌스 및 규정 준수 요구 사항을 충족하는 데 도움이 됩니다.
- AWS CDK 구성을 사용하면 개발에 드는 시간과 노력을 줄일 수 있습니다. 자세한 내용은 구성 라이브러리 API 레퍼런스를 참고하세요.
- AWS CDK 를 기반으로 CloudFormation 합니다. 개념에 CloudFormation 익숙하다면 AWS CDK 개념을 더 쉽게 이해할 수 있습니다.
- 단위 테스트 및 스냅샷 테스트를 수행하는 AWS CDK 데 도움이 됩니다.
- 기능이 에서 기본적으로 지원되지 않는 경우 수준 1 AWS CDK 구문과 원시 재정의를 사용할 수 있습니다. 또는 API를 직접 호출하는 CloudFormation 사용자 지정 리소스를 사용할 수도 있습니다.
- CloudFormation 스택을 삭제하여 리소스를 효율적으로 정리할 수 있습니다.

사용의 AWS CDK단점:

- 에는 각각 부트스트랩 환경이 AWS CDK 필요합니다. AWS 계정부트스트래핑은 리소스를 배포하는 모든 환경에서 수행해야 하는 일회성 작업입니다.
- 는 에서 IaC를 배포하는 데만 사용할 AWS CDK 수 있습니다. AWS 클라우드

테라폼을 IaC 도구로 사용 AWS 클라우드

HashiCorp Terraform은 클라우드 인프라를 관리하는 데 도움이 되는 코드형 인프라 (IaC) 도구입니다. Terraform을 사용하면 버전 지정, 재사용 및 공유할 수 있는 구성 파일에 클라우드 및 온프레미스 리소스를 모두 정의할 수 있습니다. 그런 다음 일관된 워크플로를 사용하여 수명 주기 전반에 걸쳐 모든 인프라를 프로비저닝하고 관리할 수 있습니다.

개발자는 Terraform 언어라는 고급 구성 언어를 사용합니다. Terraform 언어의 기본 저수준 구문은 구성 언어 (HCL) 입니다HashiCorp. Terraform 언어는 사람이 쉽게 읽고 쓸 수 있도록 설계되었습니다. Terraform 언어를 사용하여 클라우드 또는 온프레미스 인프라의 원하는 최종 상태를 설명합니다. 그런 다음 Terraform은 해당 최종 상태에 도달하기 위한 계획을 생성하고, 사용자는 계획을 실행하여 인프라를 프로비저닝합니다.

Terraform 사용의 이점:

- 테라폼은 플랫폼에 구애받지 않습니다. 모든 클라우드 서비스 공급자와 함께 사용할 수 있습니다. 기타 여러 클라우드 제공업체 전반에서 인프라를 구성, 테스트 AWS 및 배포할 수 있습니다. 조직에서 여러 클라우드 공급자를 사용하는 경우 Terraform은 클라우드 인프라를 관리하는 일관된 단일 통합 솔루션이 될 수 있습니다. 멀티 클라우드 지원에 대한 자세한 내용은 Terraform 웹 사이트의 멀티 클라우드 프로비저닝을 참조하십시오.
- Terraform은 에이전트가 없습니다. 관리형 인프라에는 소프트웨어를 설치할 필요가 없습니다.
- Terraform 모듈은 코드를 재사용하고 반복하지 마세요 (DRY) 원칙을 고수할 수 있는 강력한 방법입니다. 예를 들어, Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스, Amazon Elastic Block Store (Amazon EBS) 볼륨 및 논리적으로 그룹화된 기타 리소스를 포함하는 애플리케이션에 대한 특정 구성이 있을 수 있습니다. 이 구성 또는 애플리케이션의 사본을 여러 개 생성해야 하는 경우 전체 코드를 여러 번 복사하는 대신 리소스를 Terraform 모듈로 패키징하고 모듈의 여러 인스턴스를 생성할 수 있습니다. 이러한 모듈은 구성을 구성, 캡슐화 및 재사용하는 데 도움이 될 수 있습니다. 또한 일관성을 제공하고 모범 사례를 보장합니다.
- Terraform은 인프라의 드리프트 (Terraform 블로그 게시물) 를 감지하고 관리할 수 있습니다. 예를 들어 Terraform에서 관리하는 리소스가 Terraform 외부에서 수정된 경우 Terraform CLI를 사용하여 드리프트를 감지하고 원하는 상태로 복원할 수 있습니다.

테라폼 사용의 단점:

- 클라우드 공급자와 관련된 새 기능이나 새 리소스에 대한 지원이 제공되지 않을 수 있습니다.
- Terraform은 다음과 같이 상태를 자동으로 관리하지 않습니다. AWS CloudFormation기본적으로 로컬 파일에 저장되지만 Amazon S3 버킷이나 Terraform Enterprise를 통해 원격으로 저장할 수도 있습니다.
- Terraform 상태에는 데이터베이스 암호와 같은 민감한 데이터가 포함될 수 있으며, 이로 인해 보안 문제가 발생할 수 있습니다. 상태 파일을 암호화하고, 원격으로 저장하고, 파일 버전 관리를 활성화하고, 읽기 및 쓰기 작업에 최소한의 권한을 사용하는 것이 가장 좋습니다. 자세한 내용은 AWS Secrets Manager Terraform을 HashiCorp 사용하여 민감한 데이터 보안을 참조하십시오.
- 2023년 8월, Hashicorp는 Mozilla 퍼블릭 라이선스에 따라 더 이상 오픈 소스로 라이선스를 받지 않겠다고 발표했습니다. 대신 이제는 비즈니스 소스 라이선스에 따라 라이선스가 부여되었습니다.

Pulumi를의 IaC 도구로 사용 AWS 클라우드

Pulumi는 오픈 소스 코드형 인프라 도구입니다. TypeScript, JavaScript, Python, Go, .NET, Java 및 YAML과 같은 기존의 일반적인 프로그래밍 언어를 지원합니다. 또한 기본 에코시스템을 사용하여 Pulumi SDK를 통해 클라우드 리소스와 상호 작용합니다. 다운로드 가능한 CLI, 런타임, 라이브러리 및 호스팅 서비스가 함께 작동하여 클라우드 인프라를 프로비저닝, 업데이트 및 관리합니다.

Pulumi SDK를 사용하여 모든 클라우드에서 컨테이너, 서버리스 함수, 호스팅 서비스 및 인프라를 사용하는 클라우드 소프트웨어를 생성하고 배포할 수 있습니다.

선택적으로 Pulumi를 Pulumi 클라우드와 페어링할 수 있습니다. Pulumi Cloud는 상태 및 암호를 저장하는 관리형 서비스이며 Pulumi 인프라의 배포를 관리합니다.

Pulumi 사용의 이점:

- Pulumi는 50개 이상의 클라우드 및 서비스형 소프트웨어(SaaS) 공급자의 인프라를 제공합니다.
- 클라우드 복잡성을 줄이도록 설계된 완전하고 일관된 인터페이스를 제공합니다.

Pulumi 사용 시 단점:

- Pulumi에는 지원을 제공하는 활성 커뮤니티가 있지만 Terraform 커뮤니티보다 작습니다.
- Pulumi에는 더 가파른 학습 곡선이 있습니다.

IaC 도구 선택

그렇다면 어떤 도구를 선택해야 할까요?

도구 옵션이 너무 다양하고 비즈니스 요구 사항도 다양하기 때문에 one-size-fits-all 접근 방법이 없습니다. 이 가이드에서 설명하는 각 도구의 장단점 외에도 비즈니스 요구 사항 및 운영 모델에 대한 다음 권장 사항을 고려해 보십시오.

- 종속성이나 종속 항목이 최소화된 서버리스 AWS 솔루션을 관리하거나 배포하는 경우 AWS Serverless Application Model (AWS SAM) 이 좋은 옵션일 수 있습니다. 와 같은 기능을 모두 갖추고 있습니다. AWS CloudFormation또한 서버리스 애플리케이션의 테스트 및 배포를 간소화합니다. AWS 클라우드
- 인프라를 완전히 관리하는 AWS경우 AWS CloudFormation 좋은 AWS Cloud Development Kit (AWS CDK) 옵션입니다. out-of-the-box 상태 관리 기능을 제공하며 새 기능이나 AWS 리소스를 기본적으로 사용할 수도 있습니다.
- 멀티 클라우드 또는 하이브리드 클라우드 인프라 관리를 위한 멀티 프로바이더 유틸리티를 원한다면 플랫폼에 구애받지 않으므로 Terraform이 좋은 선택일 수 있습니다. Terraform을 사용하면 다양한 플러그인을 사용할 수 있으며 엔터프라이즈 지원 옵션을 갖춘 대규모 커뮤니티가 있습니다.
- 모범 사례가 포함된 하향식 배포가 있고 일반적인 프로그래밍 언어를 사용하여 재사용 가능한 모듈을 생성, 게시 및 배포하는 오케스트레이션이 있다면 좋은 옵션이 AWS CDK 될 수 있습니다.
- 조직에서 높은 수준의 위험을 감수할 수 있고 멀티 클라우드 또는 하이브리드 클라우드 환경을 지원해야 하는 경우 Pulumi를 사용해 보세요.

다음 단계

이 가이드에서는 AWS 리소스와 인프라를 구축, 배포 및 관리하는 데 사용할 수 있는 여러 IaC (코드형 인프라) 도구의 장단점에 대해 설명했습니다. 선택한 도구에 따라 다음 리소스를 방문하여 시작하는 것이 좋습니다.

AWS Cloud Development Kit (AWS CDK)

- AWS CDK 인트로 워크숍

AWS CloudFormation

- AWS CloudFormation 랩
- AWS CloudFormation 워크숍

HashiCorp 테라폼

- AWS 테라폼 워크숍
- 테라폼 리포지토리용 CDK () GitHub

폴루미

- 폴루미 리포지토리 () GitHub
- AWS 폴루미 워크숍을 통한 현대화

리소스

AWS 설명서

- in을 사용하여 AWS CDK IaC 프로젝트를 TypeScript 만드는 모범 사례 (AWS 규범적 지침)
- cdk-nag 규칙 팩을 사용하여 AWS CDK 애플리케이션 또는 CloudFormation 템플릿에서 모범 사례를 확인하십시오 (규범적 지침).AWS
- 소개 AWS: 코드형 인프라 DevOps (백서)AWS
- AWS CloudFormation 설명서

기타 문서

- 코드형 인프라 시작하기 (HashiCorp웹 사이트)
- HashiCorp테라폼 문서
- 풀루미 설명서
- 풀루미: 코드형 인프라란 무엇인가 (풀루미 웹사이트)

도구

- GitHubcdk-nag ()
- cfn-nag () GitHub
- 테라테스트

기여자

작성

- 시아막 헤쉬마티, 선임 아키텍트, DevOps AWS
- 메이슨 케이힐, 선임 컨설턴트, DevOps AWS
- 샌딥 강가파디야, 선임 건축가, DevOps AWS
- 샌딥 가완데, 선임 수석 컨설턴트, AWS
- 라즈네시 타기, 리드 컨설턴트, AWS

리뷰

- 데이비드 하워튼, 시니어 인게이지먼트 매니저, AWS
- 스티븐 구겐하이머, 선임 클라우드 애플리케이션 아키텍트, AWS

테크니컬 라이팅

- 릴리 AbouHarb, 선임 테크니컬 라이터, AWS

문서 기록

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

변경 사항	설명	날짜
Pulumi 업데이트	Pulumi 장을 업데이트했습니다.	2026년 2월 17일
최초 게시	—	2024년 2월 23일

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