aws

개발자 가이드

AWS Infrastructure Composer

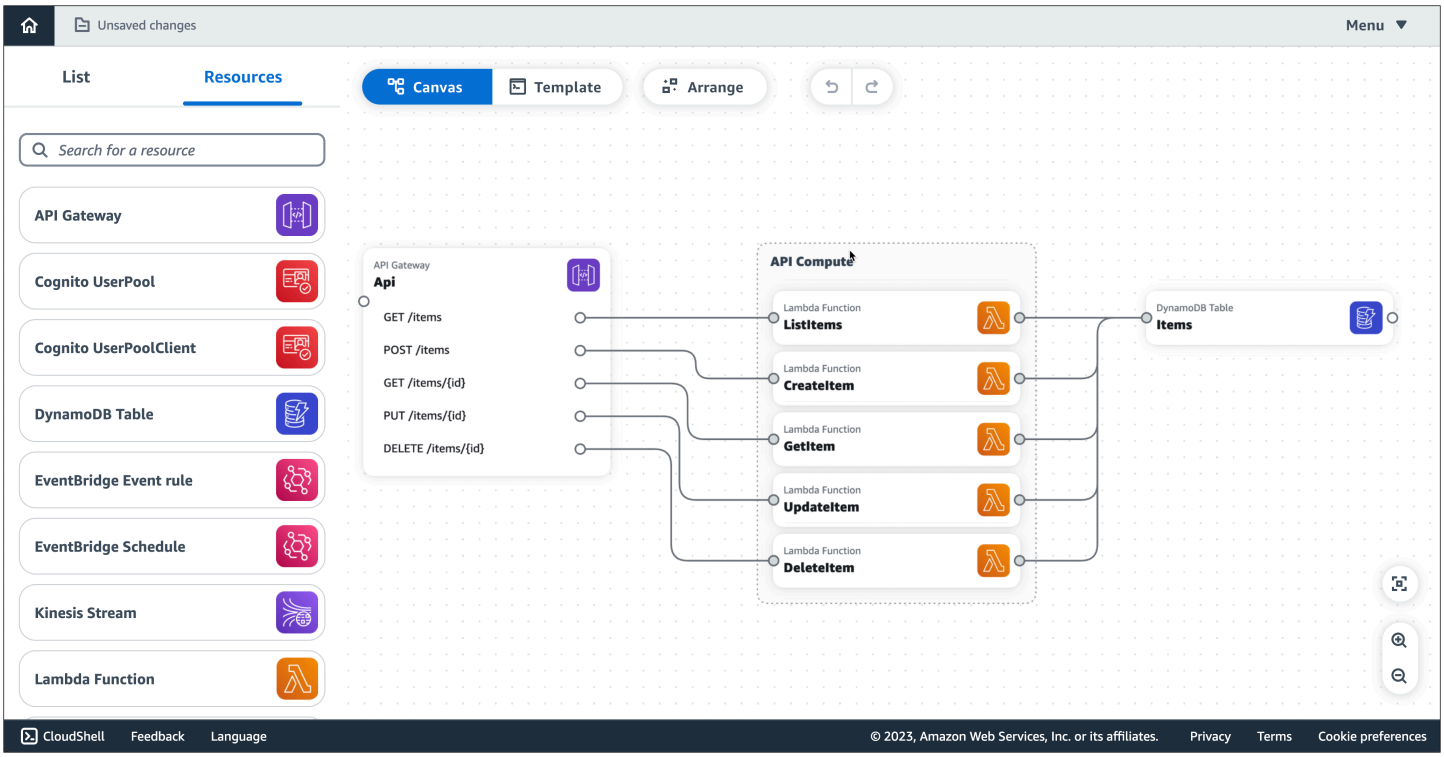
AWS Infrastructure Composer: 개발자 가이드

Table of Contents

AWS Infrastructure Composer란 무엇인가요?

AWS Infrastructure Composer 를 사용하면 최신 애플리케이션을 시각적으로 구성할 수 있습니다 AWS. 특히 Infrastructure Composer를 사용하면 전문가가 될 필요 AWS CloudFormation 없이에서 지원하는 모든 AWS 서비스의 최신 애플리케이션을 시각화, 구축 및 배포할 수 있습니다 CloudFormation.

AWS CloudFormation 인프라를 구성할 때 매력적인 drag-and-drop 인터페이스를 통해 Infrastructure Composer는 코드형 인프라(IaC) 템플릿을 생성하는 동시에 AWS 모범 사례를 따릅니다. 다음 이미지는 Infrastructure Composer의 시각적 캔버스에서 리소스를 드래그, 드롭, 구성 및 연결하는 것이 얼마나 쉬운지 보여줍니다.



Infrastructure Composer는 Infrastructure Composer 콘솔, AWS Toolkit for Visual Studio Code및 CloudFormation 콘솔 모드에서 사용할 수 있습니다.

주제

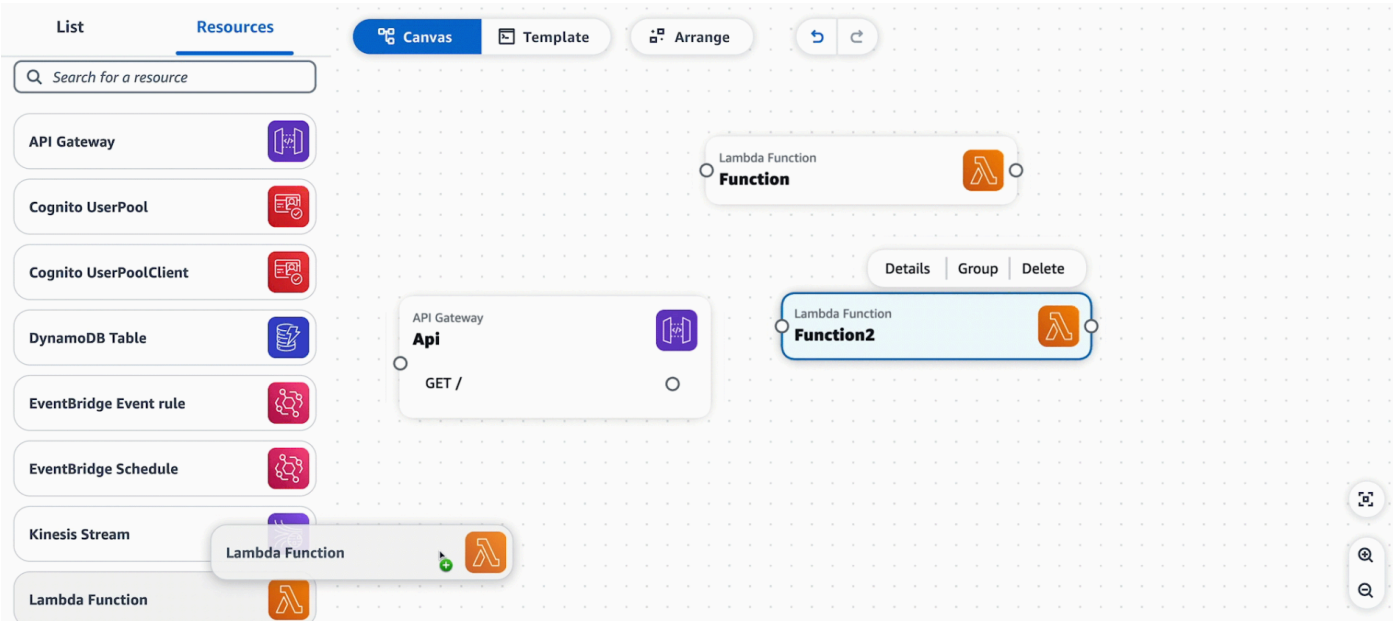
- 애플리케이션 아키텍처 구성
- 코드형 인프라(IaC) 템플릿 정의
- 기존 워크플로와 통합
- Infrastructure Composer에 액세스하는 방법

- 자세히 알아보기
- 다음 단계
- 에 대한 서버리스 개념 AWS Infrastructure Composer

애플리케이션 아키텍처 구성

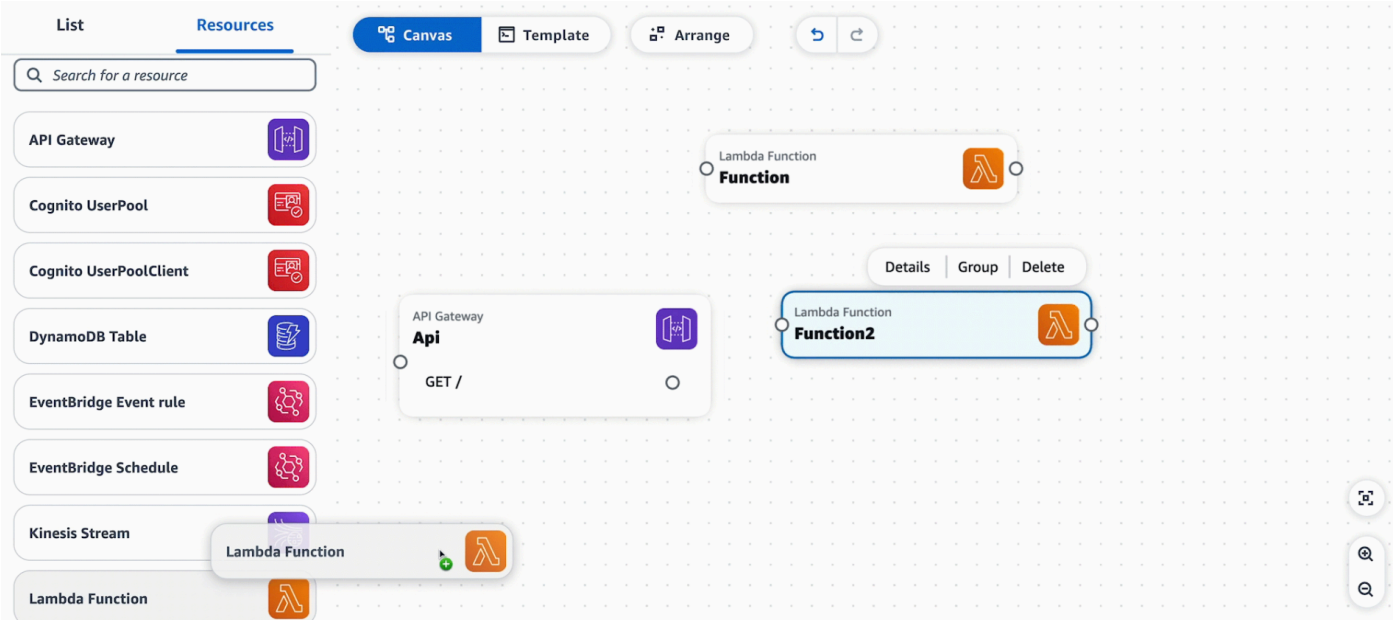
카드를 사용하여 빌드

Infrastructure Composer 캔버스에 카드를 배치하여 애플리케이션 아키텍처를 시각화하고 빌드합니다.



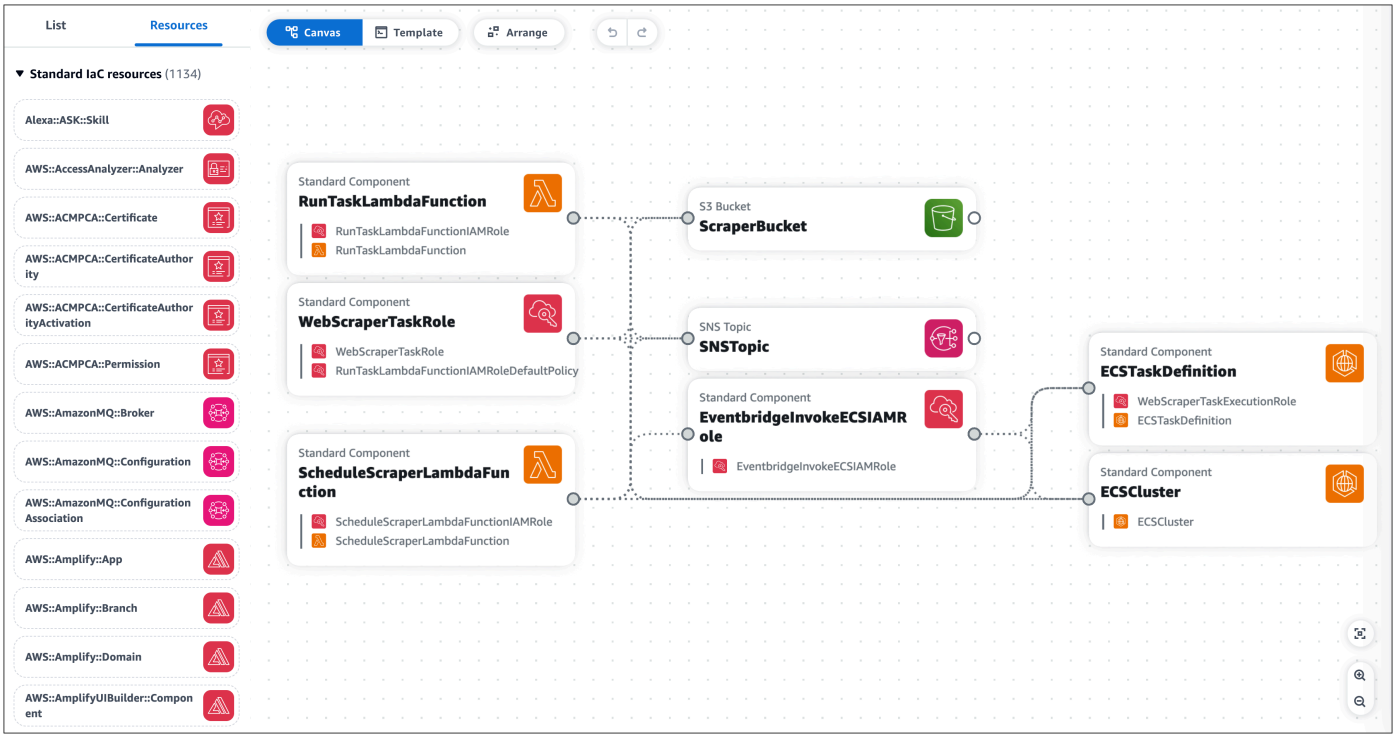
카드를 함께 연결

리소스를 함께 시각적으로 연결하여 리소스가 서로 상호 작용하는 방식을 구성합니다. 큐레이션된 속성 패널을 통해 속성을 추가로 지정합니다.



모든 AWS CloudFormation 리소스 작업

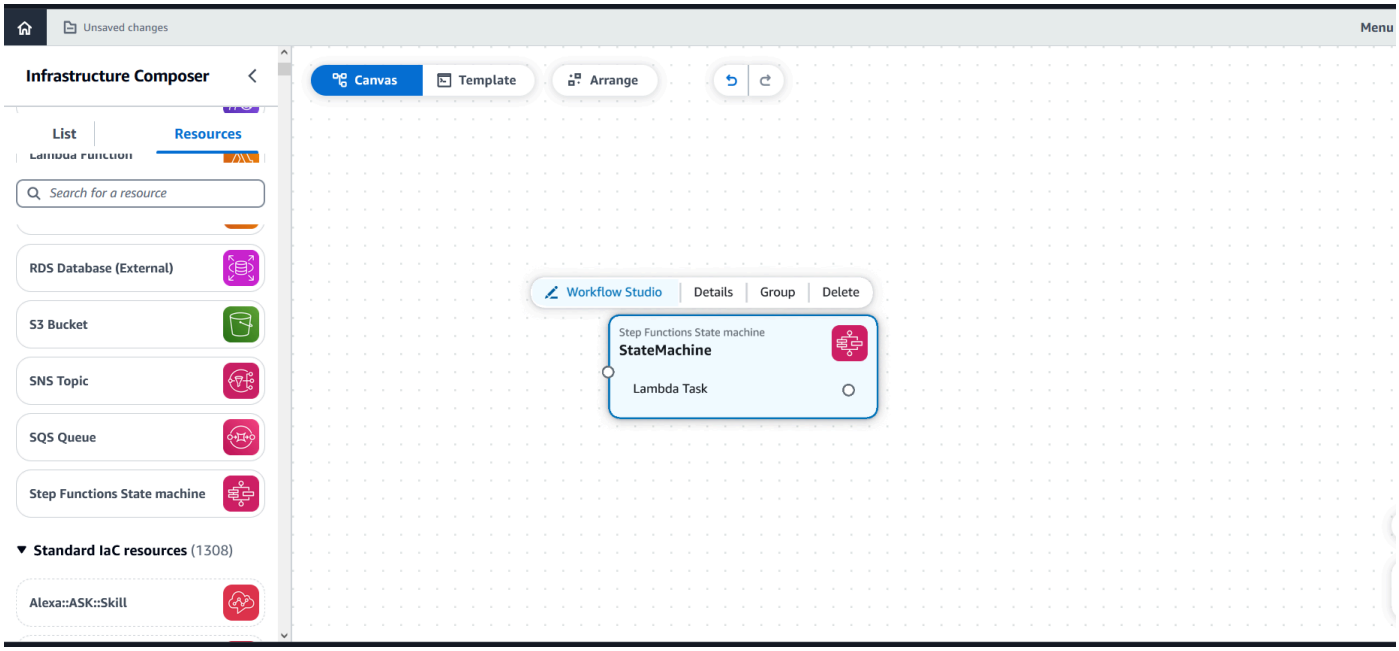
CloudFormation 리소스를 캔버스로 끌어 애플리케이션 아키텍처를 구성합니다. Infrastructure Composer는 리소스의 속성을 지정하는 데 사용할 수 있는 시작 IaC 템플릿을 제공합니다. 자세한 내용은 Infrastructure Composer에서 카드 구성 및 수정를 참조하세요.



를 사용하여 추가 기능에 액세스 AWS 서비스

애플리케이션을 빌드 AWS 서비스 할 때 일반적으로 함께 사용되거나 구성되는 Infrastructure Composer 기능입니다. 자세한 내용은 Amazon VPC와 통합를 참조하세요.

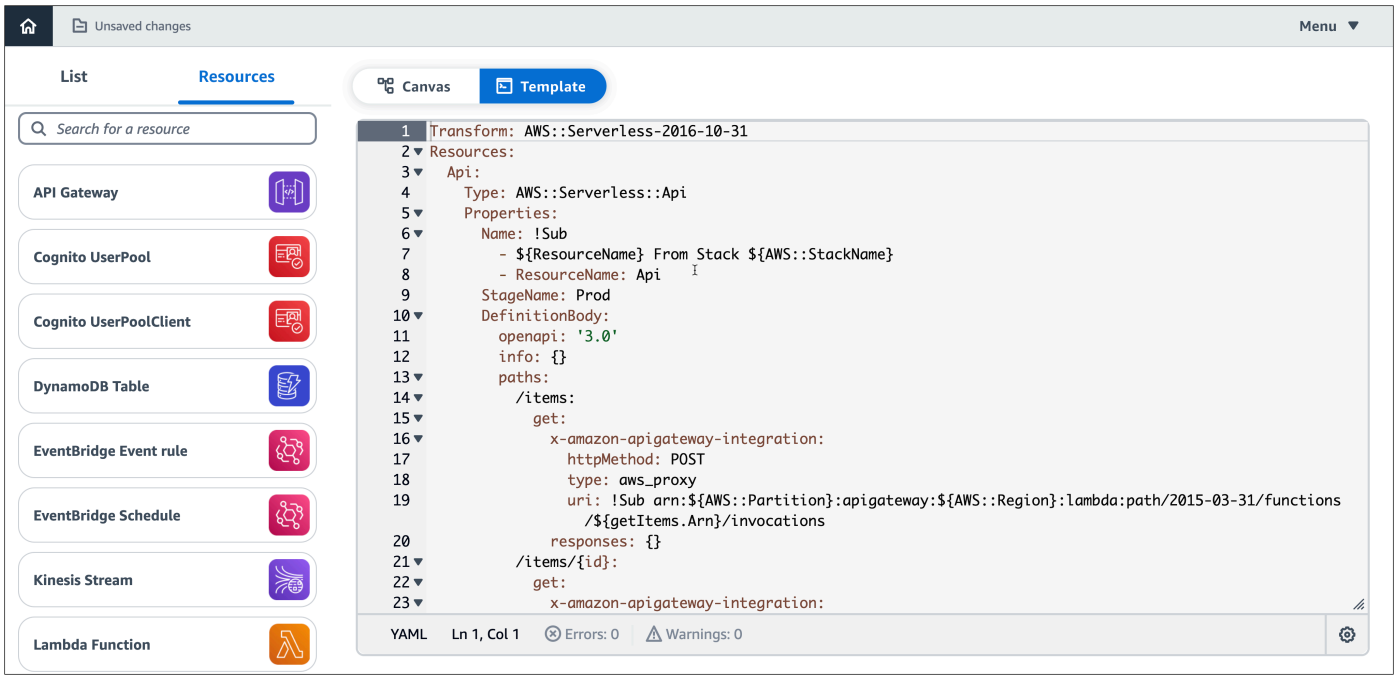
다음은 Infrastructure Composer 캔버스 내에서 Workflow Studio 직접 Step Functions를 시작하기 위한 통합을 제공하는 AWS Step Functions 기능의 예입니다.



코드형 인프라(IaC) 템플릿 정의

Infrastructure Composer에서 인프라 코드 생성

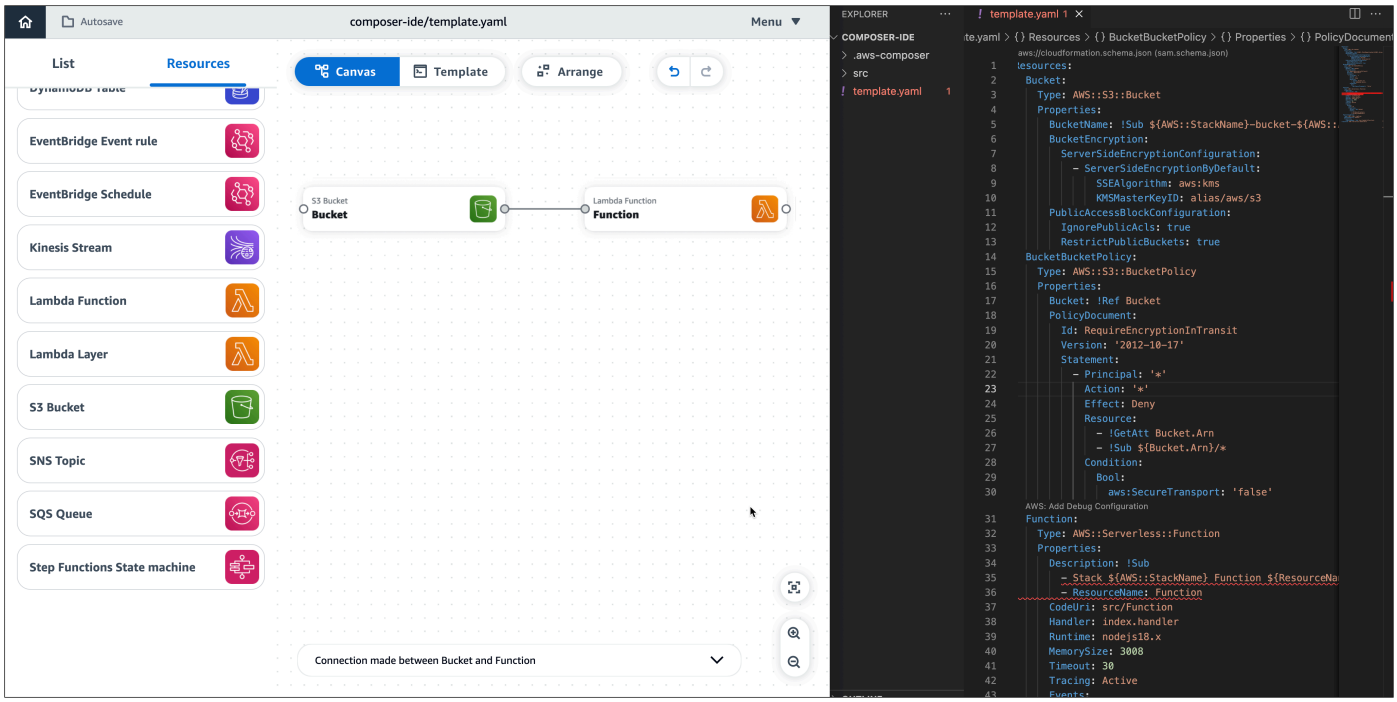
작성하면 Infrastructure Composer는 AWS 모범 사례에 따라 AWS CloudFormation 및 AWS Serverless Application Model (AWS SAM) 템플릿을 자동으로 생성합니다. Infrastructure Composer 내에서 직접 템플릿을 보고 수정할 수 있습니다. Infrastructure Composer는 시각적 캔버스와 템플릿 코드 간의 변경 사항을 자동으로 동기화합니다.



기존 워크플로와 통합

기존 템플릿 및 프로젝트 가져오기

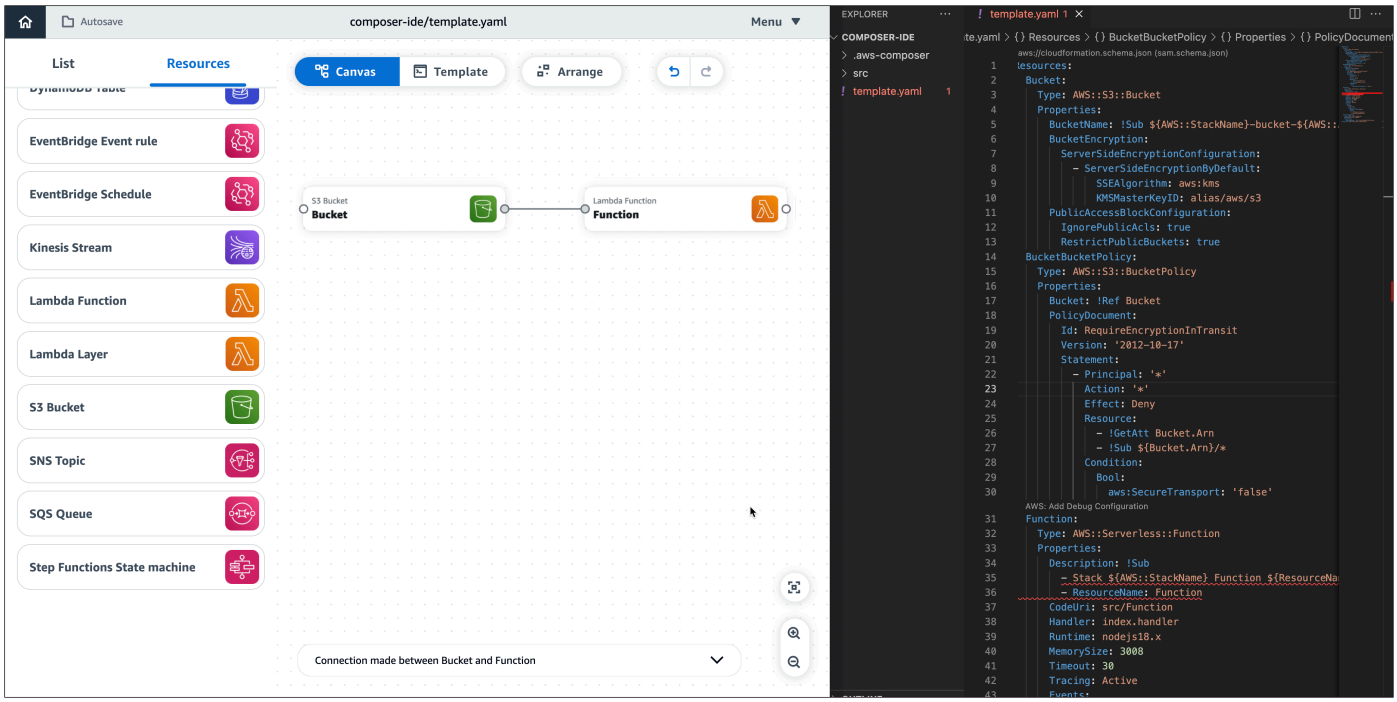
기존 CloudFormation 및 AWS SAM 템플릿을 가져와서 설계를 더 잘 이해하고 수정할 수 있도록 시각화합니다. Infrastructure Composer 내에서 생성한 템플릿을 내보내고 배포를 위해 기존 워크플로에 통합합니다.



Infrastructure Composer에 액세스하는 방법

Infrastructure Composer 콘솔에서

Infrastructure Composer 콘솔을 통해 Infrastructure Composer에 액세스하여 빠르게 시작할 수 있습니다. 또한 로컬 동기화 모드를 사용하여 Infrastructure Composer를 로컬 시스템과 자동으로 동기화하고 저장할 수 있습니다.



CloudFormation 콘솔에서

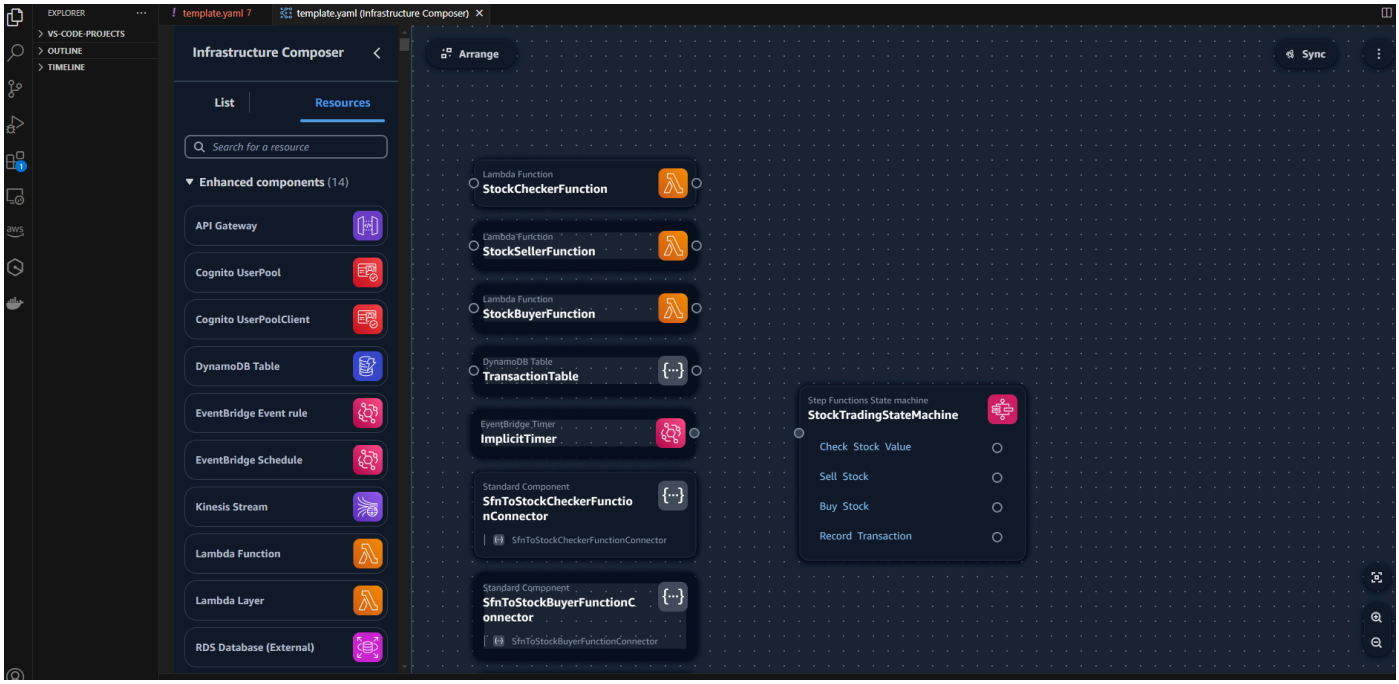
Infrastructure Composer 콘솔은 CloudFormation 스택 워크플로와 통합된 CloudFormation Designer의 개선 사항인 CloudFormation 콘솔 모드도 지원합니다. CloudFormation 이 새로운 도구는 이제 CloudFormation 템플릿을 시각화하는 데 권장되는 도구입니다.

Lambda 콘솔에서

Infrastructure Composer를 사용하면 Lambda 콘솔에서 Lambda 함수를 가져올 수도 있습니다. 자세한 내용은 Lambda 콘솔에서 Infrastructure Composer로 함수 가져오기를 참조하세요.

에서 AWS Toolkit for Visual Studio Code

Toolkit for VS Code 확장을 통해 Infrastructure Composer에 액세스하여 Infrastructure Composer를 로컬 개발 환경으로 가져옵니다.



자세히 알아보기

Infrastructure Composer에 대해 계속 알아보려면 다음 리소스를 참조하세요.

- Infrastructure Composer 카드
- 서버리스 애플리케이션 시각적 구성 및 생성 | 서버리스 업무 시간 – Infrastructure Composer 개요 및 데모.

다음 단계

Infrastructure Composer를 설정하려면 섹션을 참조하세요Infrastructure Composer 콘솔 시작하기.

에 대한 서버리스 개념 AWS Infrastructure Composer

를 사용하기 전에 기본 서버리스 개념에 대해 알아봅니다 AWS Infrastructure Composer.

서버리스 개념

이벤트 중심 아키텍처

서버리스 애플리케이션은 컴퓨팅용 및 데이터베이스 관리를 AWS Lambda 위한 Amazon DynamoDB와 같이 각각 특수 역할을 수행하는 개별 AWS 서비스로 구성됩니다. 이러한 서비스는 이벤트 기반 아키텍처를 통해 서로 느슨하게 통합됩니다. 이벤트 기반 아키텍처에 대해 자세히 알아보려면 이벤트 기반 아키텍처란 무엇인가요?를 참조하세요.

코드형 인프라(IaC)

코드형 인프라(IaC)는 개발자가 코드를 다루는 것과 동일한 방식으로 인프라를 취급하는 방식으로, 인프라 프로비저닝에도 동일하게 애플리케이션 코드 개발의 엄격함을 적용합니다. 템플릿 파일에서 인프라를 정의하고, 배포하고 AWS, 리소스를 AWS 생성합니다. IAC를 사용하면 프로비저닝 AWS 할 항목을 코드로 정의합니다. 자세한 내용은 백서의 DevOps 소개 AWS AWS 에서 코드형 인프라를 참조하세요.

서버리스 기술

AWS 서버리스 기술을 사용하면 자체 서버를 관리할 필요 없이 애플리케이션을 구축하고 실행할 수 있습니다. 모든 서버 관리는에서 수행되므로 자동 조정 및 기본 제공 고가용성과 같은 많은 이점을 AWS제공하므로 프로덕션에 신속하게 아이디어를 사용할 수 있습니다. 서버리스 기술을 사용하면 서버 관리 및 운영에 대해 걱정할 필요 없이 제품의 핵심에만 집중할 수 있습니다. 서버리스에 대한 자세한 내용은 서버리스 켜기 AWS를 참조하세요.

코어 AWS 서버리스 서비스에 대한 기본 소개는 Serverless 101: Understanding the serverless services at Serverless Land를 참조하세요.

Infrastructure Composer 카드

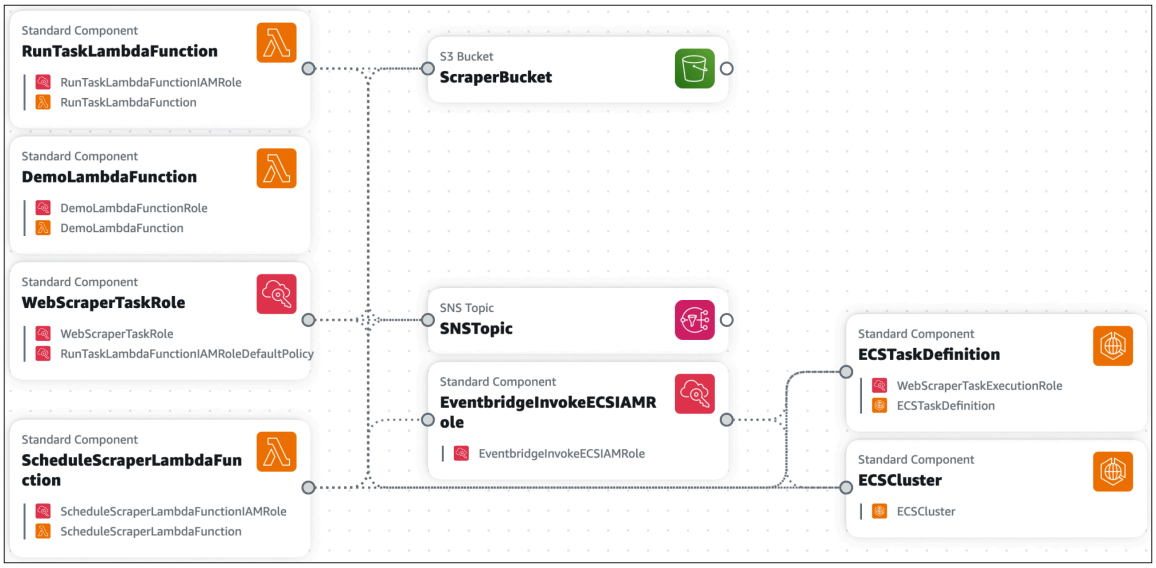
Infrastructure Composer는 CloudFormation 리소스에 대한 코드형 인프라(IaC) 작성 프로세스를 간소화합니다. Infrastructure Composer를 효과적으로 사용하려면 먼저 Infrastructure Composer 카드와 카드 연결이라는 두 가지 기본 개념을 이해해야 합니다.

Infrastructure Composer에서 카드는 CloudFormation 리소스를 나타냅니다. 카드에는 두 가지 일반적인 범주가 있습니다.

- 향상된 구성 요소 카드 - 사용 편의성과 기능을 향상하고 다양한 사용 사례에 맞게 설계된 단일 큐레이션된 카드에 결합된 CloudFormation 리소스 모음입니다. 향상된 구성 요소 카드는 Infrastructure Composer의 리소스 팔레트에 나열된 첫 번째 카드입니다.
- 표준 IaC 리소스 카드 - 단일 AWS CloudFormation 리소스입니다. 캔버스로 끌면 각 표준 IaC 리소스 카드에 표준 구성 요소 레이블이 지정되고 여러 리소스로 결합될 수 있습니다.

> **Note**
>
> 카드에 따라 표준 IaC 리소스 카드를 시각적 캔버스로 끌면 표준 구성 요소 카드에 레이블이 지정될 수 있습니다. 즉, 카드는 하나 이상의 표준 IaC 리소스 카드 모음입니다.

리소스 팔레트에서 일부 유형의 카드를 사용할 수 있지만 기존 CloudFormation 또는 AWS Serverless Application Model (AWS SAM) 템플릿을 Infrastructure Composer로 가져올 때 캔버스에 카드가 표시될 수도 있습니다. 다음 이미지는 다양한 카드 유형을 포함하는 가져온 애플리케이션의 예입니다.



주제

Infrastructure Composer의 향상된 구성 요소 카드

향상된 구성 요소 카드는 Infrastructure Composer에서 생성하고 관리합니다. 각 카드에는 애플리케이션을 빌드할 때 일반적으로 함께 사용되는 CloudFormation 리소스가 포함되어 있습니다 AWS. 인프라 코드는 AWS 모범 사례에 따라 Infrastructure Composer에서 생성합니다. 향상된 구성 요소 카드는 애플리케이션 설계를 시작하는 좋은 방법입니다.

향상된 구성 요소 카드는 리소스 팔레트의 향상된 구성 요소 섹션에서 사용할 수 있습니다.

향상된 구성 요소 카드를 Infrastructure Composer 내에서 완전히 구성하고 사용하여 서버리스 애플리케이션을 설계하고 구축할 수 있습니다. 기존 코드 없이 애플리케이션을 설계할 때는 향상된 구성 요소 카드를 사용하는 것이 좋습니다.

이 표에는 카드의 주요 리소스에 대한 AWS CloudFormation 또는 AWS Serverless Application Model (AWS SAM) 템플릿 사양에 대한 링크가 포함된 향상된 구성 요소가 표시됩니다.

Card	레퍼런스
Amazon API Gateway	AWS::Serverless::API
Amazon Cognito UserPool	AWS::Cognito::UserPool
Amazon Cognito UserPoolClient	AWS::Cognito::UserPoolClient
Amazon DynamoDB 테이블	AWS::DynamoDB::Table
Amazon EventBridge 이벤트 규칙	AWS::Events::Rule
EventBridge 일정	AWS::Scheduler::Schedule
Amazon Kinesis 스트림	AWS::Kinesis::Stream
AWS Lambda 함수	AWS::Serverless::Function

Card	레퍼런스
Lambda 계층	AWS::Serverless::LayerVersion
Amazon Simple Storage Service(Amazon S3) 버킷	AWS::S3::Bucket
Amazon Simple Notification Service(Amazon SNS) 주제	AWS::SNS::Topic
Amazon Simple Queue Service(Amazon SQS) 대기열	AWS::SQS::Queue
AWS Step Functions 상태 시스템	AWS::Serverless::StateMachine

예제

다음은 S3 버킷 향상된 구성 요소의 예입니다.

S3 버킷 구성 요소 카드를 캔버스로 끌어서 템플릿을 보면 템플릿에 다음 두 CloudFormation 리소스가 추가됩니다.

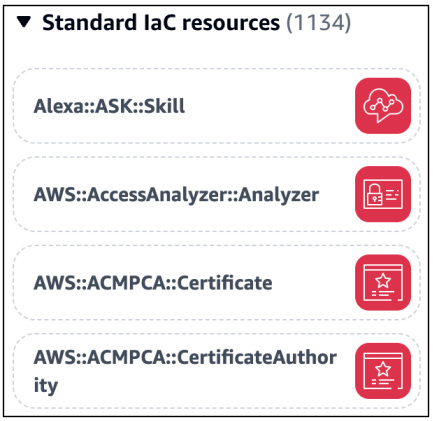
- `AWS::S3::Bucket`
- `AWS::S3::BucketPolicy`

S3 버킷 향상된 구성 요소 카드는 Amazon Simple Storage Service(Amazon S3) 버킷이 애플리케이션의 다른 서비스와 상호 작용하는 데 필요한 두 CloudFormation 리소스를 나타냅니다.

Infrastructure Composer의 표준 구성 요소 카드

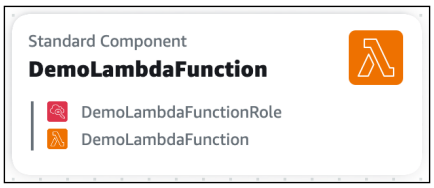
표준 구성 요소 카드를 Infrastructure Composer의 시각적 캔버스에 배치하기 전에 Infrastructure Composer의 리소스 팔레트에 표준(IaC) 리소스 카드로 나열됩니다. 표준(IaC) 리소스 카드는 단일 CloudFormation 리소스를 나타냅니다. 각 표준 IaC 리소스 카드는 시각적 캔버스에 배치되면 표준 구

성 요소로 레이블이 지정된 카드가 되며 여러 CloudFormation 리소스를 나타내도록 결합할 수 있습니다.

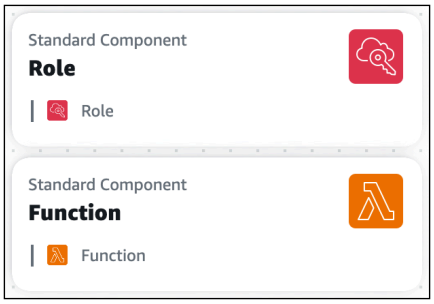


각 표준 IaC 리소스 카드는 CloudFormation 리소스 유형으로 식별할 수 있습니다. 다음은 리소스 유형을 나타내는 표준 IaC `AWS::ECS::Cluster` CloudFormation 리소스 카드의 예입니다.

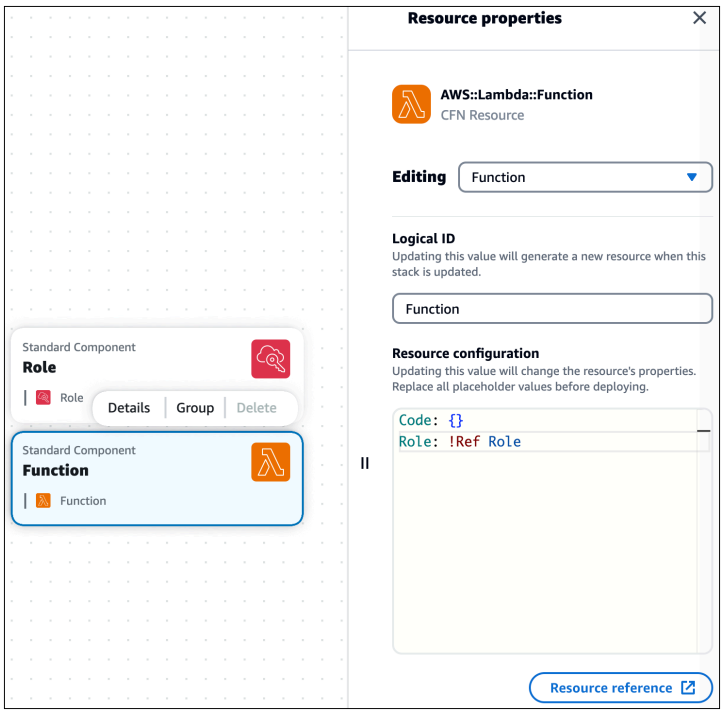
각 표준 구성 요소 카드는 포함된 CloudFormation 리소스를 시각화합니다. 다음은 두 개의 표준 IaC 리소스가 포함된 표준 구성 요소 카드의 예입니다.



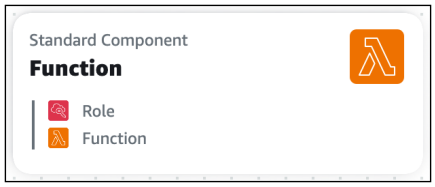
표준 구성 요소 카드의 속성을 구성할 때 Infrastructure Composer는 관련 카드를 함께 결합할 수 있습니다. 예를 들어, 다음은 두 가지 표준 구성 요소 카드입니다.



리소스를 나타내는 표준 구성 요소 카드의 `AWS::Lambda::Function` 리소스 속성 패널에서 논리적 ID로 AWS Identity and Access Management (IAM) 역할을 참조합니다.



템플릿을 저장한 후 두 표준 구성 요소 카드가 단일 표준 구성 요소 카드로 결합됩니다.



Infrastructure Composer의 카드 연결

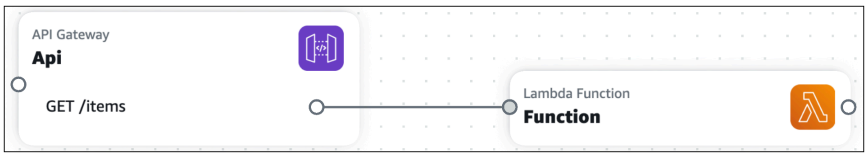
에서는 두 카드 간의 AWS Infrastructure Composer연결이 선으로 시각적으로 표시됩니다. 이러한 줄은 애플리케이션 내의 이벤트 기반 관계를 나타냅니다.

주제

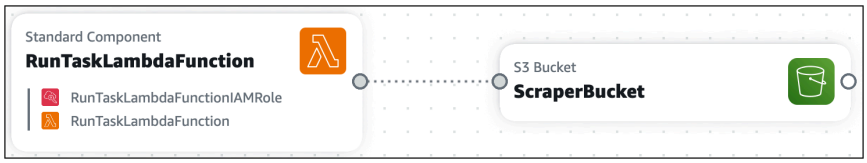
- 카드 간 연결
- 향상된 구성 요소 카드 간의 연결
- 표준 IaC 리소스 카드와의 연결

카드 간 연결

카드를 함께 연결하는 방법은 카드 유형에 따라 다릅니다. 각 향상된 카드에는 커넥터 포트가 하나 이상 있습니다. 연결하려면 커넥터 포트를 하나 선택하고 다른 카드의 포트로 드래그하면 Infrastructure Composer가 두 리소스를 연결하거나이 구성이 지원되지 않는다는 메시지를 표시합니다.



위에서 볼 수 있듯이 향상된 구성 요소 카드 사이의 선은 단색입니다. 반대로 표준 IaC 리소스 카드(표준 구성 요소 카드라고도 함)에는 커넥터 포트가 없습니다. 이러한 카드의 경우 애플리케이션의 템플릿에 이러한 이벤트 기반 관계를 지정해야 하며, Infrastructure Composer는 연결을 자동으로 감지하고 카드 사이에 점선으로 시각화합니다.

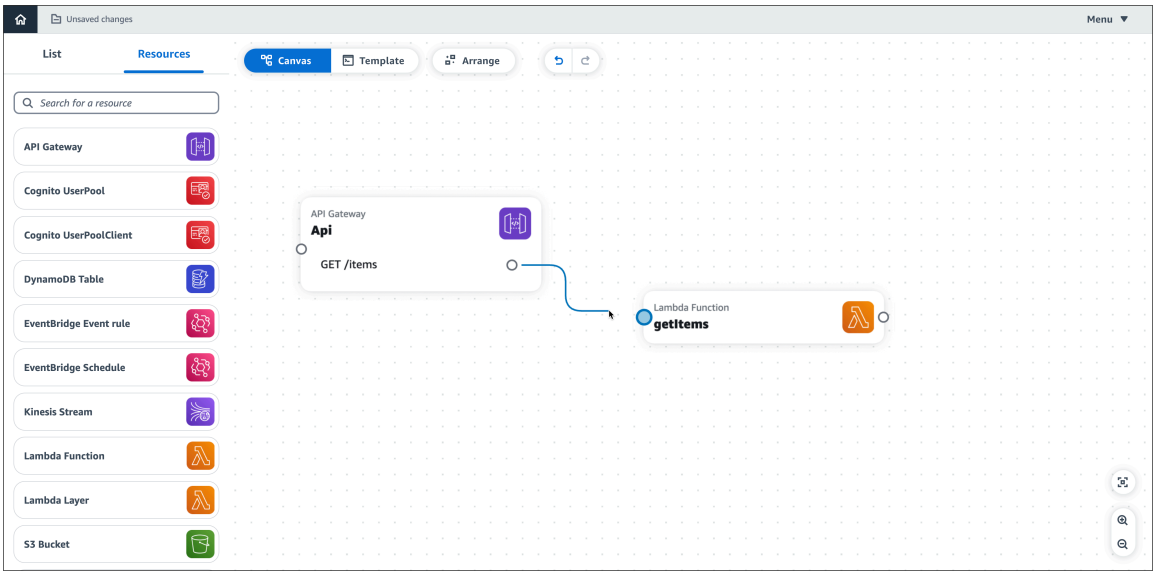


자세한 내용은 아래 섹션을 참조하세요.

향상된 구성 요소 카드 간의 연결

Infrastructure Composer에서는 두 개의 향상된 구성 요소 카드 간의 연결이 실선으로 시각적으로 표시됩니다. 이러한 줄은 애플리케이션 내의 이벤트 기반 관계를 나타냅니다.

두 카드를 연결하려면 한 카드의 포트를 클릭하고 다른 카드의 포트로 끕니다.



> **Note**
>
> 표준 IaC 리소스 카드에는 커넥터 포트가 없습니다. 이러한 카드의 경우 애플리케이션의 템플릿에 이벤트 기반 관계를 지정해야 하며, Infrastructure Composer는 연결을 자동으로 감지하고 카드 사이에 점선으로 시각화합니다.

자세한 내용은 Infrastructure Composer의 시각적 캔버스에서 카드 연결 단원을 참조하십시오.

향상된 구성 요소 카드 프로비저닝

줄로 시각적으로 표시된 두 카드 간의 연결은 필요한 경우 다음을 프로비저닝합니다.

- AWS Identity and Access Management (IAM) 정책
- 환경 변수
- 이벤트

IAM 정책

리소스에 다른 리소스를 호출할 권한이 필요한 경우 Infrastructure Composer는 AWS Serverless Application Model (AWS SAM) 정책 템플릿을 사용하여 리소스 기반 정책을 프로비저닝합니다.

- IAM 권한 및 정책에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 개요: 권한 및 정책을 참조하세요.
- AWS SAM 정책 템플릿에 대한 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 AWS SAM 정책 템플릿을 참조하세요.

환경 변수

환경 변수는 리소스의 동작에 영향을 미치도록 변경할 수 있는 임시 값입니다. 필요한 경우 Infrastructure Composer는 리소스 간의 환경 변수를 활용하도록 인프라 코드를 정의합니다.

이벤트

리소스는 다양한 유형의 이벤트를 통해 다른 리소스를 호출할 수 있습니다. 필요한 경우 Infrastructure Composer는 리소스가 이벤트 유형을 통해 상호 작용하는 데 필요한 인프라 코드를 정의합니다.

표준 IaC 리소스 카드와의 연결

모든 CloudFormation 리소스는 리소스 팔레트에서 표준 IaC 리소스 카드로 사용할 수 있습니다. 표준 IaC 리소스 카드를 캔버스로 끌면 표준 IaC 리소스 카드가 표준 구성 요소 카드가 되고, 그러면 Infrastructure Composer에 애플리케이션의 리소스에 대한 시작 템플릿을 생성하라는 메시지가 표시됩니다.

자세한 내용은 Infrastructure Composer의 표준 카드 단원을 참조하십시오.

Infrastructure Composer 콘솔 시작하기

이 섹션의 주제를 사용하여 시각적 캔버스를 사용하여 애플리케이션을 설계하는 방법을 설정하고 AWS Infrastructure Composer 알아봅니다. 이 섹션의 둘러보기 및 자습서는 기본 사용자 환경인 Infrastructure Composer 콘솔에 표시됩니다. 이 섹션의 주제에서는 Infrastructure Composer를 사용하기 위한 사전 조건을 완료하고, Infrastructure Composer 콘솔을 사용하고, 프로젝트를 로드 및 수정하고, 첫 번째 애플리케이션을 빌드하는 방법을 보여줍니다.

Infrastructure Composer는 AWS Toolkit for Visual Studio Code 및 CloudFormation 콘솔 모드에서도 사용할 수 있습니다. 도구 간의 경험은 일반적으로 동일하지만 각 도구 간에는 몇 가지 차이점이 있습니다. 이러한 각 도구에서 Infrastructure Composer를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer를 사용할 수 있는 위치.

주제

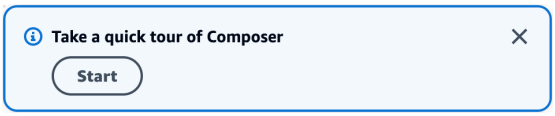
- Infrastructure Composer 콘솔 둘러보기
- Infrastructure Composer 데모 프로젝트 로드 및 수정
- Infrastructure Composer를 사용하여 첫 번째 애플리케이션 구축

Infrastructure Composer 콘솔 둘러보기

AWS Infrastructure Composer 작동 방식에 대한 일반적인 아이디어를 얻으려면 Infrastructure Composer 콘솔에 내장된를 둘러보세요. Infrastructure Composer 콘솔에 대한 개요는 섹션을 참조하세요Infrastructure Composer 콘솔 둘러보기. Infrastructure Composer 사용에 대한 자세한 지침은 섹션을 참조하세요에서를 작성하는 방법 AWS Infrastructure Composer.

Infrastructure Composer 둘러보기

1. Infrastructure Composer 콘솔에 로그인합니다.
2. 홈 페이지에서 데모 열기를 선택합니다.
3. 오른쪽 상단 모서리의 Composer 빠른 둘러보기 창에서 시작을 선택합니다.



4. Composer 투어 창에서 다음을 수행합니다.
   - 다음 단계로 이동하려면 다음을 선택합니다.

- 이전 단계로 돌아가려면 이전을 선택합니다.
- 마지막 단계에서 둘러보기를 완료하려면 종료를 선택합니다.

이 투어에서는 카드 사용, 구성 및 연결과 같은 기본 Infrastructure Composer 기능에 대한 간략한 개요를 제공합니다. 자세한 정보는 에서를 작성하는 방법 AWS Infrastructure Composer 섹션을 참조하세요.

다음 단계

Infrastructure Composer에서 프로젝트를 로드하고 수정하려면 섹션을 참조하세요Infrastructure Composer 데모 프로젝트 로드 및 수정.

Infrastructure Composer 데모 프로젝트 로드 및 수정

이 자습서를 사용하여 Infrastructure Composer의 사용자 인터페이스에 익숙해지고 Infrastructure Composer 데모 프로젝트를 로드, 수정 및 저장하는 방법을 알아봅니다.

이 자습서는 Infrastructure Composer 콘솔에서 수행됩니다. 완료되면를 시작할 준비가 된 것입니다Infrastructure Composer를 사용하여 첫 번째 애플리케이션 구축.

주제

1단계: 데모 열기

데모 프로젝트를 생성하여 Infrastructure Composer 사용을 시작합니다.

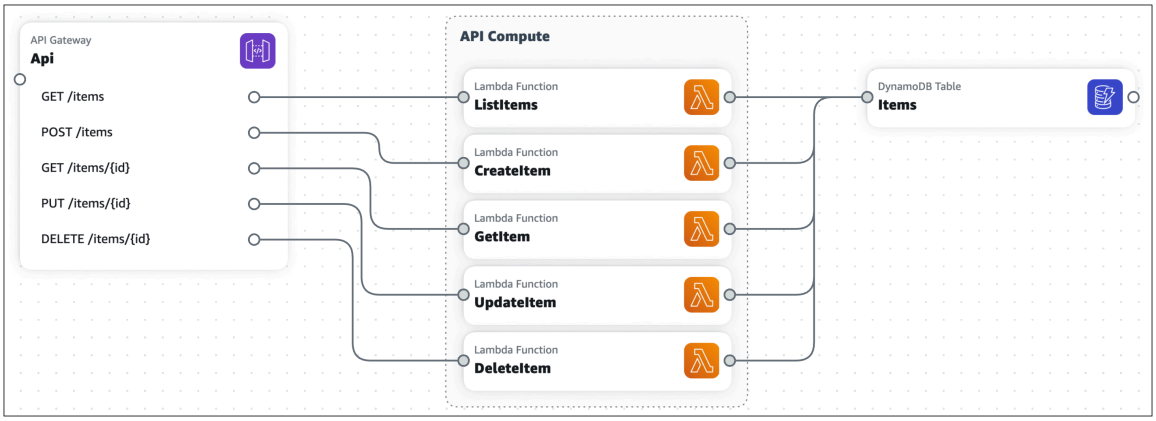
데모 프로젝트를 생성하려면

1. Infrastructure Composer 콘솔에 로그인합니다.
2. 홈 페이지에서 데모 열기를 선택합니다.

데모 애플리케이션은 다음을 포함하는 기본 생성, 읽기, 삭제 및 업데이트(CRUD) 서버리스 애플리케이션입니다.

- 5개의 경로가 있는 Amazon API Gateway 리소스입니다.
- AWS Lambda 함수 5개.
- Amazon DynamoDB 테이블.

데모 이미지는 다음과 같습니다.

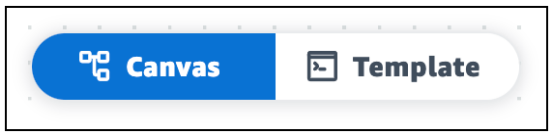


2단계: Infrastructure Composer의 시각적 캔버스 살펴보기

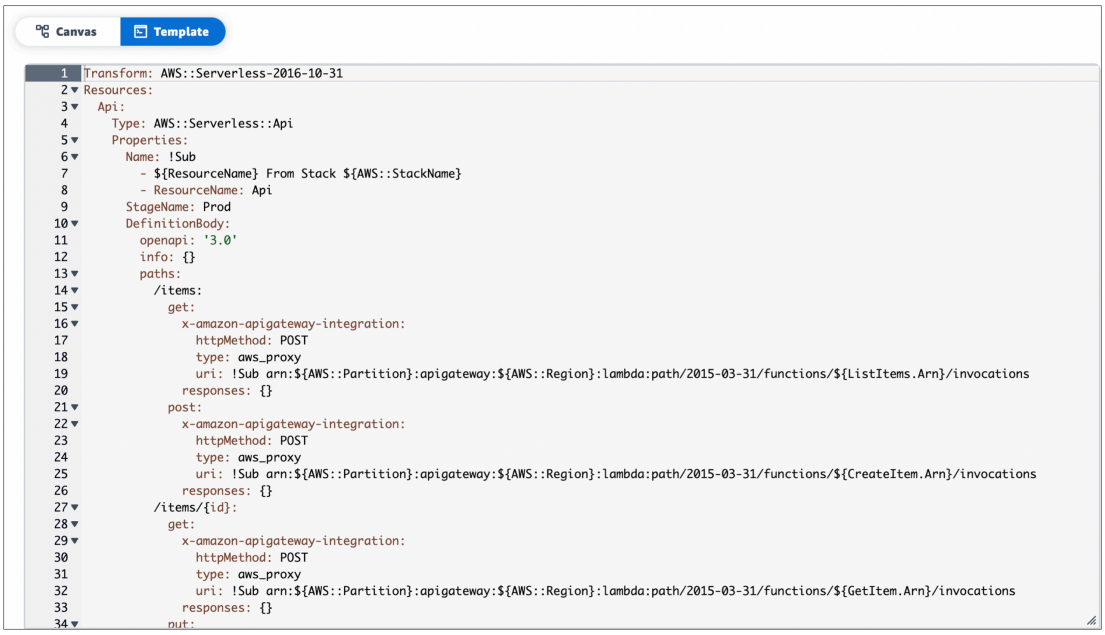
Infrastructure Composer 데모 프로젝트를 빌드하기 위한 시각적 캔버스의 기능에 대해 알아봅니다. 시각적 캔버스 레이아웃에 대한 개요는 섹션을 참조하세요시각적 개요.

시각적 캔버스의 기능을 탐색하려면

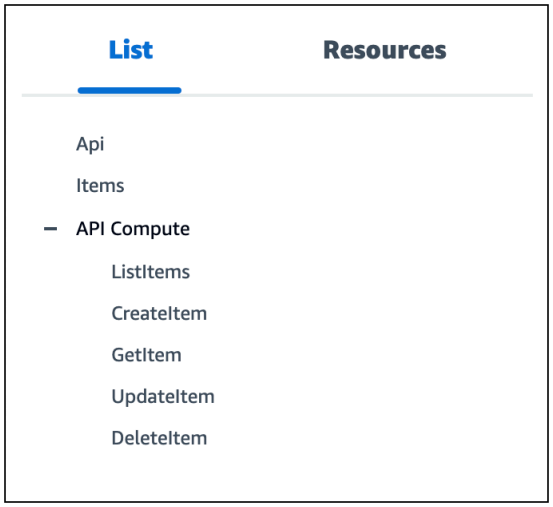
1. 새 애플리케이션 프로젝트 또는 기존 애플리케이션 프로젝트를 열면 기본 보기 영역 위에 표시된 대로 Infrastructure Composer가 캔버스 보기를 로드합니다.

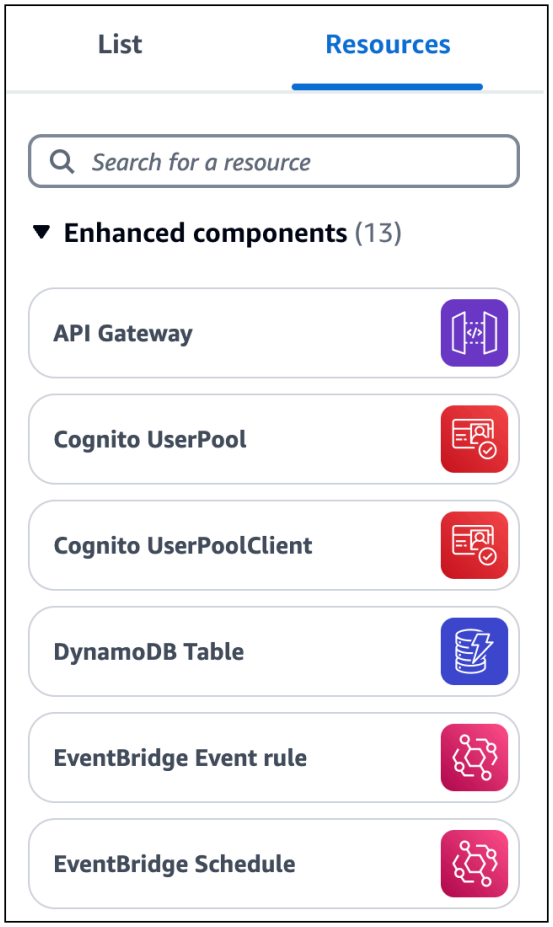
   애플리케이션의 인프라 코드를 기본 보기 영역에 표시하려면 템플릿을 선택합니다. 예를 들어 Infrastructure Composer 데모 프로젝트의 AWS Serverless Application Model (AWS SAM) 템플릿 보기는 다음과 같습니다.



2. 애플리케이션의 캔버스 보기를 다시 표시하려면 캔버스를 선택합니다.
3. 트리 보기로 구성된 애플리케이션의 리소스를 표시하려면 목록을 선택합니다.



4. 리소스 팔레트를 표시하려면 리소스를 선택합니다. 이 팔레트에는 애플리케이션 아키텍처를 확장하는 데 사용할 수 있는 카드가 있습니다. 카드를 검색하거나 목록을 스크롤할 수 있습니다.



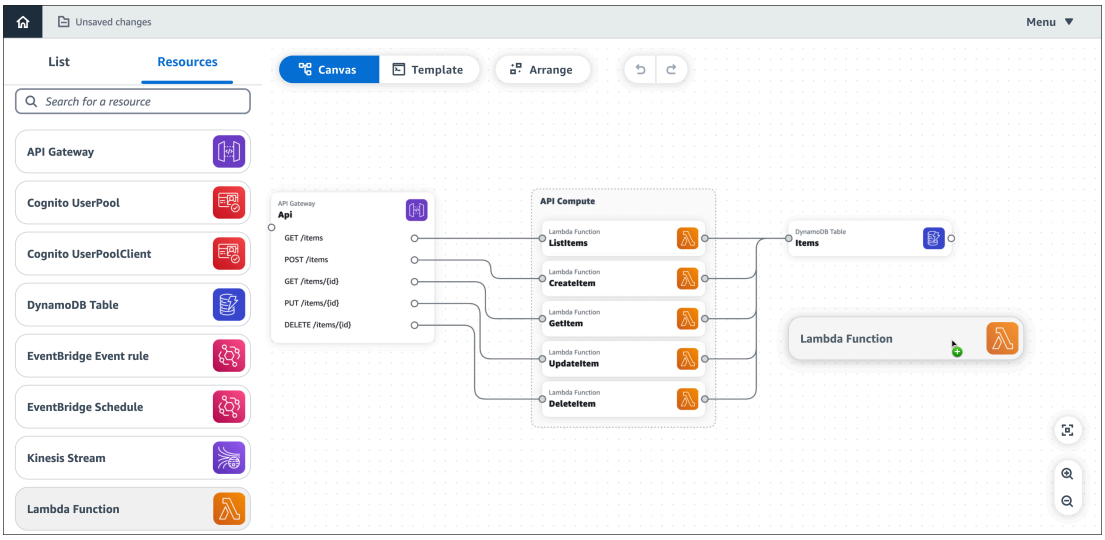
5. 시각적 캔버스 주위를 이동하려면 기본 제스처를 사용합니다. 자세한 내용은 캔버스에 카드 배치 단원을 참조하십시오.

3단계: 애플리케이션 아키텍처 확장

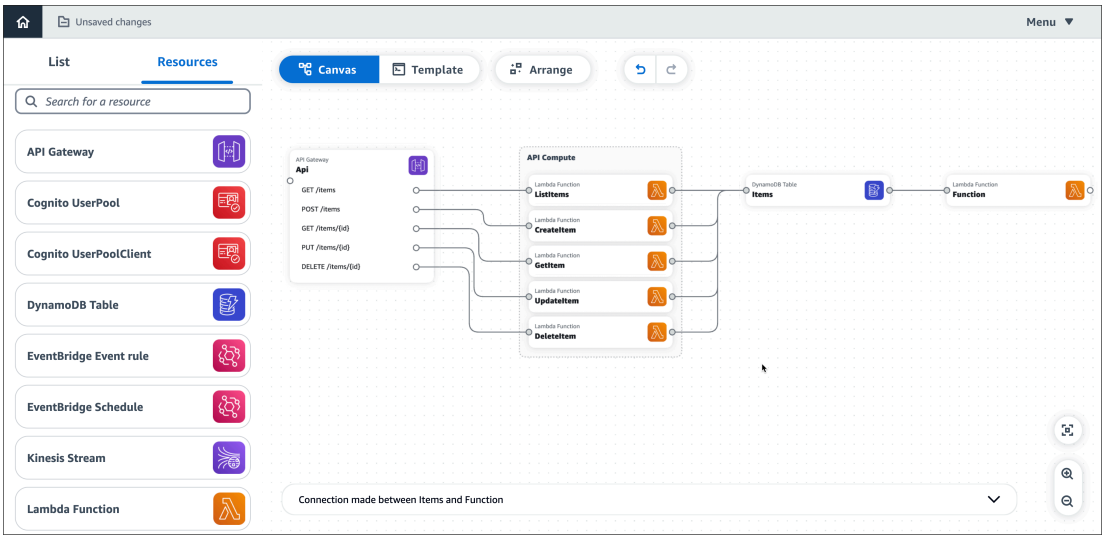
이 단계에서는 DynamoDB 테이블에 Lambda 함수를 추가하여 애플리케이션 아키텍처를 확장합니다.

DynamoDB 테이블에 Lambda 함수를 추가하려면

1. 리소스 팔레트(리소스)에서 Lambda 함수 향상된 구성 요소 카드를 캔버스로 DynamoDB 테이블 카드 오른쪽에 드래그합니다.



2. DynamoDB 테이블을 Lambda 함수에 연결합니다. 연결하려면 DynamoDB 테이블 카드의 오른쪽 포트를 클릭하고 Lambda 함수 카드의 왼쪽 포트로 끕니다.
3. 정렬을 선택하여 캔버스 보기에서 카드를 구성합니다.



4. Lambda 함수를 구성합니다. 구성하려면 다음 중 하나를 수행합니다.
   - 캔버스 보기에서 리소스 속성 패널에서 함수의 속성을 수정합니다. 패널을 열려면 Lambda 함수 카드를 두 번 클릭합니다. 또는 카드를 선택한 다음 세부 정보를 선택합니다. 리소스 속성 패널에 나열된 구성 가능한 Lambda 함수 속성에 대한 자세한 내용은 AWS Lambda 개발자 안내서를 참조하세요.
   - 템플릿 보기에서 함수()의 코드를 수정합니다`AWS::Serverless::Function`. Infrastructure Composer는 변경 사항을 캔버스에 자동으로 동기화합니다. 템플릿의 함수 리소스에 AWS SAM 대한 자세한 내용은 AWS SAM 리소스 및 속성 참조의 AWS::Serverless::Function을 참조하세요.

4단계: 애플리케이션 저장

애플리케이션 템플릿을 로컬 시스템에 수동으로 저장하거나 로컬 동기화를 활성화하여 애플리케이션을 저장합니다.

애플리케이션 템플릿을 수동으로 저장하려면

1. 메뉴에서 저장 > 템플릿 파일 저장을 선택합니다.
2. 템플릿의 이름을 입력하고 로컬 시스템에서 템플릿을 저장할 위치를 선택합니다. 저장을 누릅니다.

로컬 동기화 활성화에 대한 지침은 섹션을 참조하세요Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장.

다음 단계

첫 번째 애플리케이션 빌드를 시작하려면 섹션을 참조하세요Infrastructure Composer를 사용하여 첫 번째 애플리케이션 구축.

Infrastructure Composer를 사용하여 첫 번째 애플리케이션 구축

이 자습서에서는 AWS Infrastructure Composer 를 사용하여 데이터베이스의 사용자를 관리하는 생성, 읽기, 업데이트 및 삭제(CRUD) 서버리스 애플리케이션을 빌드합니다.

이 자습서에서는에서 Infrastructure Composer를 사용합니다 AWS Management Console. Google Chrome 또는 및 Microsoft Edge전체 화면 브라우저 창을 사용하는 것이 좋습니다.

서버리스를 처음 사용하시나요?

다음 주제에 대해 기본적으로 이해하는 것이 좋습니다.

- 이벤트 중심 아키텍처
- 코드형 인프라(IaC)
- 서버리스 기술

자세한 내용은 에 대한 서버리스 개념 AWS Infrastructure Composer를 참조하세요.

주제

리소스 속성 참조

애플리케이션을 빌드하는 동안이 표를 참조하여 Amazon API Gateway 및 AWS Lambda 리소스의 속성을 구성합니다.

방법	경로	함수 이름
GET	/항목	getItems
GET	/items/{id}	getItem
PUT	/items/{id}	updateItem
POST	/항목	addItem
DELETE	/items/{id}	deleteItem

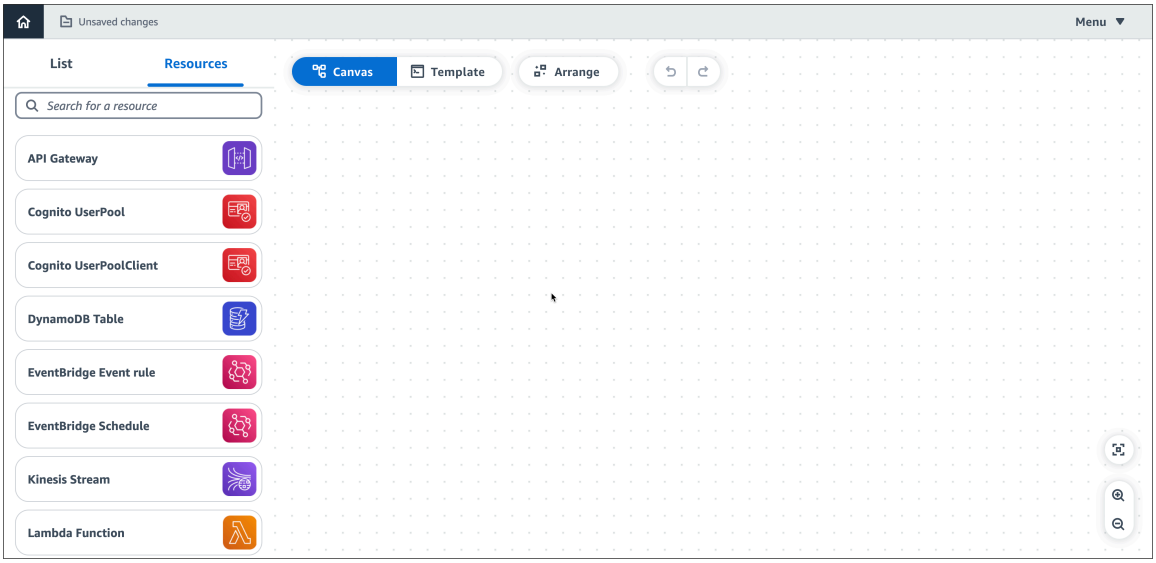
1단계: 프로젝트 생성

CRUD 서버리스 애플리케이션을 시작하려면 Infrastructure Composer에서 새 프로젝트를 생성하고 로컬 동기화를 활성화합니다.

새 빈 프로젝트를 생성하려면

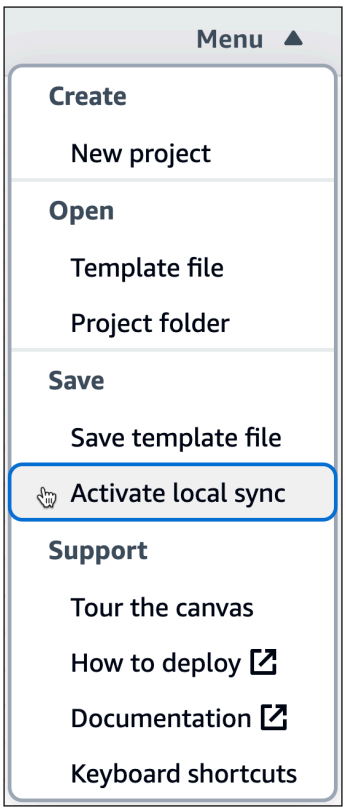
1. Infrastructure Composer 콘솔에 로그인합니다.
2. 홈 페이지에서 프로젝트 생성을 선택합니다.

다음 이미지와 같이 Infrastructure Composer는 시각적 캔버스를 열고 시작(비어 있음) 애플리케이션 템플릿을 로드합니다.



로컬 동기화를 활성화하려면

1. Infrastructure Composer 메뉴에서 저장 > 로컬 동기화 활성화를 선택합니다.



2. 프로젝트 위치에서 폴더 선택을 누르고 디렉터리를 선택합니다. 여기서 Infrastructure Composer는 설계에 따라 템플릿 파일과 폴더를 저장하고 동기화합니다.

   프로젝트 위치에는 기존 애플리케이션 템플릿이 포함되어서는 안 됩니다.

   > **Note**
   >
   > 로컬 동기화에는 파일 시스템 액세스 API를 지원하는 브라우저가 필요합니다. 자세한 내용은 Data Infrastructure Composer는에 액세스할 수 있습니다. 단원을 참조하십시오.

3. 액세스를 허용하라는 메시지가 표시되면 파일 보기를 선택합니다.
4. 활성화를 눌러 로컬 동기화를 켭니다. 변경 사항을 저장하라는 메시지가 표시되면 변경 사항 저장을 선택합니다.

   활성화하면 캔버스의 왼쪽 상단에 자동 저장 표시기가 표시됩니다.

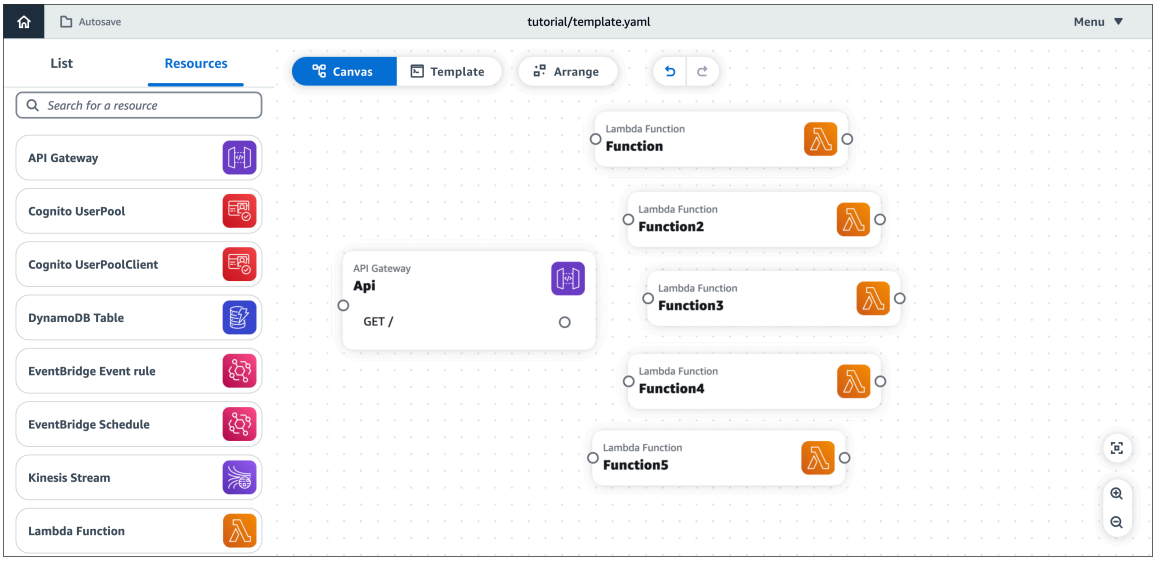
2단계: 캔버스에 카드 추가

API Gateway REST API와 5개의 Lambda 함수로 시작하여 향상된 구성 요소 카드를 사용하여 애플리케이션 아키텍처를 설계합니다.

캔버스에 API Gateway 및 Lambda 카드를 추가하려면

리소스 팔레트의 향상된 구성 요소 섹션에서 다음을 수행합니다.

1. API Gateway 카드를 캔버스로 드래그합니다.
2. Lambda 함수 카드를 캔버스로 드래그합니다. 캔버스에 Lambda 함수 카드를 5개 추가할 때까지 반복합니다.



3단계: API Gateway REST API 구성

그런 다음 API Gateway 카드에 5개의 경로를 추가합니다.

API Gateway 카드에 경로를 추가하려면

1. API Gateway 카드의 리소스 속성 패널을 엽니다. 패널을 열려면 카드를 두 번 클릭합니다. 또는 카드를 선택한 다음 세부 정보를 선택합니다.
2. 리소스 속성 패널의 라우팅에서 다음을 수행합니다.

> **Note**
>
> 다음 각 경로에 대해 리소스 속성 참조 테이블에 지정된 HTTP 메서드 및 경로 값을 사용합니다.

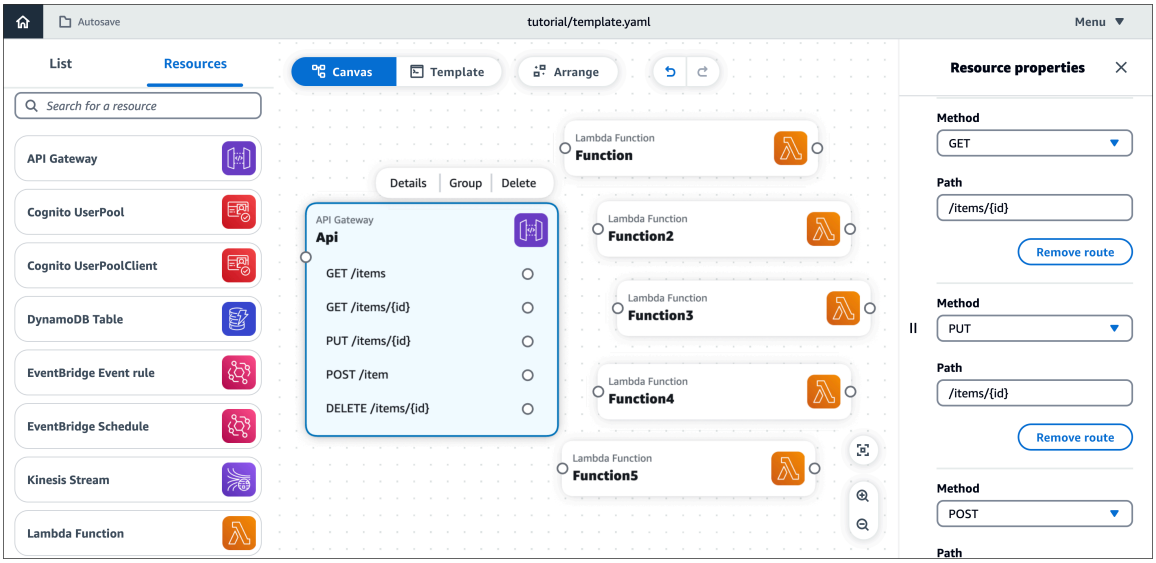
   a. 메서드에서 지정된 HTTP 메서드를 선택합니다. GET을 예로 들 수 있습니다.

b. 경로에 지정된 경로를 입력합니다. 예를 들어 **`/items`**입니다.

c. 경로 추가를 선택합니다.

d. 지정된 경로 5개를 모두 추가할 때까지 이전 단계를 반복합니다.

3. 저장을 선택합니다.

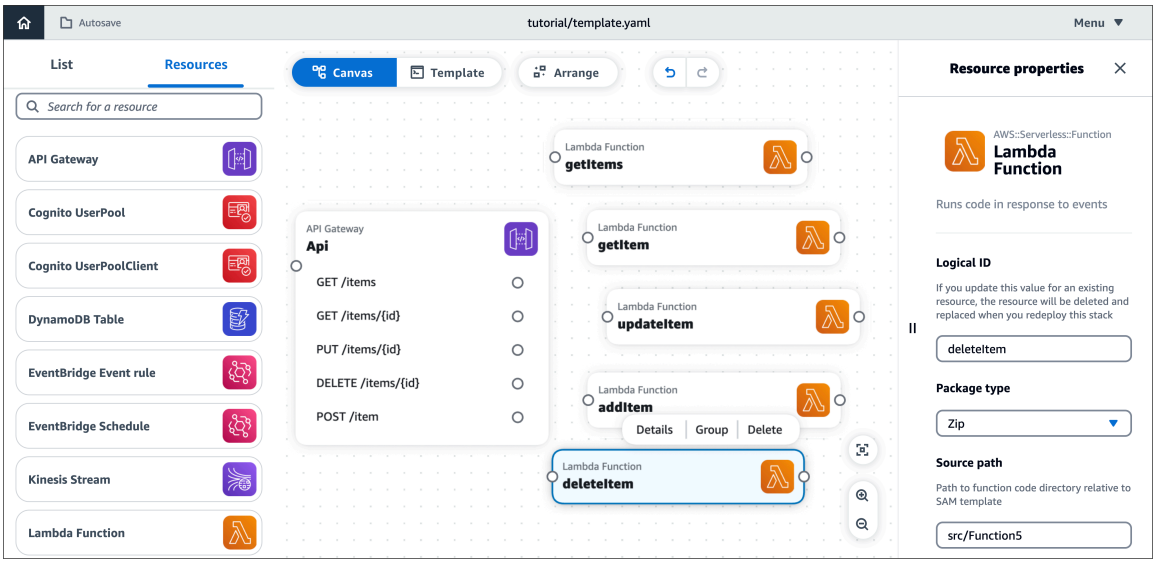


4단계: Lambda 함수 구성

리소스 속성 참조 테이블에 지정된 대로 5개의 Lambda 함수 각각에 이름을 지정합니다.

Lambda 함수의 이름을 지정하려면

1. Lambda 함수 카드의 리소스 속성 패널을 엽니다. 패널을 열려면 카드를 두 번 클릭합니다. 또는 카드를 선택한 다음 세부 정보를 선택합니다.
2. 리소스 속성 패널의 논리적 ID에 지정된 함수 이름을 입력합니다. 예를 들어 **`getItems`**입니다.
3. 저장을 선택합니다.
4. 5개 함수의 이름을 모두 지정할 때까지 이전 단계를 반복합니다.

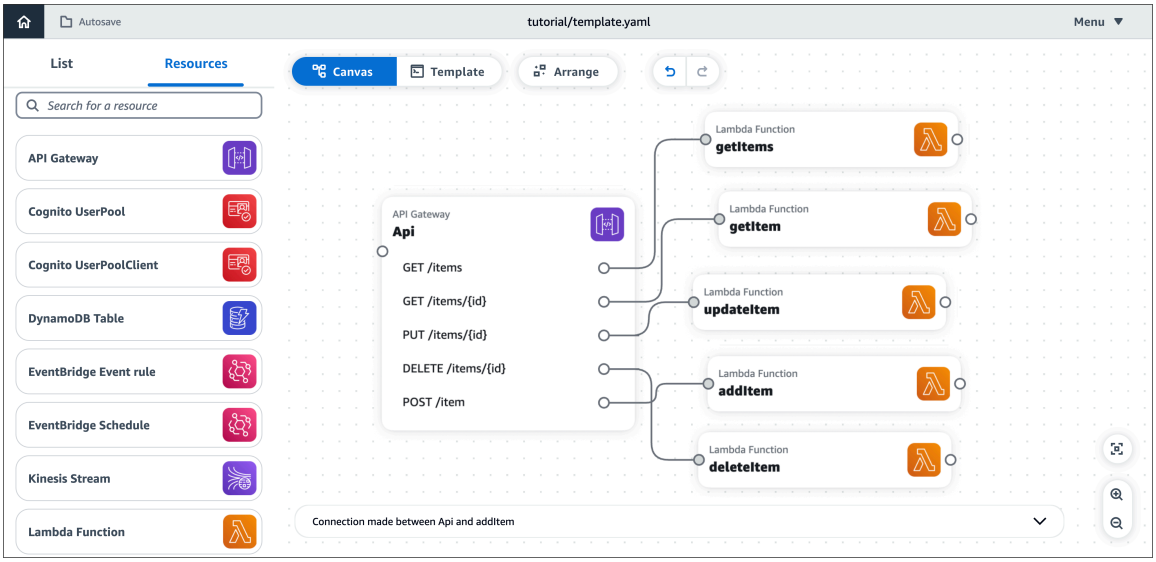


5단계: 카드 연결

리소스 속성 참조 표에 지정된 대로 API Gateway 카드의 각 경로를 관련 Lambda 함수 카드에 연결합니다.

카드를 연결하려면

1. API Gateway 카드에서 오른쪽 포트를 클릭하고 지정된 Lambda 함수 카드의 왼쪽 포트로 끕니다. 예를 들어 GET /items 포트를 클릭하고 getItems의 왼쪽 포트로 끕니다.
2. API Gateway 카드의 경로 5개를 모두 해당 Lambda 함수 카드에 연결할 때까지 이전 단계를 반복합니다.



6단계: 캔버스 구성

Lambda 함수를 그룹화하고 모든 카드를 정렬하여 시각적 캔버스를 구성합니다.

함수를 그룹화하려면

1. Shift를 길게 누른 다음 캔버스에서 각 Lambda 함수 카드를 선택합니다.
2. 그룹을 선택합니다.

그룹 이름을 지정하려면

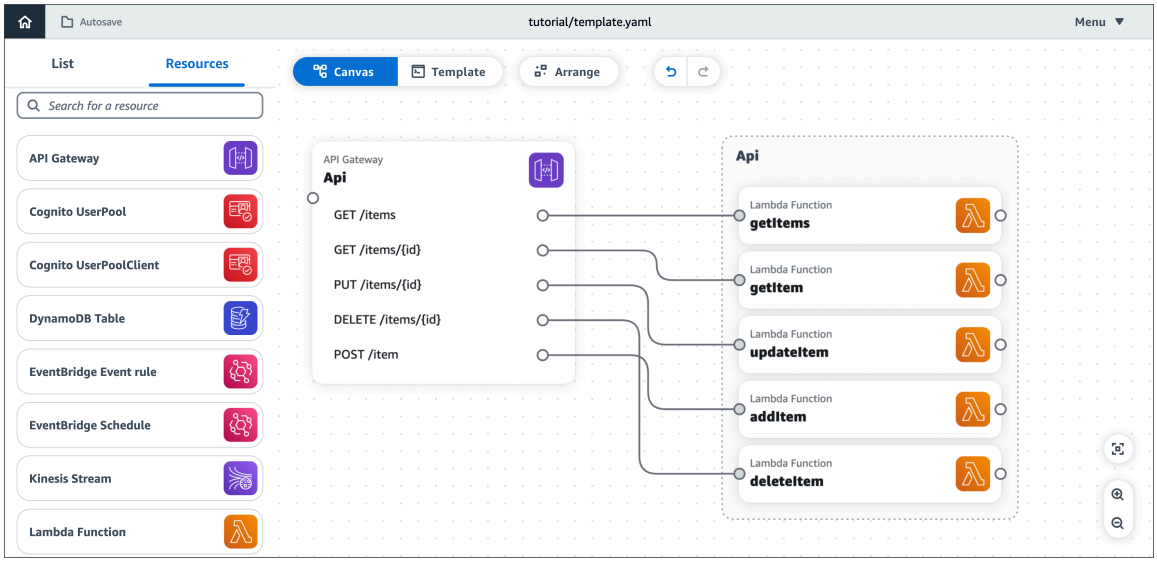
1. 그룹 이름(그룹) 근처의 그룹 상단을 두 번 클릭합니다.

   그룹 속성 패널이 열립니다.
2. 그룹 속성 패널의 그룹 이름에를 입력합니다**API**.
3. 저장을 선택합니다.

카드를 정렬하려면

캔버스의 기본 보기 영역 위에서 정렬을 선택합니다.

Infrastructure Composer는 다음과 같이 새 그룹(API)을 포함하여 시각적 캔버스의 모든 카드를 정렬합니다.

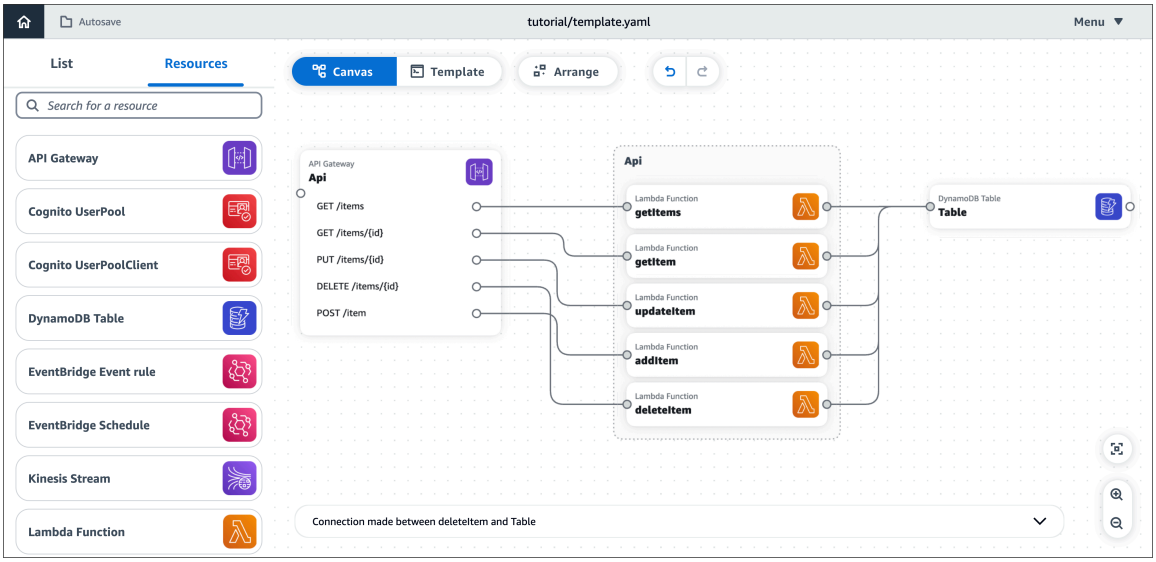


7단계: DynamoDB 테이블 추가 및 연결

이제 애플리케이션 아키텍처에 DynamoDB 테이블을 추가하고 Lambda 함수에 연결합니다.

DynamoDB 테이블을 추가하고 연결하려면

1. 리소스 팔레트(리소스)의 향상된 구성 요소 섹션에서 DynamoDB 테이블 카드를 캔버스로 드래그합니다.
2. Lambda 함수 카드에서 오른쪽 포트를 클릭하고 DynamoDB 테이블 카드의 왼쪽 포트로 끕니다.
3. 5개의 Lambda 함수 카드를 모두 DynamoDB 테이블 카드에 연결할 때까지 이전 단계를 반복합니다.
4. (선택 사항) 캔버스에서 카드를 재구성하고 재정렬하려면 정렬을 선택합니다.



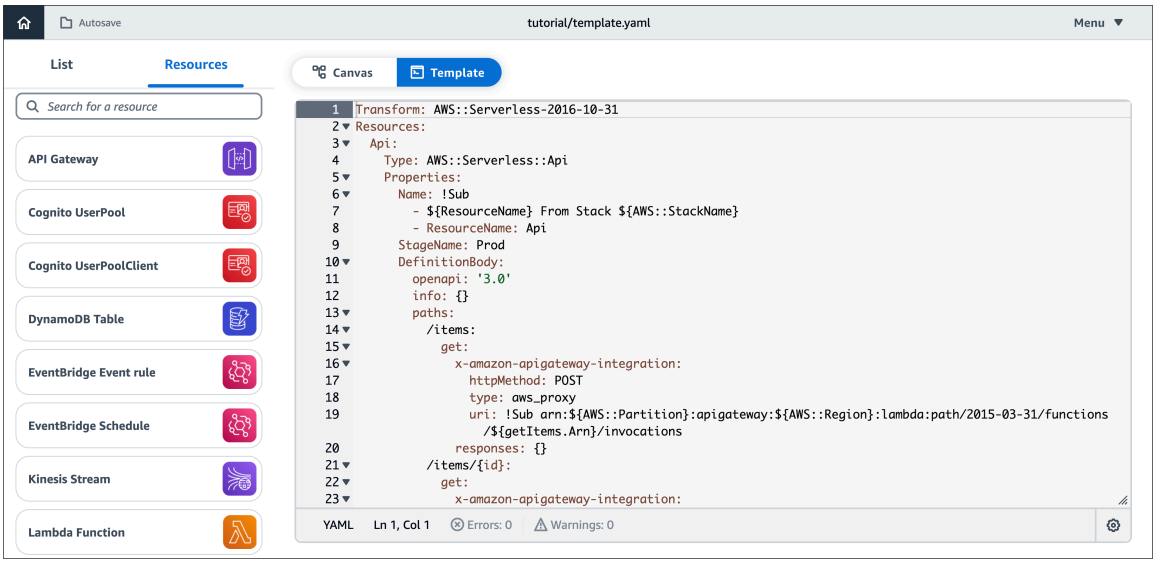
8단계: AWS CloudFormation 템플릿 검토

축하합니다! 배포 준비가 완료된 서버리스 애플리케이션을 성공적으로 설계했습니다. 마지막으로 템플릿을 선택하여 Infrastructure Composer가 자동으로 생성한 AWS CloudFormation 템플릿을 검토합니다.

템플릿에서 Infrastructure Composer는 다음을 정의했습니다.

- 템플릿을 (AWS SAM) 템플릿으로 지정하는 `Transform` 선언입니다 AWS Serverless Application Model . 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 AWS SAM 템플릿 구조를 참조하세요.

- 5개의 경로가 있는 API Gateway REST API를 지정하는 `AWS::Serverless::Api` 리소스입니다.
- 환경 변수 및 권한 정책을 포함하여 Lambda 함수의 구성을 지정하는 5개의 `AWS::Serverless::Function` 리소스입니다.
- DynamoDB 테이블과 해당 속성을 지정하는 `AWS::DynamoDB::Table` 리소스입니다.
- 리소스 그룹(API)에 대한 정보가 포함된 `Metadata` 섹션입니다. 이 섹션에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 메타데이터를 참조하세요.



9단계: 개발 워크플로에 통합

추가 테스트 및 배포를 위해 Infrastructure Composer가 생성한 템플릿 파일 및 프로젝트 디렉터리를 사용합니다.

- 로컬 동기화를 사용하면 Infrastructure Composer를 로컬 시스템의 IDE에 연결하여 개발 속도를 높일 수 있습니다. 자세한 내용은 Infrastructure Composer 콘솔을 로컬 IDE에 연결를 참조하세요.
- 로컬 동기화를 사용하면 로컬 시스템의 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)를 사용하여 애플리케이션을 테스트하고 배포할 수 있습니다. 자세한 내용은 Infrastructure Composer 서버리스 애플리케이션을 AWS 클라우드에 배포를 참조하세요.

다음 단계

이제 Infrastructure Composer를 사용하여 자체 애플리케이션을 빌드할 준비가 되었습니다. Infrastructure Composer 사용에 대한 자세한 내용은 섹션을 참조하세요에서를 작성하는 방법 AWS

Infrastructure Composer. 애플리케이션을 배포할 준비가 되면 섹션을 참조하세요Infrastructure Composer 서버리스 애플리케이션을 AWS 클라우드에 배포.

Infrastructure Composer를 사용할 수 있는 위치

콘솔, 소스 AWS Toolkit for Visual Studio Code및 CloudFormation 콘솔 모드의 Infrastructure Composer에서 Infrastructure Composer를 사용할 수 있습니다. 각각 약간 다른 사용 사례에 따라 다르지만 전반적으로 비슷한 경험입니다. 이 섹션에서는 각 경험에 대한 세부 정보를 제공합니다.

이 주제는 기본 콘솔 환경에 대한 포괄적인 개요AWS Infrastructure Composer 콘솔 사용입니다. 이 주제CloudFormation 콘솔 모드에서는 CloudFormation 스택 워크플로와 통합된 Infrastructure Composer 버전에 대한 세부 정보를 제공합니다. 에서는 VS Code에서 Infrastructure Composer에 액세스하고 사용하는 방법에 대한 정보를 AWS Toolkit for Visual Studio Code 제공합니다.

주제

- AWS Infrastructure Composer 콘솔 사용
- CloudFormation 콘솔 모드에서 Infrastructure Composer 사용
- 에서 Infrastructure Composer 사용 AWS Toolkit for Visual Studio Code

AWS Infrastructure Composer 콘솔 사용

이 섹션에서는 Infrastructure Composer 콘솔 AWS Infrastructure Composer 에서 액세스 및 사용에 대한 세부 정보를 제공합니다. 이는 Infrastructure Composer의 기본 환경이며 Infrastructure Composer에 익숙해지는 좋은 방법입니다. Infrastructure Composer 콘솔을 로컬 IDE와 통합할 수도 있습니다. 자세한 내용은 Infrastructure Composer 콘솔을 로컬 IDE에 연결을 참조하세요.

또한 VS Code의 AWS 도구 키트에서 Infrastructure Composer에 액세스할 수 있으며, 에서 사용하도록 특별히 설계된 Infrastructure Composer 모드를 사용할 수 CloudFormation 있습니다.

Infrastructure Composer 사용에 대한 일반 설명서는 섹션을 참조하세요작성 방법.

주제

- AWS Infrastructure Composer 콘솔 시각적 개요
- Infrastructure Composer 콘솔에서 프로젝트 관리
- Infrastructure Composer 콘솔을 로컬 IDE에 연결
- Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용
- Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장

- Lambda 콘솔에서 Infrastructure Composer로 함수 가져오기
- Infrastructure Composer의 시각적 캔버스 이미지 내보내기

AWS Infrastructure Composer 콘솔 시각적 개요

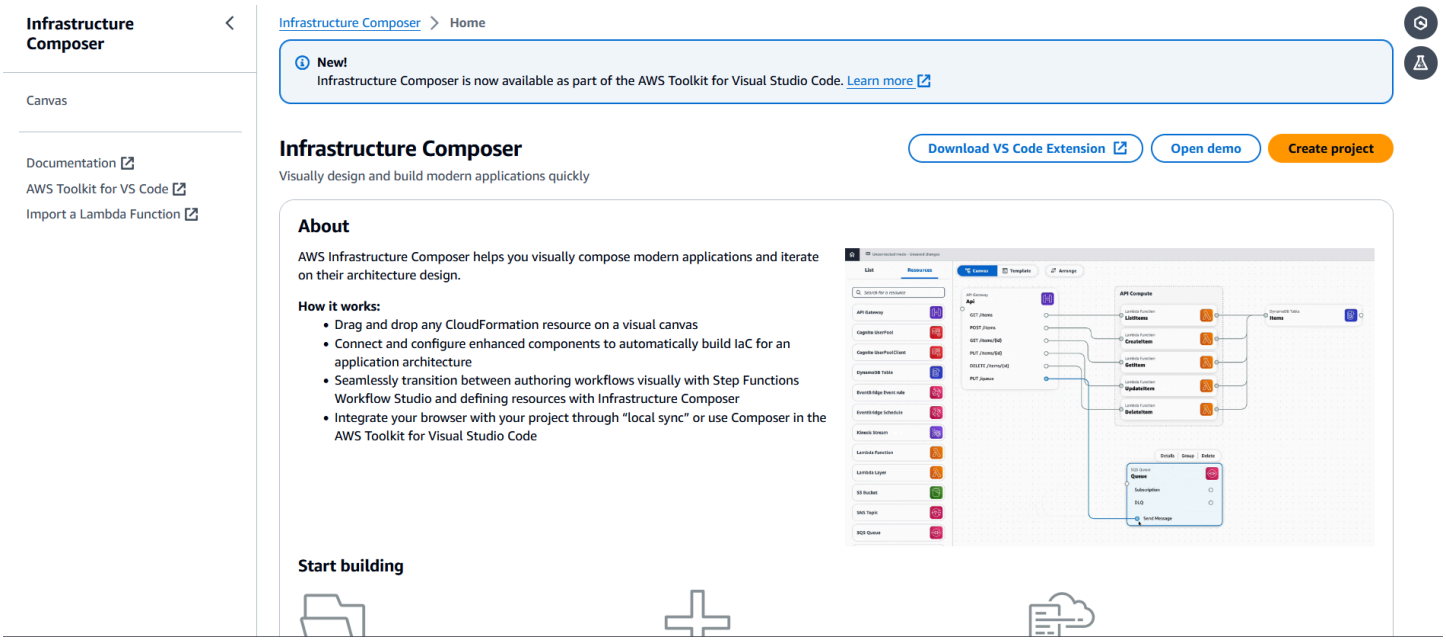
이 섹션에서는 AWS Infrastructure Composer 콘솔에 대한 시각적 개요를 제공합니다.

주제

- 홈 페이지
- 시각적 디자이너 및 시각적 캔버스

홈 페이지

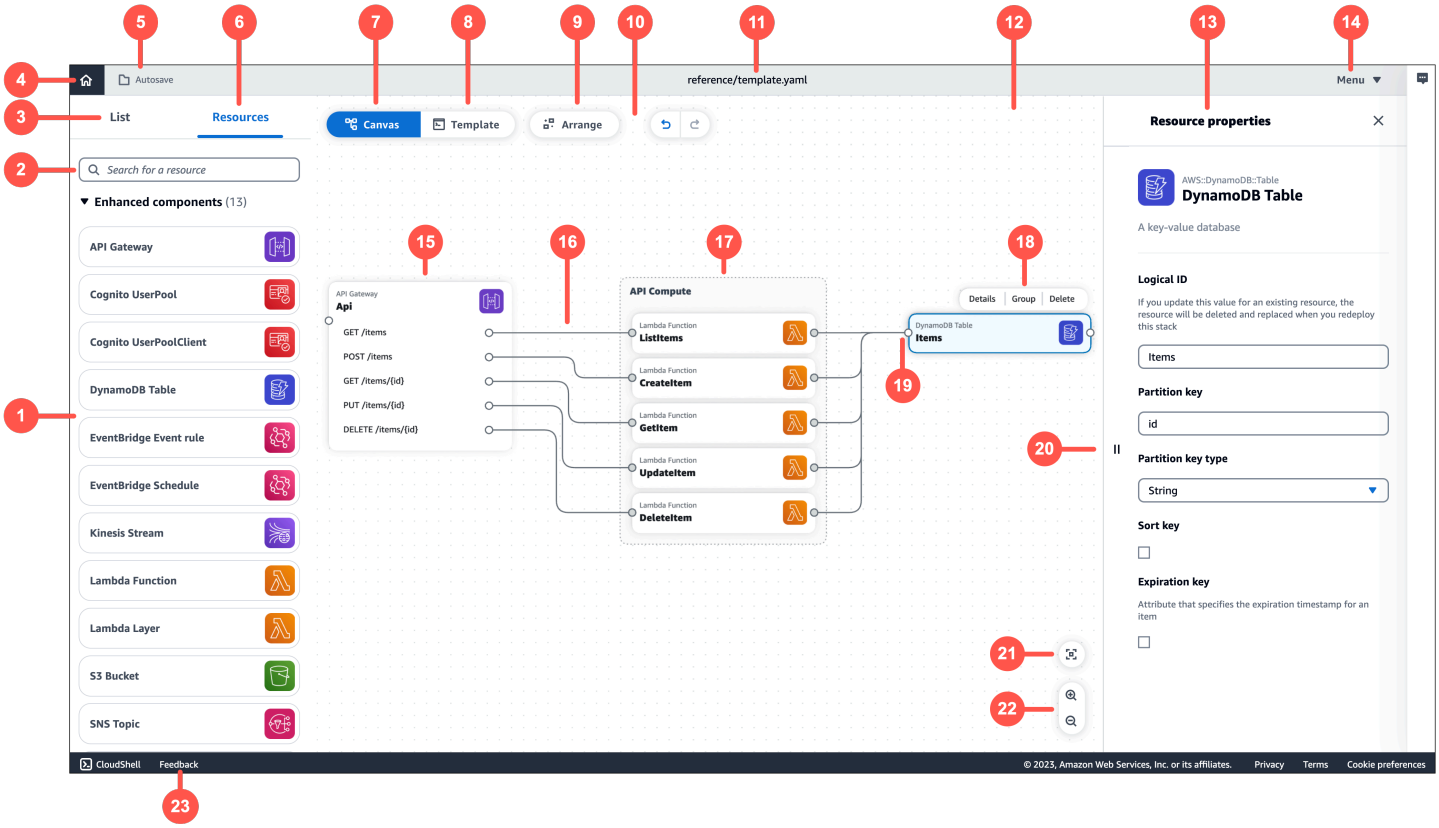
다음 이미지는 Infrastructure Composer 콘솔의 홈 페이지입니다.



1. 설명서 - Infrastructure Composer 설명서로 이동합니다.
2. Canvas - 캔버스로 이동하여 프로젝트를 생성하거나 로드합니다.
3. 데모 - Infrastructure Composer 데모 애플리케이션을 엽니다.
4. 프로젝트 생성 - 프로젝트를 생성하거나 로드합니다.
5. 빌드 시작 - 애플리케이션 빌드를 시작하는 빠른 링크입니다.
6. 피드백 - 여기로 이동하여 피드백을 제출합니다.

시각적 디자이너 및 시각적 캔버스

다음 이미지는 Infrastructure Composer의 시각적 디자이너와 시각적 캔버스입니다.



1. 리소스 팔레트 - 설계할 수 있는 카드를 표시합니다.
2. 리소스 검색 창 - 캔버스에 추가할 수 있는 카드를 검색합니다.
3. 목록 - 애플리케이션 리소스의 트리 보기를 표시합니다.
4. 홈 - Infrastructure Composer 홈페이지로 이동하려면 여기를 선택합니다.
5. 저장 상태 - Infrastructure Composer 변경 사항이 로컬 시스템에 저장되는지 여부를 나타냅니다. 상태는 다음을 포함합니다.
   - 자동 저장 - 로컬 동기화가 활성화되고 프로젝트가 자동으로 동기화되고 저장됩니다.
   - 변경 사항 저장됨 - 애플리케이션 템플릿이 로컬 시스템에 저장됩니다.
   - 저장되지 않은 변경 사항 - 애플리케이션 템플릿에 로컬 시스템에 저장되지 않은 변경 사항이 있습니다.
6. 리소스 - 리소스 팔레트를 표시합니다.
7. Canvas - 기본 보기 영역에 애플리케이션의 캔버스 보기를 표시합니다.
8. 템플릿 - 기본 보기 영역에 애플리케이션의 템플릿 보기를 표시합니다.

9. 정렬 - 캔버스에서 애플리케이션 아키텍처를 정렬합니다.

10.실행 취소 및 다시 실행 - 지원되는 경우 실행 취소 및 다시 실행 작업을 수행합니다.

11.템플릿 이름 - 설계하려는 템플릿의 이름을 나타냅니다.

12.기본 보기 영역 - 선택에 따라 캔버스 또는 템플릿을 표시합니다.

13.리소스 속성 패널 - 캔버스에서 선택한 카드의 관련 속성을 표시합니다. 이 패널은 동적입니다. 표시된 속성은 카드를 구성할 때 변경됩니다.

14.메뉴 - 다음과 같은 일반 옵션을 제공합니다.

- 프로젝트 만들기
- 템플릿 파일 또는 프로젝트 열기
- 템플릿 파일 저장
- 로컬 동기화 활성화
- 캔버스 내보내기
- 지원 받기
- 키보드 바로 가기

15.카드 - 캔버스에 카드 보기를 표시합니다.

16.라인 - 카드 간의 연결을 나타냅니다.

17.그룹 - 시각적 구성을 위해 선택한 카드를 그룹화합니다.

18.카드 작업 - 카드에 대해 수행할 수 있는 작업을 제공합니다.

a. 세부 정보 - 리소스 속성 패널을 가져옵니다.

b. 그룹 - 선택한 카드를 그룹화합니다.

c. 삭제 - 캔버스에서 카드를 삭제합니다.

19.포트 - 다른 카드를 가리키는 연결입니다.

20.리소스 속성 필드 - 카드에 대해 구성할 선별된 속성 필드 세트입니다.

21.리 센터링 - 애플리케이션 다이어그램을 시각적 캔버스의 리 센터링합니다.

22.확대 - 캔버스를 확대 및 축소합니다.

23.피드백 - 여기로 이동하여 피드백을 제출합니다.

Infrastructure Composer 콘솔에서 프로젝트 관리

이 주제에서는 Infrastructure Composer 콘솔에서 프로젝트를 관리하기 위해 수행하는 기본 작업에 대한 지침을 제공합니다. 여기에는 새 프로젝트 생성, 프로젝트 저장, 프로젝트 또는 템플릿 가져오기와

같은 일반적인 작업이 포함됩니다. 로컬 동기화 모드를 활성화하면 기존 프로젝트를 로드할 수도 있습니다. 로컬 동기화 모드를 활성화한 후 다음을 수행할 수 있습니다.

- 시작 템플릿과 폴더 구조로 구성된 새 프로젝트를 생성합니다.
- 프로젝트 템플릿과 파일이 포함된 상위 폴더를 선택하여 기존 프로젝트를 로드합니다.
- Infrastructure Composer를 사용하여 템플릿 및 폴더 관리

로컬 동기화 모드에서 Infrastructure Composer는 프로젝트의 템플릿과 폴더 변경 사항을 로컬 시스템에 자동으로 저장합니다. 브라우저가 로컬 동기화 모드를 지원하지 않거나 로컬 동기화 모드가 활성화되지 않은 상태에서 Infrastructure Composer를 사용하려는 경우 새 템플릿을 생성하거나 기존 템플릿을 로드할 수 있습니다. 변경 사항을 저장하려면 템플릿을 로컬 시스템으로 내보내야 합니다.

> **Note**
>
> Infrastructure Composer는 다음으로 구성된 애플리케이션을 지원합니다.
>
> - 인프라 코드를 정의하는 CloudFormation 또는 AWS Serverless Application Model 템플릿입니다.
> - Lambda 함수 코드, 구성 파일 및 빌드 폴더와 같은 프로젝트 파일을 구성하는 폴더 구조입니다.

주제

Infrastructure Composer 콘솔에서 새 프로젝트 생성

새 프로젝트를 생성하면 Infrastructure Composer가 시작 템플릿을 생성합니다. 캔버스에서 애플리케이션을 설계할 때 템플릿이 수정됩니다. 작업을 저장하려면 템플릿을 내보내거나 로컬 동기화 모드를 활성화해야 합니다.

새 프로젝트 생성

1. Infrastructure Composer 콘솔에 로그인합니다.

2. 홈 페이지에서 프로젝트 생성을 선택합니다.

> **Note**
>
> Infrastructure Composer에서 기존를 로드할 수도 있지만 먼저 로컬 동기화 모드를 활성화해야 합니다. 활성화되면 로컬 동기화가 활성화된 기존 Infrastructure Composer 프로젝트 로드를 참조하여 기존 프로젝트를 로드합니다.

Infrastructure Composer 콘솔에서 기존 프로젝트 폴더 가져오기

로컬 동기화 모드를 사용하여 기존 프로젝트의 상위 폴더를 가져올 수 있습니다. 프로젝트에 여러 템플릿이 포함된 경우 로드할 템플릿을 선택할 수 있습니다.

홈 페이지에서 기존 프로젝트를 가져오려면

1. Infrastructure Composer 콘솔에 로그인합니다.
2. 홈 페이지에서 CloudFormation 템플릿 로드를 선택합니다.
3. 프로젝트 위치에서 폴더 선택을 선택합니다. 프로젝트의 상위 폴더를 선택하고 선택을 선택합니다.

   > **Note**
   >
   > 이 프롬프트가 표시되지 않으면 브라우저가 로컬 동기화 모드에 필요한 파일 시스템 액세스 API를 지원하지 않을 수 있습니다. 자세한 내용은 Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용 단원을 참조하십시오.

4. 브라우저에서 메시지가 표시되면 파일 보기를 선택합니다.
5. 템플릿 파일의 경우 드롭다운 목록에서 템플릿을 선택합니다. 프로젝트에 단일 템플릿이 포함된 경우 Infrastructure Composer가 자동으로 해당 템플릿을 선택합니다.
6. 생성(Create)을 선택합니다.

캔버스에서 기존 프로젝트를 가져오려면

1. 캔버스에서 메뉴를 선택하여 메뉴를 엽니다.
2. 열기 섹션에서 프로젝트 폴더를 선택합니다.

> **Note**
>
> 프로젝트 폴더 옵션을 사용할 수 없는 경우 브라우저가 로컬 동기화 모드에 필요한 파일 시스템 액세스 API를 지원하지 않을 수 있습니다. 자세한 내용은 Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용 단원을 참조하십시오.

3. 프로젝트 위치에서 폴더 선택을 선택합니다. 프로젝트의 상위 폴더를 선택하고 선택을 선택합니다.
4. 브라우저에서 메시지가 표시되면 파일 보기를 선택합니다.
5. 템플릿 파일의 경우 드롭다운 목록에서 템플릿을 선택합니다. 프로젝트에 단일 템플릿이 포함된 경우 Infrastructure Composer가 자동으로 해당 템플릿을 선택합니다.
6. 생성(Create)을 선택합니다.

기존 프로젝트 폴더를 가져오면 Infrastructure Composer가 로컬 동기화 모드를 활성화합니다. 프로젝트의 템플릿 또는 파일에 대한 변경 사항은 로컬 시스템에 자동으로 저장됩니다.

Infrastructure Composer 콘솔에서 기존 프로젝트 템플릿 가져오기

기존 CloudFormation 또는 AWS SAM 템플릿을 가져오면 Infrastructure Composer는 캔버스에서 애플리케이션 아키텍처의 시각화를 자동으로 생성합니다.

로컬 시스템에서 프로젝트 템플릿을 가져올 수 있습니다.

기존 프로젝트 템플릿을 가져오려면

1. Infrastructure Composer 콘솔에 로그인합니다.
2. 프로젝트 생성을 선택하여 빈 캔버스를 엽니다.
3. 메뉴를 선택하여 메뉴를 엽니다.
4. 열기 섹션에서 템플릿 파일을 선택합니다.
5. 템플릿을 선택하고 열기를 선택합니다.

템플릿에 대한 변경 사항을 저장하려면 템플릿을 내보내거나 로컬 동기화 모드를 활성화해야 합니다.

Infrastructure Composer 콘솔에 기존 프로젝트 템플릿 저장

로컬 동기화 모드를 사용하지 않는 경우 템플릿을 내보내 변경 사항을 저장해야 합니다. 로컬 동기화 모드를 활성화한 경우 템플릿을 수동으로 저장할 필요가 없습니다. 변경 사항은 로컬 시스템에 자동으로 저장됩니다.

기존 프로젝트 템플릿을 저장하려면

1. Infrastructure Composer 캔버스에서 메뉴를 선택하여 메뉴를 엽니다.
2. 저장 섹션에서 템플릿 파일 저장을 선택합니다.
3. 템플릿의 이름을 입력합니다.
4. 템플릿을 저장할 위치를 선택합니다.
5. 저장을 선택합니다.

Infrastructure Composer 콘솔을 로컬 IDE에 연결

Infrastructure Composer 콘솔을 로컬 통합 개발 환경(IDE)에 연결하려면 로컬 동기화 모드를 사용합니다. 이 모드는 데이터를 로컬 시스템에 자동으로 동기화하고 저장합니다. 로컬 동기화 모드에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장. 로컬 동기화 모드 사용에 대한 지침은 섹션을 참조하세요Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장.

> **Note**
>
> 일부 브라우저에서는 로컬 동기화 활성화 옵션을 사용할 수 없습니다. Google Chrome 및 Microsoft Edge에서 사용할 수 있습니다.

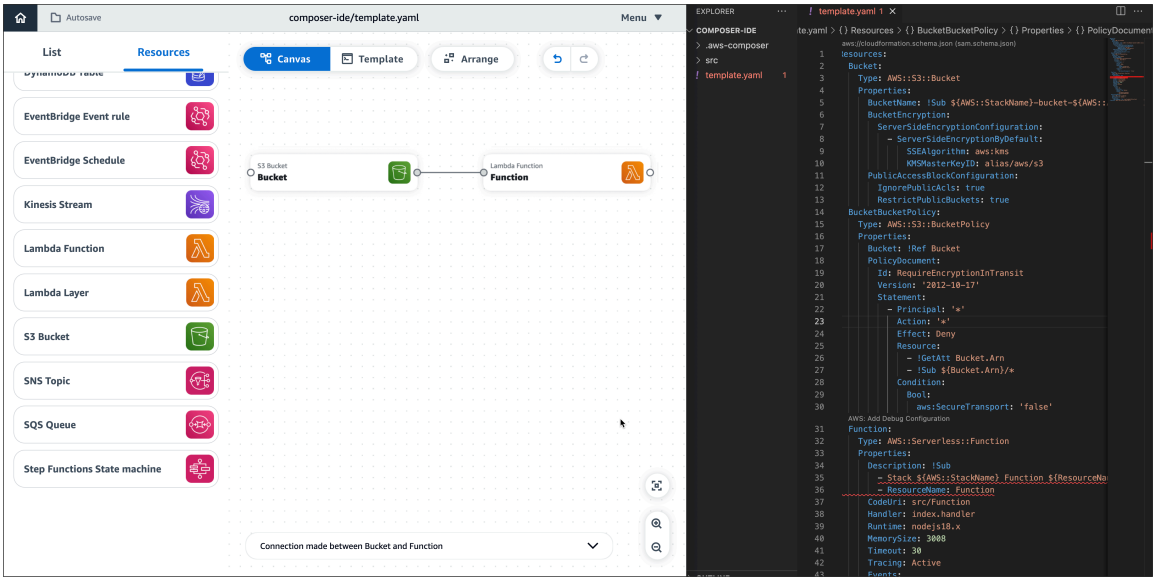
로컬 IDE에서 Infrastructure Composer 사용의 이점

Infrastructure Composer에서 설계하면 로컬 템플릿과 프로젝트 디렉터리가 자동으로 동기화되고 저장됩니다.

로컬 IDE를 사용하여 변경 사항을 보고 템플릿을 수정할 수 있습니다. 로컬에서 변경한 내용은 Infrastructure Composer에 자동으로 동기화됩니다.

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)와 같은 로컬 도구를 사용하여 애플리케이션을 빌드, 테스트, 배포하는 등의 작업을 수행할 수 있습니다. 다음 예제에서는 리소스를

Infrastructure Composer의 시각적 캔버스로 드래그 앤 드롭하여 로컬 IDE의 AWS SAM 템플릿에 마크업을 생성하는 방법을 보여줍니다.



Infrastructure Composer를 로컬 IDE와 통합

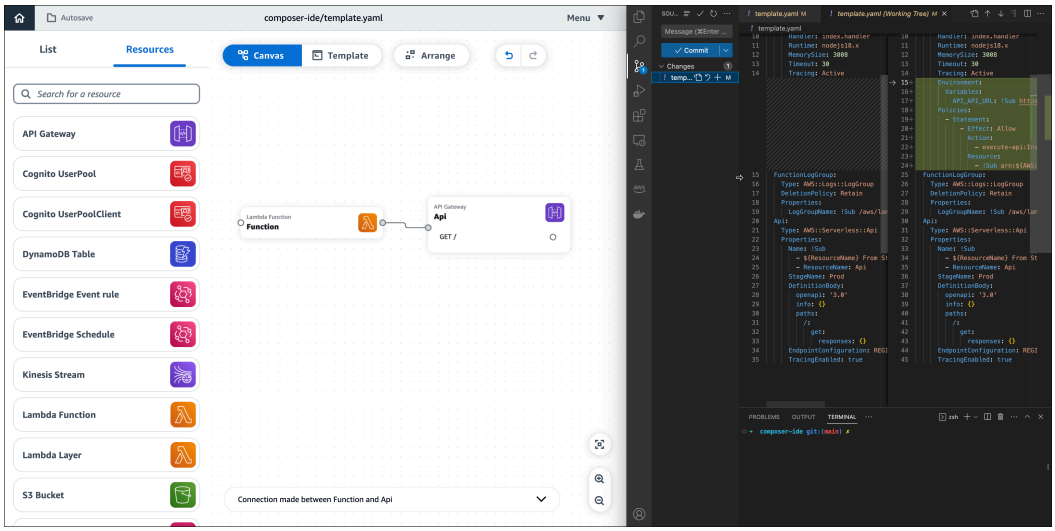
Infrastructure Composer를 로컬 IDE와 통합하려면

1. Infrastructure Composer에서 프로젝트를 생성 또는 로드하고 화면 오른쪽 상단의 메뉴 버튼을 선택하고 로컬 동기화 활성화를 선택하여 로컬 동기화를 활성화합니다.

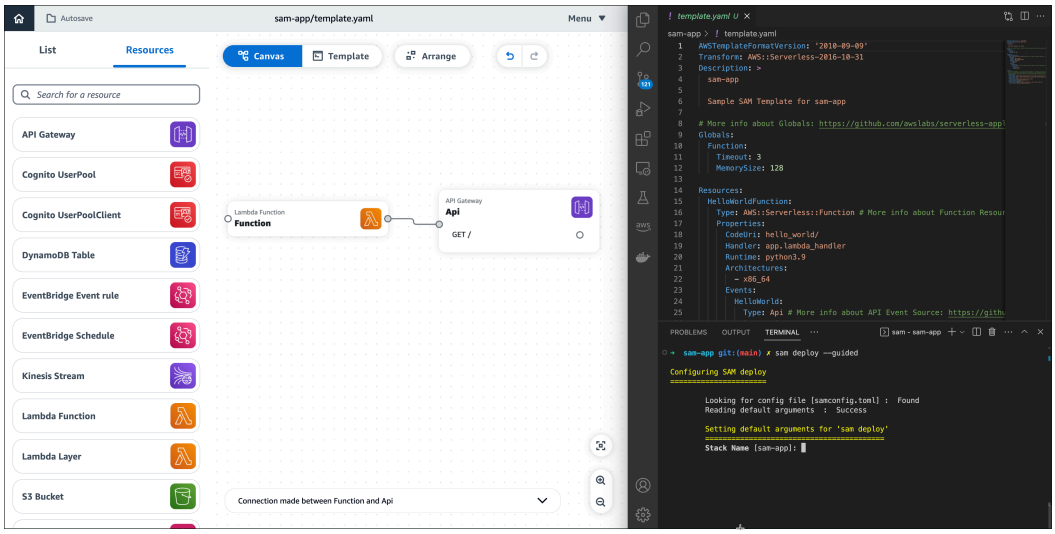
   > **Note**
   >
   > 일부 브라우저에서는 로컬 동기화 활성화 옵션을 사용할 수 없습니다. Google Chrome 및 Microsoft Edge에서 사용할 수 있습니다.

2. 로컬 IDE에서 Infrastructure Composer와 동일한 프로젝트 폴더를 엽니다.
3. 로컬 IDE와 함께 Infrastructure Composer를 사용합니다. Infrastructure Composer에서 수행한 업데이트는 로컬 시스템과 자동으로 동기화됩니다. 다음은 수행할 수 있는 작업의 몇 가지 예입니다.

   a. 선택한 버전 제어 시스템을 사용하여 Infrastructure Composer에서 수행하는 업데이트를 추적합니다.



b. AWS SAM CLI를 로컬에서 사용하여 애플리케이션을 빌드, 테스트, 배포하는 등의 작업을 수행할 수 있습니다. 자세한 내용은 Infrastructure Composer 서버리스 애플리케이션을 AWS 클라우드에 배포를 참조하세요.



Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용

Infrastructure Composer 콘솔은 로컬 동기화 모드와 Lambda 콘솔에서 함수 가져오기를 지원합니다. 이러한 기능을 사용하려면 파일 시스템 액세스 API를 지원하는 웹 브라우저가 필요합니다. 최신 버전의 Google Chrome 및 Microsoft Edge는 파일 시스템 액세스 API의 모든 기능을 지원하며 Infrastructure Composer에서 로컬 동기화 모드와 함께 사용할 수 있습니다.

파일 시스템 액세스 API를 사용하면 웹 페이지가 로컬 파일 시스템에 액세스하여 파일을 읽거나 쓰거나 저장할 수 있습니다. 이 기능은 기본적으로 꺼져 있으며 시각적 프롬프트를 통해 허용하려면 권한이 필요합니다. 이 액세스 권한이 부여되면 웹 페이지의 브라우저 세션 기간 동안 유지됩니다.

파일 시스템 액세스 API에 대한 자세한 내용은 다음을 참조하세요.

- mdn 웹 문서의 파일 시스템 액세스 API.
- 파일 시스템 액세스 API: web.dev 웹 사이트의 로컬 파일에 대한 액세스 간소화.

로컬 동기화 모드

로컬 동기화 모드를 사용하면 Infrastructure Composer에서 설계할 때 템플릿 파일과 프로젝트 폴더를 로컬로 자동으로 동기화하고 저장할 수 있습니다. 이 기능을 사용하려면 파일 시스템 액세스 API를 지원하는 웹 브라우저가 필요합니다.

Data Infrastructure Composer는에 액세스할 수 있습니다.

Infrastructure Composer는 허용된 프로젝트 폴더와 해당 프로젝트 폴더의 하위 폴더에 대한 읽기 및 쓰기 액세스 권한을 얻습니다. 이 액세스는 설계 시 생성된 템플릿 파일, 프로젝트 폴더 및 백업 디렉터리를 생성, 업데이트 및 저장하는 데 사용됩니다. Infrastructure Composer에서 액세스하는 데이터는 다른 용도로 사용되지 않으며 로컬 파일 시스템 외부에 저장되지 않습니다.

민감한 데이터에 대한 액세스

파일 시스템 액세스 API는 민감한 데이터가 포함될 수 있는 특정 디렉터리에 대한 액세스를 제외하거나 제한합니다. Infrastructure Composer 로컬 동기화 모드와 함께 사용할 이러한 디렉터리 중 하나를 선택하면 오류가 발생합니다. 연결할 다른 로컬 디렉터리를 선택하거나 로컬 동기화가 비활성화된 상태에서 Infrastructure Composer를 기본 모드로 사용할 수 있습니다.

민감한 디렉터리의 예를 비롯한 자세한 내용은 파일 시스템 W3C 액세스 W3C 초안 커뮤니티 그룹 보고서에서 의도한 것보다 더 많은 또는 더 민감한 파일에 대한 액세스 권한을 부여하는 사용자를 참조하세요.

를 사용하는 경우 파일 시스템 액세스 APIWindows Subsystem for Linux (WSL)는 Windows 시스템 내 위치 때문에 전체 Linux 디렉터리에 대한 액세스를 제외합니다. 로컬 동기화가 비활성화된 Infrastructure Composer를 사용하거나 WSL 디렉터리의 프로젝트 파일을의 작업 디렉터리로 동기화하는 솔루션을 구성할 수 있습니다Windows. 그런 다음 Windows 디렉터리와 함께 Infrastructure Composer 로컬 동기화 모드를 사용합니다.

Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장

이 섹션에서는 Infrastructure Composer의 로컬 동기화 모드를 사용하여 프로젝트를 자동으로 동기화하고 로컬 시스템에 저장하는 방법에 대한 정보를 제공합니다.

다음과 같은 이유로 로컬 동기화를 사용하는 것이 좋습니다.

새 프로젝트에 대해 로컬 동기화를 활성화하거나 로컬 동기화가 활성화된 기존 프로젝트를 로드할 수 있습니다.

- 기본적으로 설계 시 애플리케이션 템플릿을 수동으로 저장해야 합니다. 로컬 동기화를 사용하여 변경 시 애플리케이션 템플릿을 로컬 시스템에 자동으로 저장합니다.
- 로컬 동기화는 프로젝트 폴더, 백업 폴더 및 지원되는 외부 파일을 관리하고 로컬 시스템에 자동으로 동기화합니다.
- 로컬 동기화를 사용하는 경우 Infrastructure Composer를 로컬 IDE와 연결하여 개발 속도를 높일 수 있습니다. 자세한 내용은 Infrastructure Composer 콘솔을 로컬 IDE에 연결를 참조하세요.

저장되는 로컬 동기화 모드

로컬 동기화 모드는 다음을 자동으로 동기화하고 로컬 시스템에 저장합니다.

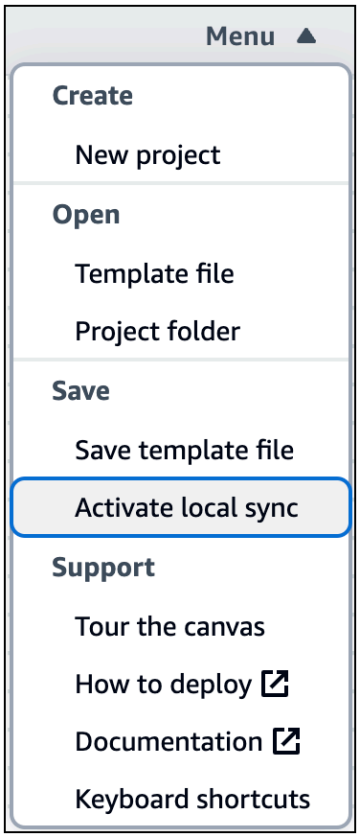
- 애플리케이션 템플릿 파일 - 코드형 인프라 AWS Serverless Application Model (IaC AWS SAM)가 포함된 AWS CloudFormation 또는 () 템플릿입니다.
- 프로젝트 폴더 - AWS Lambda 함수를 구성하는 일반 디렉터리 구조입니다.
- 백업 디렉터리 - 프로젝트 위치의 루트에서 `.aws-composer`생성된 라는 백업 디렉터리입니다. 이 디렉터리에는 애플리케이션 템플릿 파일 및 프로젝트 폴더의 백업 사본이 포함되어 있습니다.
- 외부 파일 - Infrastructure Composer 내에서 사용할 수 있는 지원되는 외부 파일입니다. 자세한 내용은 Infrastructure Composer에서 외부 파일 참조를 참조하세요.

브라우저 요구 사항

로컬 동기화 모드에는 파일 시스템 액세스 API를 지원하는 브라우저가 필요합니다. 자세한 내용은 Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용 단원을 참조하십시오.

로컬 동기화 모드 활성화

로컬 동기화 모드는 기본적으로 비활성화되어 있습니다. Infrastructure Composer 메뉴를 통해 로컬 동기화 모드를 활성화할 수 있습니다.



로컬 동기화 및 기존 로드 프로젝트를 활성화하는 방법에 대한 지침은 다음 주제를 참조하세요.

- Infrastructure Composer에서 로컬 동기화 활성화
- 로컬 동기화가 활성화된 기존 Infrastructure Composer 프로젝트 로드

Infrastructure Composer에서 로컬 동기화 활성화

로컬 동기화를 활성화하려면 다음 단계를 완료하세요.

1. Infrastructure Composer 홈 페이지에서 프로젝트 생성을 선택합니다.
2. Infrastructure Composer 메뉴에서 로컬 동기화 활성화를 선택합니다.
3. 프로젝트 위치에서 폴더 선택을 누르고 디렉터리를 선택합니다. 여기서 Infrastructure Composer는 설계에 따라 템플릿 파일과 폴더를 저장하고 동기화합니다.

   > **Note**
   >
   > 프로젝트 위치에는 기존 애플리케이션 템플릿이 포함되어서는 안 됩니다.

4. 액세스를 허용하라는 메시지가 표시되면 파일 보기를 선택합니다.

5. 활성화를 누릅니다. 변경 사항을 저장하라는 메시지가 표시되면 변경 사항 저장을 선택합니다.

   활성화하면 캔버스의 왼쪽 상단에 자동 저장 표시기가 표시됩니다.

로컬 동기화가 활성화된 기존 Infrastructure Composer 프로젝트 로드

로컬 동기화가 활성화된 기존 프로젝트를 로드하려면 다음 단계를 완료하세요.

1. Infrastructure Composer 홈 페이지에서 CloudFormation 템플릿 로드를 선택합니다.
2. Infrastructure Composer 메뉴에서 열기 > 프로젝트 폴더를 선택합니다.
3. 프로젝트 위치에서 폴더 선택을 누르고 프로젝트의 루트 폴더를 선택합니다.
4. 액세스를 허용하라는 메시지가 표시되면 파일 보기를 선택합니다.
5. 템플릿 파일에서 애플리케이션 템플릿을 선택하고 생성을 누릅니다.
6. 변경 사항을 저장하라는 메시지가 표시되면 변경 사항 저장을 선택합니다.

   활성화하면 캔버스의 왼쪽 상단에 자동 저장 표시기가 표시됩니다.

Lambda 콘솔에서 Infrastructure Composer로 함수 가져오기

Infrastructure Composer는 AWS Lambda 콘솔과의 통합을 제공합니다. Lambda 콘솔에서 Infrastructure Composer 콘솔로 Lambda 함수를 가져올 수 있습니다. 그런 다음 Infrastructure Composer 캔버스를 사용하여 애플리케이션 아키텍처를 추가로 설계합니다.

- 이 통합에는 파일 시스템 액세스 API를 지원하는 브라우저가 필요합니다. 자세한 내용은 Infrastructure Composer의 로컬 파일에 대한 웹 페이지 액세스 허용 단원을 참조하십시오.
- Lambda 함수를 Infrastructure Composer로 가져올 때 로컬 동기화 모드를 활성화하여 변경 사항을 저장해야 합니다. 자세한 내용은 Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장 단원을 참조하십시오.

이 통합 사용을 시작하려면 AWS Lambda 개발자 안내서의 AWS Lambda 에서 사용을 AWS Infrastructure Composer 참조하세요.

Infrastructure Composer의 시각적 캔버스 이미지 내보내기

이 주제에서는 AWS Infrastructure Composer 콘솔 내보내기 캔버스 기능에 대해 설명합니다.

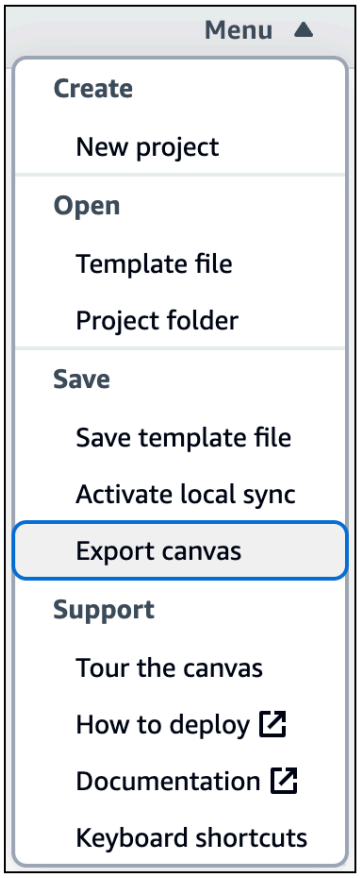
모든 Infrastructure Composer 기능에 대한 시각적 개요는 섹션을 참조하세요AWS Infrastructure Composer 콘솔 시각적 개요.

내보내기 캔버스 정보

캔버스 내보내기 기능은 애플리케이션의 캔버스를 로컬 시스템으로 이미지로 내보냅니다.

- Infrastructure Composer는 시각적 디자이너 UI 요소를 제거하고 애플리케이션의 다이어그램만 내보냅니다.
- 기본 이미지 파일 형식은 입니다png.
- 파일은 로컬 시스템의 기본 다운로드 위치로 내보내집니다.

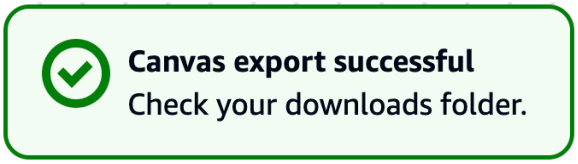
메뉴에서 캔버스 내보내기 기능에 액세스할 수 있습니다.



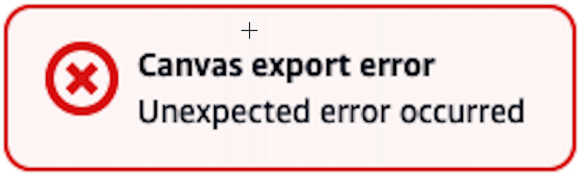
캔버스 내보내기

캔버스를 내보내면 Infrastructure Composer에 상태 메시지가 표시됩니다.

내보내기에 성공하면 다음 메시지가 표시됩니다.



내보내기에 실패한 경우 오류 메시지가 표시됩니다. 오류가 발생하면 내보내기를 다시 시도하세요.



CloudFormation 콘솔 모드에서 Infrastructure Composer 사용

CloudFormation 콘솔 모드의 Infrastructure Composer는 CloudFormation 템플릿을 시각화하는 데 권장되는 도구입니다. 이 도구를 사용하여 CloudFormation 템플릿을 생성하고 편집할 수도 있습니다.

이 모드는 Infrastructure Composer 콘솔과 어떻게 다른가요?

CloudFormation 콘솔 모드의 Infrastructure Composer는 일반적으로 기본 Infrastructure Composer 콘솔과 기능이 동일하지만 몇 가지 차이점이 있습니다.

- 이 모드는 CloudFormation 콘솔의 스택 워크플로와 통합됩니다. 이렇게 하면 Infrastructure Composer를에서 직접 사용할 수 있습니다 CloudFormation.
- Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장데이터를 로컬 시스템에 자동으로 동기화하고 저장하는 기능인는 지원되지 않습니다.
- Lambda 관련 카드(Lambda 함수 및 Lambda 계층)에는이 모드에서 사용할 수 없는 코드 빌드 및 패키징 솔루션이 필요합니다.

> **Note**
>
> 이러한 카드와 로컬 동기화는 Infrastructure Composer 콘솔 또는에서 사용할 수 있습니다 AWS Toolkit for Visual Studio Code.

CloudFormation 콘솔에서 Infrastructure Composer를 열면 Infrastructure Composer가 CloudFormation 콘솔 모드에서 열립니다. 이 모드에서는 Infrastructure Composer를 사용하여 템플릿을 시각화, 생성 및 업데이트할 수 있습니다.

CloudFormation 콘솔 모드에서 Infrastructure Composer에 액세스하는 방법

CloudFormation 콘솔 모드의 Infrastructure Composer는 CloudFormation Designer에서 업그레이드한 것입니다. Infrastructure Composer를 사용하여 CloudFormation 템플릿을 시각화하는 것이 좋습니다. 이 도구를 사용하여 CloudFormation 템플릿을 생성하고 편집할 수도 있습니다.

1. Cloudformation 콘솔로 이동하여 로그인합니다.
2. 왼쪽 탐색 메뉴에서 Infrastructure Composer를 선택합니다. 그러면 CloudFormation 콘솔 모드의 Infrastructure Composer로 이동합니다.

> **Note**
>
> CloudFormation 콘솔 모드에서 Infrastructure Composer를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요CloudFormation 콘솔 모드에서 Infrastructure Composer 사용.

CloudFormation 콘솔 모드에서 Infrastructure Composer의 배포 시각화

이 주제의 지침에 따라 배포된 CloudFormation 스택/인프라 컴포저 템플릿을 시각화합니다.

1. CloudFormation 콘솔로 이동하여 로그인합니다.
2. 편집할 스택을 선택합니다.
3. 템플릿 탭을 선택합니다.
4. Infrastructure Composer를 선택합니다.

Infrastructure Composer는 스택/템플릿을 시각화합니다. 여기에서도 변경할 수 있습니다.

CloudFormation 콘솔 모드에서 Infrastructure Composer에서 새 템플릿 생성

이 주제의 지침에 따라 새 템플릿을 생성합니다.

1. CloudFormation 콘솔로 이동하여 로그인합니다.
2. 왼쪽 탐색 메뉴에서 Infrastructure Composer를 선택합니다. 그러면 CloudFormation 콘솔 모드에서 Infrastructure Composer가 열립니다.
3. 리소스 팔레트에서 필요한 리소스(카드)를 드래그, 드롭, 구성 및 연결합니다.

> **Note**
>
> Infrastructure Composer 사용에 대한 작성 방법 자세한 내용은 섹션을 참조하고 Lambda 관련 카드(Lambda 함수 및 Lambda 계층)에는 CloudFormation 콘솔 모드에서 Infrastructure Composer에서 사용할 수 없는 코드 빌드 및 패키징 솔루션이 필요합니다. 이러한 카드는 Infrastructure Composer 콘솔 또는에서 사용할 수 있습니다 AWS Toolkit for Visual Studio Code. 이러한 도구 사용에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer를 사용할 수 있는 위치.

4. 카드를 두 번 클릭하여 리소스 속성 패널을 사용하여 카드 구성 방법을 지정합니다.
5. 카드를 연결하여 애플리케이션의 이벤트 기반 워크플로를 지정합니다.
6. 템플릿을 선택하여 인프라 코드를 보고 편집합니다. 변경 사항은 캔버스 보기와 자동으로 동기화됩니다.
7. 템플릿을 스택으로 내보낼 준비가 되면 템플릿 생성을 선택합니다.
8. 확인 및 CloudFormation으로 내보내기 버튼을 선택합니다. 그러면 템플릿을 성공적으로 가져왔음을 확인하는 메시지가 포함된 스택 생성 워크플로로 돌아갑니다.

> **Note**
>
> 리소스가 있는 템플릿만 내보낼 수 있습니다.

9. 스택 생성 워크플로에서 다음을 선택합니다.
10. 스택 이름을 입력하고 나열된 파라미터를 검토한 후 다음을 선택합니다.

> **Note**
>
> 스택 이름은 문자로 시작해야 하며 문자, 숫자, 대시만 포함해야 합니다.

11. 다음 정보를 제공한 후 다음을 선택합니다.

    - 스택과 연결된 태그
    - 스택 권한
    - 스택의 실패 옵션

> **Note**
>
> 스택 관리에 대한 지침은 CloudFormation 사용 설명서의 CloudFormation 모범 사례를 참조하세요.

12. 스택 세부 정보가 올바른지 확인하고 페이지 하단에서 승인을 확인한 다음 제출 버튼을 선택합니다.

CloudFormation 는 템플릿의 데이터를 기반으로 스택 생성을 시작합니다.

CloudFormation 콘솔 모드에서 Infrastructure Composer의 기존 스택 업데이트

이 주제의 지침에 따라 기존 CloudFormation 스택을 업데이트합니다.

> **Note**
>
> 파일을 로컬에 저장하는 경우를 사용하는 것이 좋습니다AWS Toolkit for Visual Studio Code.

1. CloudFormation 콘솔로 이동하여 로그인합니다.
2. 편집할 스택을 선택합니다.
3. 업데이트 버튼을 선택합니다. 이렇게 하면 스택 업데이트 마법사로 이동합니다.
4. 오른쪽에서 Infrastructure Composer에서 편집을 선택합니다.
5. Infrastructure Composer에서 편집 레이블이 지정된 아래 버튼을 선택합니다. 그러면 CloudFormation 콘솔 모드의 Infrastructure Composer로 이동합니다.
6. 여기서 리소스 팔레트에서 리소스(카드)를 드래그, 드롭, 구성 및 연결할 수 있습니다.

> **Note**
>
> Infrastructure Composer 사용에 대한 작성 방법 자세한 내용은 섹션을 참조하고 Lambda 관련 카드(Lambda 함수 및 Lambda 계층)에는 CloudFormation 콘솔 모드에서 Infrastructure Composer에서 사용할 수 없는 코드 빌드 및 패키징 솔루션이 필요합니다. 이러한 카드는 Infrastructure Composer 콘솔 또는에서 사용할 수 있습니다 AWS

Toolkit for Visual Studio Code. 이러한 도구 사용에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer를 사용할 수 있는 위치.

7. 변경 사항을 내보낼 준비가 되면 템플릿 업데이트를 CloudFormation선택합니다.
8. 확인을 선택하고 CloudFormation으로 계속 진행합니다. 그러면 템플릿을 성공적으로 가져왔음을 확인하는 메시지가 포함된 스택 업데이트 워크플로로 돌아갑니다.

> **Note**
>
> 리소스가 있는 템플릿만 내보낼 수 있습니다.

9. 스택 업데이트 워크플로에서 다음을 선택합니다.
10. 나열된 파라미터를 검토하고 다음을 선택합니다.
11. 다음 정보를 제공한 후 다음을 선택합니다.

    - 스택과 연결된 태그
    - 스택 권한
    - 스택의 실패 옵션

> **Note**
>
> 스택 관리에 대한 지침은 CloudFormation 사용 설명서의 CloudFormation 모범 사례를 참조하세요.

12. 스택 세부 정보가 올바른지 확인하고 페이지 하단에서 승인을 확인한 다음 제출 버튼을 선택합니다.

CloudFormation 는 템플릿에서 수행한 업데이트를 기반으로 스택 업데이트를 시작합니다.

에서 Infrastructure Composer 사용 AWS Toolkit for Visual Studio Code

이 섹션에서는 AWS Infrastructure Composer 에서를 사용하는 방법을 설명합니다AWS Toolkit for Visual Studio Code. 여기에는의 Infrastructure Composer에 대한 시각적 개요가 포함됩니다 AWS Toolkit for Visual Studio Code. 또한이 환경에 액세스하고 VS Code에서 AWS 클라우드로 프로젝트를

동기화하는 방법을 보여주는 지침도 포함되어 있습니다. 동기화하려면의 sam sync 명령을 사용합니다 AWS SAM CLI. 또한이 섹션에서는에서 Infrastructure Composer에 있는 Amazon Q 동안를 사용하는 방법에 대한 지침을 제공합니다 AWS Toolkit for Visual Studio Code.

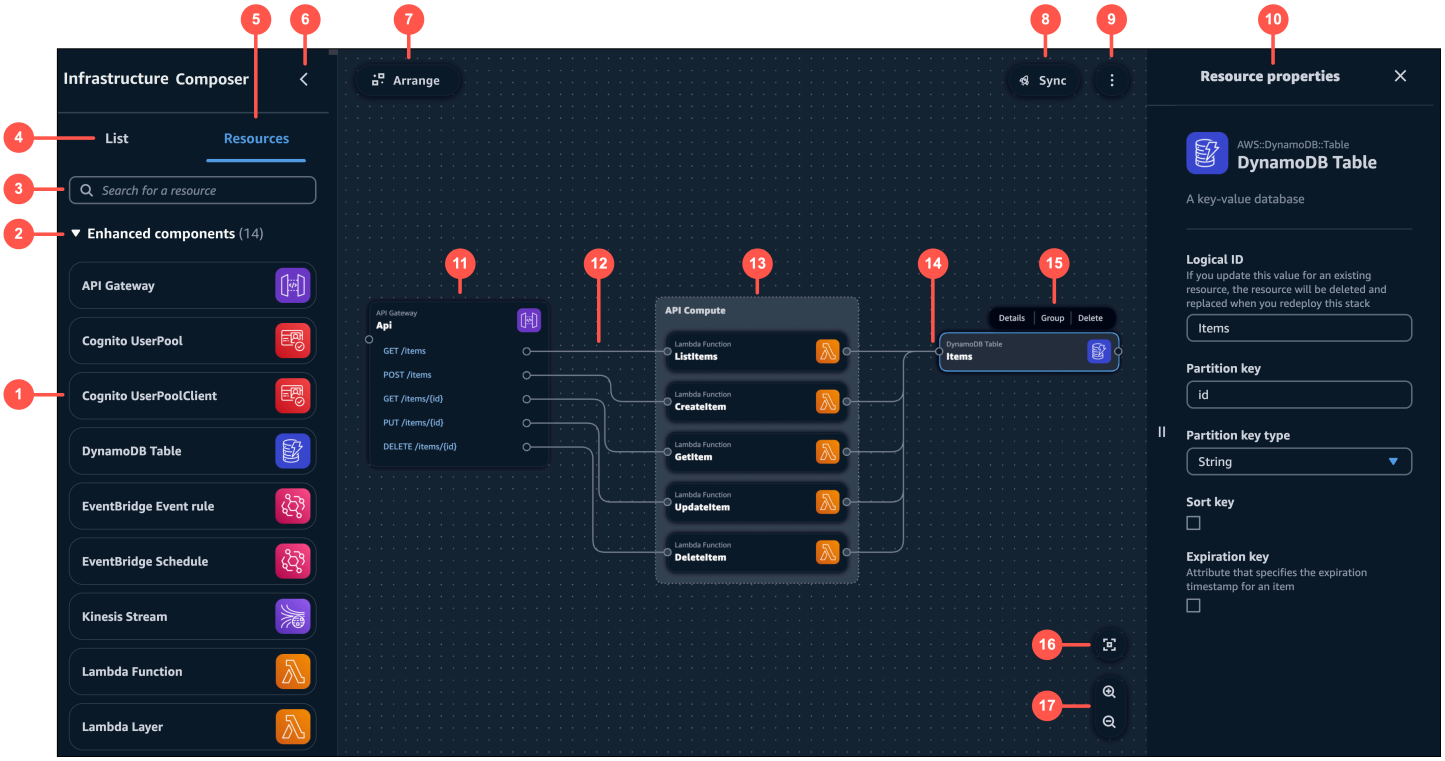
에서 Infrastructure Composer를 사용하는 방법에 대한 추가 지침은 섹션을 AWS Toolkit for Visual Studio Code참조하세요작성 방법. 이 섹션의 내용은이 환경과 Infrastructure Composer 콘솔 환경에 적용됩니다.

주제

- 의 Infrastructure Composer에 대한 시각적 개요 AWS Toolkit for Visual Studio Code
- 에서 Infrastructure Composer에 액세스 AWS Toolkit for Visual Studio Code
- Infrastructure Composer를 동기화하여에 배포 AWS 클라우드
- AWS Infrastructure Composer 와 함께 사용 Amazon Q Developer

의 Infrastructure Composer에 대한 시각적 개요 AWS Toolkit for Visual Studio Code

의 Infrastructure Composer의 시각적 디자이너 AWS Toolkit for Visual Studio Code 에는 다음 이미지에 번호가 지정되고 아래에 나열된 구성 요소가 포함된 시각적 캔버스가 포함되어 있습니다.



1. 리소스 팔레트 - 설계할 수 있는 카드를 표시합니다.
2. 카드 범주 - 카드는 Infrastructure Composer에 고유한 범주별로 구성됩니다.
3. 리소스 검색 창 - 캔버스에 추가할 수 있는 카드를 검색합니다.
4. 목록 - 애플리케이션 리소스의 트리 보기를 표시합니다.
5. 리소스 - 리소스 팔레트를 표시합니다.
6. 왼쪽 창 토글 - 왼쪽 창을 숨기거나 표시합니다.
7. 정렬 - 캔버스에서 애플리케이션 아키텍처를 정렬합니다.
8. 동기화 - AWS Serverless Application Model (AWS SAM) CLI `sam sync` 명령을 시작하여 애플리케이션을 배포합니다.
9. 메뉴 - 다음과 같은 일반 옵션을 제공합니다.
   - 캔버스 내보내기
   - 캔버스 둘러보기
   - 설명서 링크
   - 키보드 바로 가기
10. 리소스 속성 패널 - 캔버스에서 선택한 카드의 관련 속성을 표시합니다. 이 패널은 동적입니다. 표시된 속성은 카드를 구성할 때 변경됩니다.
11. 카드 - 캔버스에 카드 보기를 표시합니다.
12. 라인 - 카드 간의 연결을 나타냅니다.
13. 그룹 - 카드 그룹입니다. 시각적 구성에 대한 카드를 그룹화할 수 있습니다.
14. 포트 - 다른 카드를 가리키는 연결입니다.
15. 카드 작업 - 카드에 대해 수행할 수 있는 작업을 제공합니다.
    - 세부 정보 - 리소스 속성 패널을 가져옵니다.
    - 그룹 - 선택한 카드를 그룹화합니다.
    - 삭제 - 캔버스 및 템플릿에서 카드를 삭제합니다.
16. 리 센터링 - 애플리케이션 다이어그램을 시각적 캔버스의 리 센터링합니다.
17. 확대 - 캔버스를 확대 및 축소합니다.

에서 Infrastructure Composer에 액세스 AWS Toolkit for Visual Studio Code

이 주제의 지침에 따라에서 Infrastructure Composer에 액세스합니다 AWS Toolkit for Visual Studio Code.

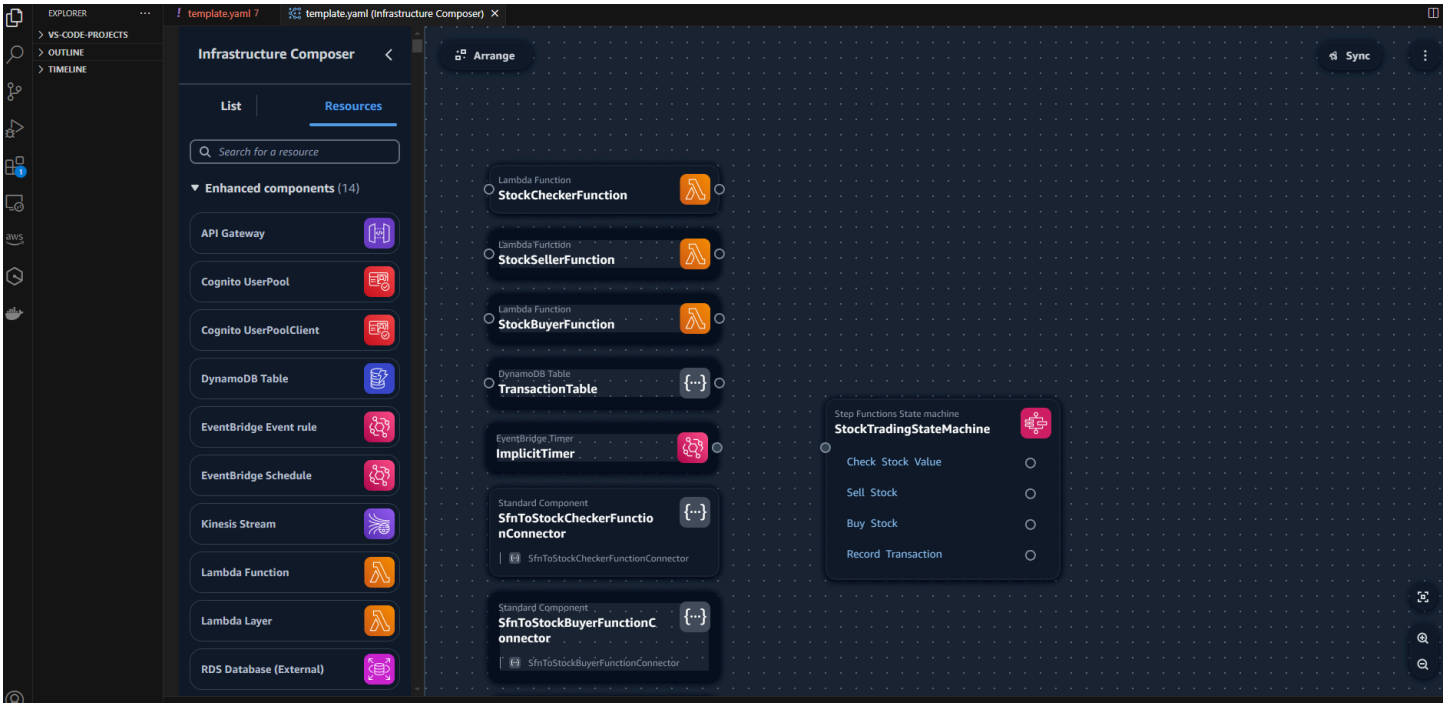
> **Note**
>
> 에서 Infrastructure Composer에 액세스하려면 먼저 Toolkit for VS Code를 다운로드하여 설치 AWS Toolkit for Visual Studio Code해야 합니다. 지침은 VS Code용 도구 키트 다운로드를 참조하세요.

Toolkit for VS Code에서 Infrastructure Composer에 액세스하려면

다음 방법 중 하나로 Infrastructure Composer에 액세스할 수 있습니다.

1. CloudFormation 또는 AWS SAM 템플릿에서 Infrastructure Composer 버튼을 선택합니다.
2. CloudFormation 또는 AWS SAM 템플릿을 마우스 오른쪽 버튼으로 클릭하여 컨텍스트 메뉴를 통해
3. VS 코드 명령 팔레트에서.

다음은 Infrastructure Composer 버튼에서 Infrastructure Composer에 액세스하는 예입니다.



Infrastructure Composer에 액세스하는 방법에 대한 자세한 내용은 도구 키트 AWS Infrastructure Composer 에서 액세스를 참조하세요.

Infrastructure Composer를 동기화하여에 배포 AWS 클라우드

AWS Infrastructure Composer 에서의 동기화 버튼을 사용하여 애플리케이션을에 AWS Toolkit for Visual Studio Code 배포합니다 AWS 클라우드.

동기화 버튼은 AWS SAM 명령줄 인터페이스()에서 `sam sync` 명령을 시작합니다CLI.

`sam sync` 명령은 새 애플리케이션을 배포하거나 로컬에서 변경한 내용을에 빠르게 동기화할 수 있습니다 AWS 클라우드. 실행에는 다음이 포함될 `sam sync` 수 있습니다.

- 로컬 `.aws-sam` 디렉터리를 생성하거나 업데이트하여 배포를 위한 로컬 애플리케이션 파일을 준비`sam build`하기 위해 로 애플리케이션을 빌드합니다.
- AWS 서비스 APIs 지원하는 리소스의 경우 AWS SAM CLI는 APIs를 사용하여 변경 사항을 배포합니다. 는 AWS SAM CLI 클라우드의 리소스를 빠르게 업데이트하기 위해이 작업을 수행합니다.
- 필요한 경우는 AWS SAM CLI AWS CloudFormation 배포를 수행하여 변경 세트를 통해 전체 스택을 업데이트합니다.

이 `sam sync` 명령은 클라우드 리소스를 빠르게 업데이트하면 개발 및 테스트 워크플로에 도움이 될 수 있는 빠른 개발 환경에 가장 적합합니다.

에 대한 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 sam sync 사용을 `sam sync`참조하세요.

설정

Infrastructure Composer에서 동기화 기능을 사용하려면 로컬 시스템에가 AWS SAM CLI 설치되어 있어야 합니다. 지침은 AWS Serverless Application Model 개발자 안내서의 설치를 AWS SAM CLI 참조하세요.

Infrastructure Composer에서 동기화 기능을 사용하면는 AWS SAM CLI 애플리케이션을에 동기화하는 데 필요한 정보에 대한 구성 파일을 참조합니다 AWS 클라우드. 구성 파일 생성, 수정 및 사용에 대한 지침은 AWS Serverless Application Model 개발자 안내서의 프로젝트 설정 구성을 참조하세요.

애플리케이션 동기화 및 배포

애플리케이션을에 동기화하려면 AWS 클라우드

1. Infrastructure Composer 캔버스에서 동기화 버튼을 선택합니다.

2. 개발 스택으로 작업하고 있음을 확인하는 메시지가 표시될 수 있습니다. 계속하려면 확인을 선택합니다.
3. Infrastructure Composer는 다음 옵션을 구성하라는 메시지를 표시할 수 있습니다.
   - AWS 리전 - 애플리케이션을 동기화할 리전입니다.
   - CloudFormation 스택 이름 - CloudFormation 스택의 이름입니다. 기존 스택 이름을 선택하거나 새 스택 이름을 생성할 수 있습니다.
   - Amazon Simple Storage Service(Amazon S3) 버킷 - Amazon S3 버킷의 이름입니다. 는 AWS SAM CLI 여기에 애플리케이션 파일과 함수 코드를 패키징하고 저장합니다. 기존 버킷을 선택하거나 새 버킷을 생성할 수 있습니다.

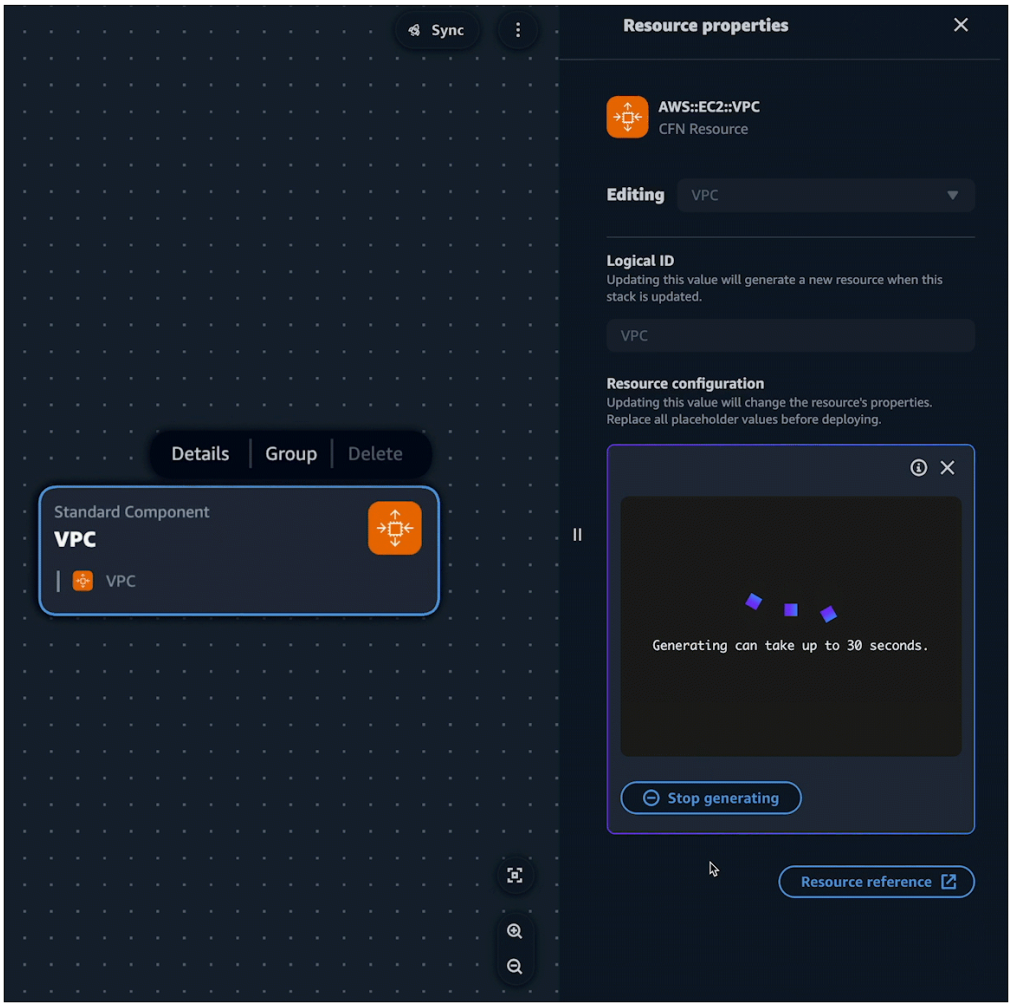
Infrastructure Composer는 `sam sync` 명령을 시작하고 AWS SAM CLI IDE에서 터미널 창을 열어 진행 상황을 출력합니다.

AWS Infrastructure Composer 와 함께 사용 Amazon Q Developer

AWS Infrastructure Composer 의는 와의 통합을 AWS Toolkit for Visual Studio Code 제공합니다 Amazon Q. Infrastructure Composer Amazon Q 내에서를 사용하여 애플리케이션을 설계할 때 AWS 리소스에 대한 인프라 코드를 생성할 수 있습니다.

Amazon Q는 범용 기계 학습 기반 코드 생성기입니다. 자세한 내용은 Amazon Q Developer 사용 설명서의 란 무엇입니까Amazon Q?를 참조하세요.

표준 리소스 및 표준 구성 요소 카드의 경우 Amazon Q를 사용하여 리소스에 대한 인프라 코드 제안을 생성할 수 있습니다.



표준 리소스 및 표준 구성 요소 카드는 CloudFormation 리소스 또는 리소스 모음을 나타낼 수 CloudFormation 있습니다. 자세한 내용은 Infrastructure Composer에서 카드 구성 및 수정를 참조하세요.

설정

Infrastructure ComposerAmazon Q에서를 사용하려면 도구 키트Amazon Q에서를 사용하여 인증해야 합니다. 지침은 Amazon Q Developer 사용 설명서Amazon Q의 VS Code 및 JetBrains에서 시작하기를 참조하세요.

Infrastructure ComposerAmazon Q Developer에서 사용

표준 리소스 또는 표준 구성 요소 카드의 리소스 속성 패널Amazon Q Developer에서를 사용할 수 있습니다.

Infrastructure ComposerAmazon Q에서를 사용하려면

1. 표준 리소스 또는 표준 구성 요소 카드에서 리소스 속성 패널을 엽니다.
2. 리소스 구성 필드를 찾습니다. 이 필드에는 카드의 인프라 코드가 포함되어 있습니다.
3. 제안 생성 버튼을 선택합니다. Amazon Q가 제안을 생성합니다.

> **Note**
>
> 이 단계에서 생성된 코드는 템플릿에서 기존 인프라 코드를 덮어쓰지 않습니다.

4. 더 많은 제안을 생성하려면 재생성을 선택합니다. 샘플을 전환하여 결과를 비교할 수 있습니다.
5. 옵션을 선택하려면 선택을 선택합니다. 코드를 애플리케이션에 저장하기 전에 여기에서 수정할 수 있습니다. 저장하지 않고 종료하려면 종료 아이콘(X)을 선택합니다.
6. 코드를 애플리케이션 템플릿에 저장하려면 리소스 속성 패널에서 저장을 선택합니다.

자세히 알아보기

Amazon Q에 대해 자세히 알아보려면 Amazon Q Developer 사용자 설명서의 Amazon Q이란?을 참조하세요.

에서를 작성하는 방법 AWS Infrastructure Composer

이 섹션에서는 Infrastructure Composer 콘솔, CloudFormation 콘솔 모드및에서 Infrastructure Composer를 사용하는 기본 사항을 다룹니다AWS Toolkit for Visual Studio Code. 보다 구체적으로,이 섹션의 주제에서는 Infrastructure Composer로 애플리케이션을 구성하는 방법에 대한 주요 세부 정보를 제공하고 추가 기능 및 바로 가기에 대한 세부 정보를 포함합니다. 콘솔과 VS Code 경험 간의 기능에는 몇 가지 변형이 있으며,이 섹션의 주제에서는 이러한 변형이 발생하는 위치를 식별하고 설명합니다.

애플리케이션을 구성한 후에는 애플리케이션 배포에 Infrastructure Composer 서버리스 애플리케이션을 AWS 클라우드에 배포 대한 정보를 검토할 준비가 됩니다.

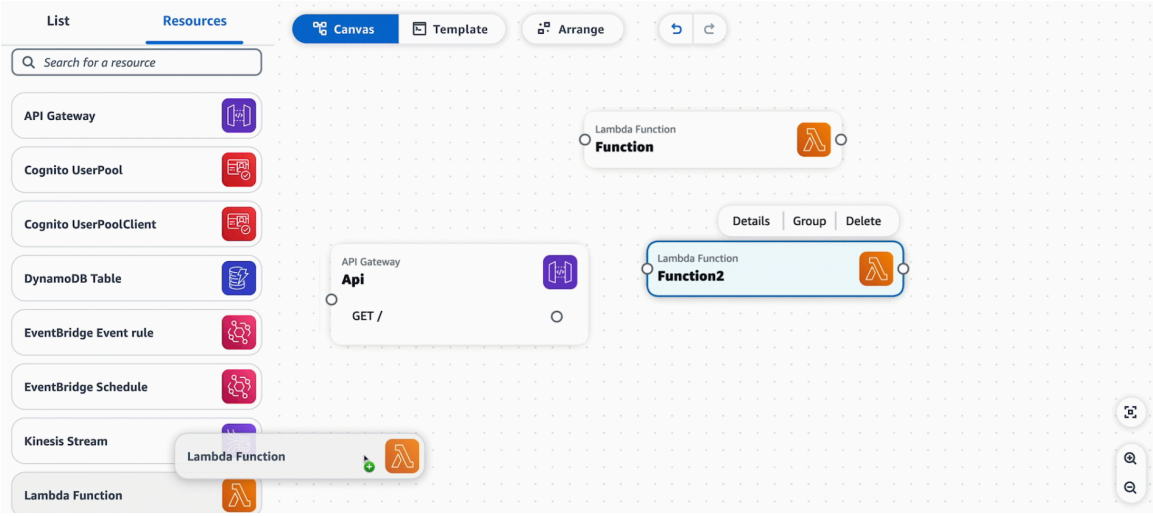
주제

- Infrastructure Composer의 시각적 캔버스에 카드 배치
- Infrastructure Composer의 시각적 캔버스에서 카드를 그룹화합니다.
- Infrastructure Composer의 시각적 캔버스에서 카드 연결
- Infrastructure Composer에서 카드 연결 해제
- Infrastructure Composer의 시각적 캔버스에 카드 정렬
- Infrastructure Composer에서 카드 구성 및 수정
- Infrastructure Composer에서 카드 삭제
- Infrastructure Composer에서 Change Inspector로 코드 업데이트 보기
- Infrastructure Composer에서 외부 파일 참조
- Infrastructure Composer를 Amazon Virtual Private Cloud(Amazon VPC)와 통합

Infrastructure Composer의 시각적 캔버스에 카드 배치

이 섹션에서는 시각적 캔버스에서 Infrastructure Composer 카드를 선택하고 드래그하는 방법을 설명합니다. 시작하기 전에 애플리케이션에 필요한 리소스와 이러한 리소스가 상호 작용하는 방법을 식별합니다. 이에 대한 팁은 단원을 참조하십시오Infrastructure Composer를 사용하여 첫 번째 애플리케이션 구축.

애플리케이션에 카드를 추가하려면 리소스 팔레트에서 카드를 끌어 시각적 캔버스에 놓습니다.



향상된 구성 요소 카드와 표준 IaC 리소스 카드라는 두 가지 유형의 카드 중에서 선택할 수 있습니다.

시각적 캔버스에 카드를 배치하면 카드를 그룹화, 연결, 정렬 및 구성할 준비가 됩니다. 이 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

- Infrastructure Composer의 시각적 캔버스에서 카드를 그룹화합니다.
- Infrastructure Composer의 시각적 캔버스에서 카드 연결
- Infrastructure Composer의 시각적 캔버스에 카드 정렬
- Infrastructure Composer에서 카드 구성 및 수정

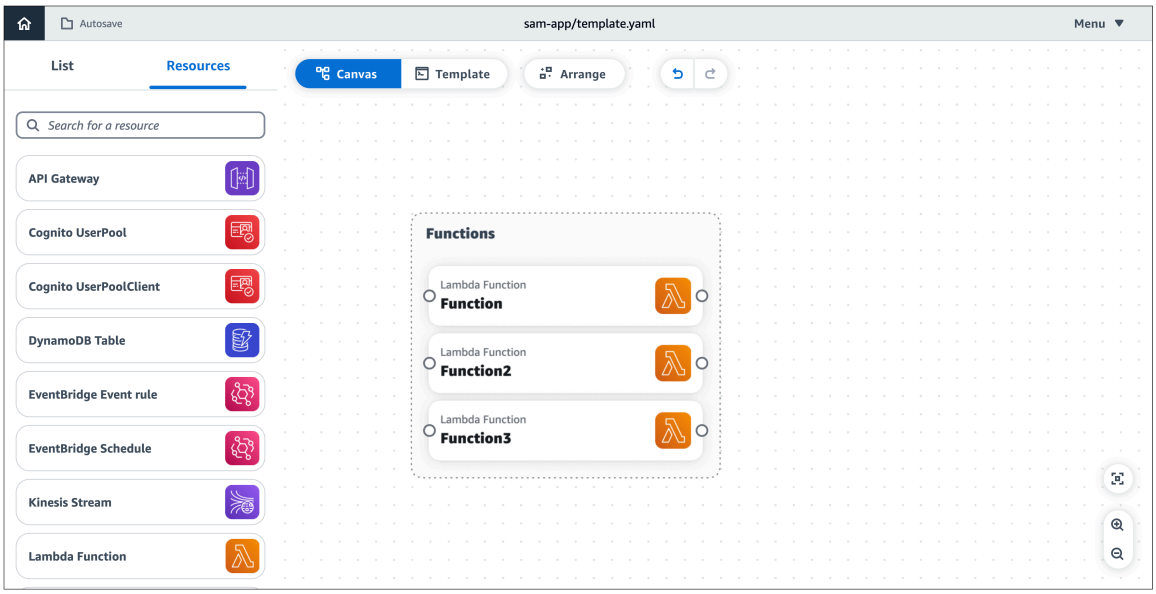
Infrastructure Composer의 시각적 캔버스에서 카드를 그룹화합니다.

이 주제에는 향상된 구성 요소 카드 및 표준 구성 요소 카드 그룹화에 대한 세부 정보가 포함되어 있습니다. 카드를 그룹화하면 작성해야 하는 코드나 마크업을 생각하지 않고도 리소스를 분류하고 구성할 수 있습니다.

향상된 구성 요소 카드 그룹화

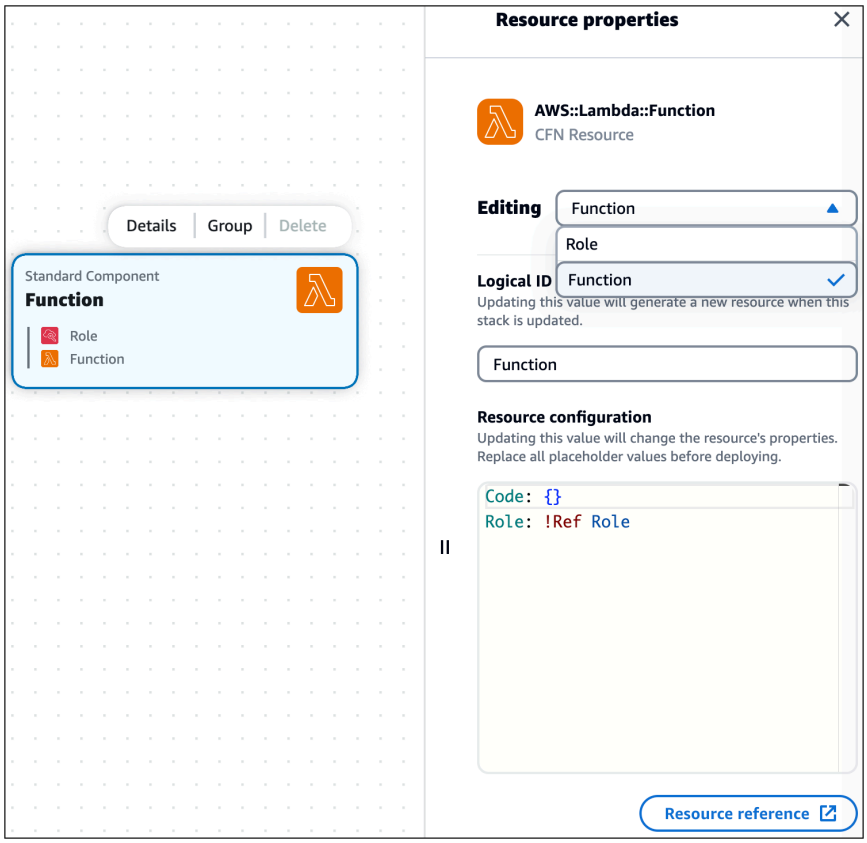
향상된 구성 요소 카드를 그룹화하는 방법에는 두 가지가 있습니다.

- Shift 키를 누른 상태에서 그룹화할 카드를 선택합니다. 그런 다음 리소스 작업 메뉴에서 그룹을 선택합니다.
- 그룹에서 원하는 카드를 선택합니다. 표시되는 메뉴에서 그룹을 선택합니다. 그러면 다른 카드를 끌어다 놓을 수 있는 그룹이 생성됩니다.



표준 구성 요소 카드를 다른 으로 그룹화

다음 예제는 리소스 속성 패널에서 표준 구성 요소 카드를 다른 카드로 그룹화할 수 있는 한 가지 방법을 보여줍니다.



리소스 속성 패널의 리소스 구성 필드에서 `Role`는 Lambda 함수에서 참조되었습니다. 그러면 역할 카드가 캔버스의 함수 카드로 그룹화됩니다.

Infrastructure Composer의 시각적 캔버스에서 카드 연결

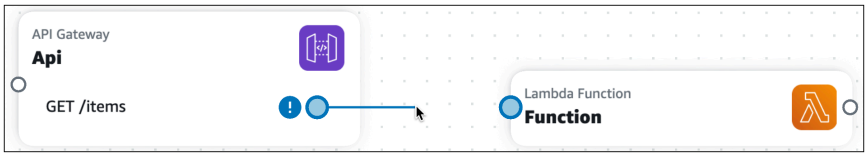
이 주제를 사용하여 Infrastructure Composer에서 카드를 연결하는 방법을 이해합니다. 이 섹션에는 향상된 구성 요소 카드와 표준 구성 요소 카드 연결에 대한 세부 정보가 포함되어 있습니다. 또한 카드를 연결할 수 있는 다양한 방법을 보여주는 몇 가지 예제도 제공합니다.

향상된 구성 요소 카드 연결

향상된 구성 요소 카드에서 포트는 연결할 수 있는 위치를 시각적으로 식별합니다.

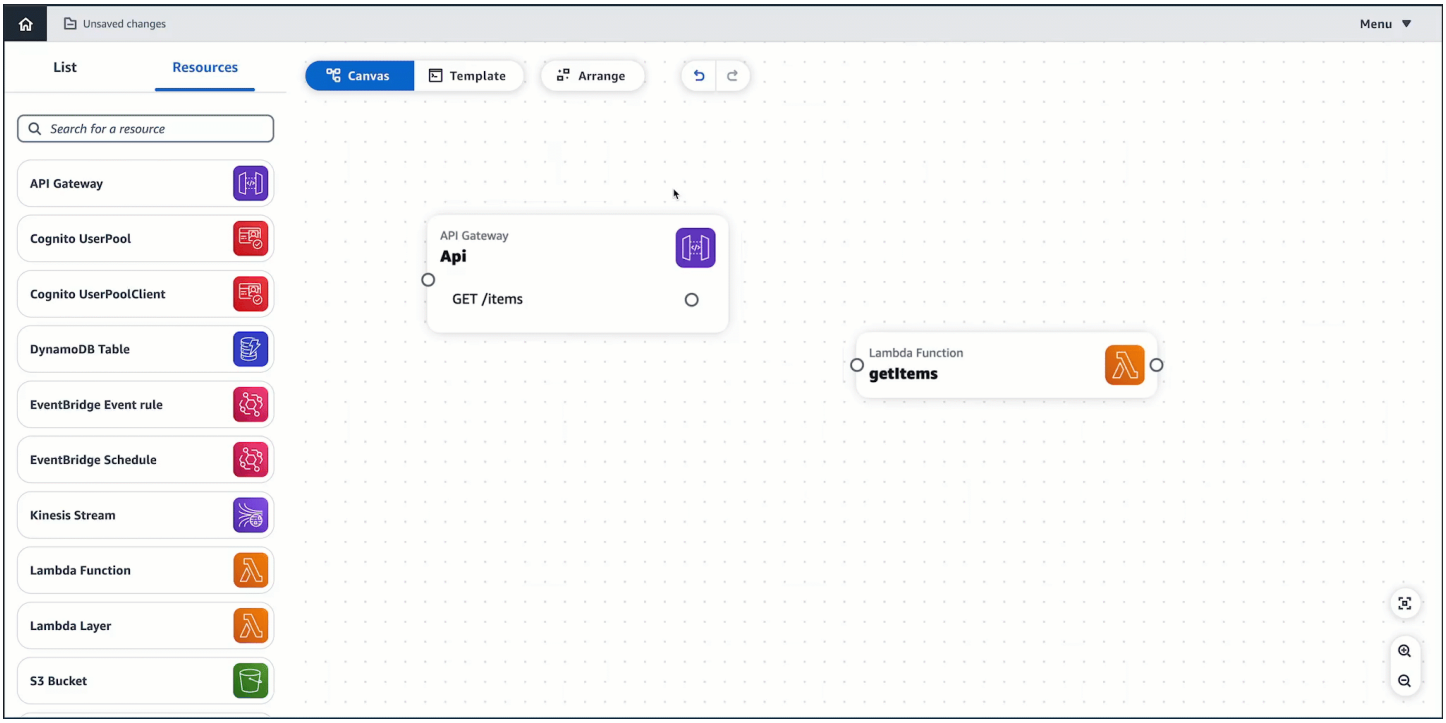
- 카드 오른쪽에 있는 포트는 카드가 다른 카드를 호출할 수 있는 기회를 나타냅니다.
- 카드 왼쪽의 포트는 다른 카드에서 카드를 호출할 수 있는 기회를 나타냅니다.

한 카드의 오른쪽 포트를 클릭하고 다른 카드의 왼쪽 포트로 끌어 카드를 연결합니다.



연결을 생성하면 연결 성공 여부를 알리는 메시지가 표시됩니다. 메시지를 선택하여 연결을 프로비저닝하기 위해 Infrastructure Composer가 변경한 내용을 확인합니다. 연결에 실패한 경우 템플릿 보기를 선택하여 인프라 코드를 수동으로 업데이트하여 연결을 프로비저닝할 수 있습니다.

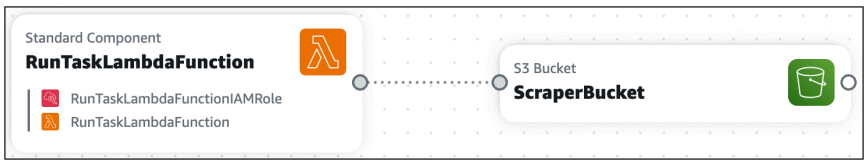
- 성공하면 메시지를 클릭하여 변경 검사기를 봅니다. 여기에서 연결을 프로비저닝하기 위해 Infrastructure Composer가 수정한 내용을 확인할 수 있습니다.
- 실패하면 메시지가 표시됩니다. 템플릿 보기를 선택하고 인프라 코드를 수동으로 업데이트하여 연결을 프로비저닝할 수 있습니다.



향상된 구성 요소 카드를 함께 연결하면 Infrastructure Composer는 템플릿에 인프라 코드를 자동으로 생성하여 리소스 간의 이벤트 기반 관계를 프로비저닝합니다.

표준 구성 요소 카드 연결(표준 IaC 리소스 카드)

표준 IaC 리소스 카드에는 다른 리소스와의 연결을 생성하는 포트가 포함되어 있지 않습니다. 카드 구성 중에 애플리케이션의 템플릿에서 이벤트 기반 관계를 지정하면 Infrastructure Composer는 이러한 연결을 자동으로 감지하고 카드 사이에 점선으로 시각화합니다. 다음은 표준 구성 요소 카드와 향상된 구성 요소 카드 간의 연결 예제입니다.



다음 예제에서는 Lambda 함수를 Amazon API Gateway Rest API와 연결하는 방법을 보여줍니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyApi:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: MyApi
```

```
  ApiGatewayMethod:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      HttpMethod: POST  # Specify the HTTP method you want to use (e.g., GET, POST,
 PUT, DELETE)
      ResourceId: !GetAtt MyApi.RootResourceId
      RestApiId: !Ref MyApi
      AuthorizationType: NONE
      Integration:
        Type: AWS_PROXY
        IntegrationHttpMethod: POST
        Uri: !Sub
          - arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${LambdaFunctionArn}/invocations
          - { LambdaFunctionArn: !GetAtt MyLambdaFunction.Arn }
      MethodResponses:
        - StatusCode: 200

  MyLambdaFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Runtime: nodejs14.x
      Code:
        S3Bucket: your-bucket-name
        S3Key: your-lambda-zip-file.zip

  LambdaExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: 'sts:AssumeRole'
      Policies:
        - PolicyName: LambdaExecutionPolicy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
```

```
                Action:
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                  - 'logs:PutLogEvents'
                Resource: 'arn:aws:logs:*:*:*'
              - Effect: Allow
                Action:
                  - 'lambda:InvokeFunction'
                Resource: !GetAtt MyLambdaFunction.Arn
```

위 예제에서 `ApiGatewayMethod:` 아래에 나열된 코드 조각은 두 카드를 연결하는 이벤트 기반 관계를 `Integration:` 지정합니다.

Infrastructure Composer에서 카드를 연결하는 예제

이 섹션의 예제를 사용하여 Infrastructure Composer에서 카드를 연결하는 방법을 이해합니다.

Amazon Simple Storage Service(Amazon S3) 버킷에 항목이 배치될 때 AWS Lambda 함수 호출

이 예제에서는 Amazon S3 버킷 카드가 Lambda 함수 카드에 연결됩니다. Amazon S3 버킷에 항목이 배치되면 함수가 호출됩니다. 그런 다음 함수를 사용하여 항목을 처리하거나 애플리케이션에서 다른 이벤트를 트리거할 수 있습니다.

이 상호 작용을 수행하려면 함수에 대한 이벤트를 정의해야 합니다. Infrastructure Composer는 다음과 같이 프로비저닝합니다.

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    ...
  MyBucketBucketPolicy:
    Type: AWS::S3::BucketPolicy
    ...
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
```

```
      ...
      Events:
        MyBucket:
          Type: S3
          Properties:
            Bucket: !Ref MyBucket
            Events:
              - s3:ObjectCreated:* # Event that triggers invocation of function
              - s3:ObjectRemoved:* # Event that triggers invocation of function
```

Lambda 함수에서 Amazon S3 버킷 호출

이 예제에서는 Lambda 함수 카드가 Amazon S3 버킷 카드를 호출합니다. Lambda 함수를 사용하여 Amazon S3 버킷의 항목에 대해 CRUD 작업을 수행할 수 있습니다.

이 상호 작용에는 Infrastructure Composer에서 프로비저닝한 다음이 필요합니다.

- Lambda 함수가 Amazon S3 버킷과 상호 작용하도록 허용하는 IAM 정책입니다.
- Lambda 함수의 동작에 영향을 미치는 환경 변수입니다.

```
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    ...
  MyBucketBucketPolicy:
    Type: AWS::S3::BucketPolicy
    ...
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      ...
      Environment:
        Variables:
          BUCKET_NAME: !Ref MyBucket
          BUCKET_ARN: !GetAtt MyBucket.Arn
      Policies:
        - S3CrudPolicy:
```

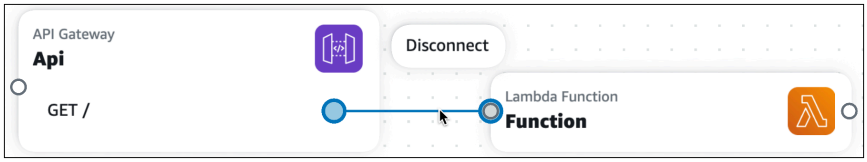
```
          BucketName: !Ref MyBucket
```

Infrastructure Composer에서 카드 연결 해제

Infrastructure Composer에서는 향상된 구성 요소 카드와 표준 구성 요소 카드를 사용하여 AWS 리소스를 연결하고 연결을 해제합니다. 이 섹션에서는 두 유형의 카드를 모두 연결 해제하는 방법을 설명합니다.

향상된 구성 요소 카드

향상된 구성 요소 카드를 연결 해제하려면 라인을 선택하고 연결 해제를 선택합니다.



Infrastructure Composer는 애플리케이션에서 이벤트 기반 관계를 제거하도록 템플릿을 자동으로 수정합니다.

표준 구성 요소 카드

표준 구성 요소 카드에는 다른 리소스와의 연결을 생성하는 포트가 포함되지 않습니다. 카드 구성 중에 애플리케이션의 템플릿에서 이벤트 기반 관계를 지정하면 Infrastructure Composer는 이러한 연결을 자동으로 감지하고 카드 사이에 점선으로 시각화합니다. 표준 구성 요소 카드의 연결을 해제하려면 애플리케이션의 템플릿에서 이벤트 기반 관계를 제거합니다.

다음 예제는 Amazon API Gateway Rest API와 연결된 Lambda 함수를 보여줍니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyApi:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: MyApi

  ApiGatewayMethod:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      HttpMethod: POST  # Specify the HTTP method you want to use (e.g., GET, POST,
 PUT, DELETE)
      ResourceId: !GetAtt MyApi.RootResourceId
```

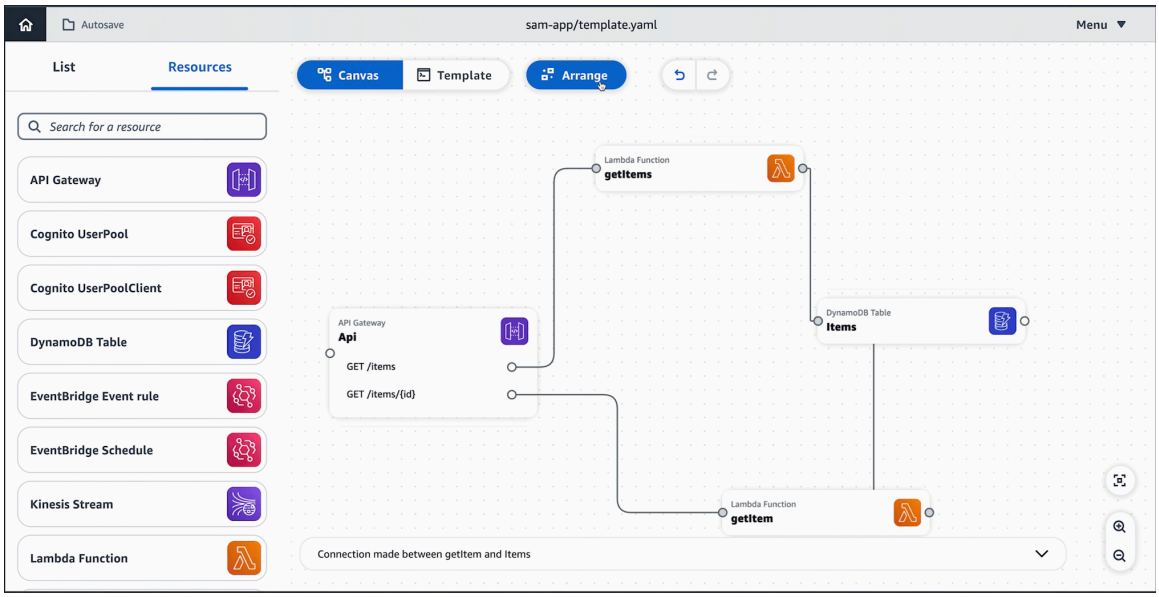
```
      RestApiId: !Ref MyApi
      AuthorizationType: NONE
      Integration:
        Type: AWS_PROXY
        IntegrationHttpMethod: POST
        Uri: !Sub
          - arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${LambdaFunctionArn}/invocations
          - { LambdaFunctionArn: !GetAtt MyLambdaFunction.Arn }
      MethodResponses:
        - StatusCode: 200

  MyLambdaFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Runtime: nodejs14.x
      Code:
        S3Bucket: your-bucket-name
        S3Key: your-lambda-zip-file.zip

  LambdaExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: 'sts:AssumeRole'
      Policies:
        - PolicyName: LambdaExecutionPolicy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                  - 'logs:PutLogEvents'
                Resource: 'arn:aws:logs:*:*:*'
              - Effect: Allow
```

```
                Action:
                  - 'lambda:InvokeFunction'
                Resource: !GetAtt MyLambdaFunction.Arn
```

두 카드 간의 연결을 제거하려면 `ApiGatewayMethod:` 아래의에 `MyLambdaFunction` 나열된에 대한 참조를 제거합니다`Integration`.

Infrastructure Composer의 시각적 캔버스에 카드 정렬

정렬을 선택하여 캔버스에서 카드를 시각적으로 정렬하고 구성합니다. 정렬 버튼을 사용하면 캔버스에 카드와 연결이 많을 때 특히 유용합니다.



Infrastructure Composer에서 카드 구성 및 수정

Infrastructure Composer에서 카드는 애플리케이션 아키텍처를 설계하는 데 사용하는 리소스를 나타냅니다. Infrastructure Composer에서 카드를 구성할 때 애플리케이션의 리소스 세부 정보를 정의합니다. 여기에는 카드의 논리적 ID 및 파티션 키와 같은 세부 정보가 포함됩니다. 이 정보가 정의되는 방식은 향상된 구성 요소 카드와 표준 카드에 따라 다릅니다.

향상된 구성 요소 카드는 사용 편의성과 기능을 향상하고 다양한 사용 사례에 맞게 설계된 단일 큐레이션된 카드에 결합된 CloudFormation 리소스 모음입니다. 표준 IaC 리소스 카드는 단일 AWS CloudFormation 리소스를 나타냅니다. 캔버스로 끌면 각 표준 IaC 리소스 카드에 표준 구성 요소 레이블이 지정됩니다.

이 주제에서는 향상된 구성 요소 카드 및 표준 구성 요소 카드 구성에 대한 세부 정보를 제공합니다.

> **Note**
>
> 이 주제는 CloudFormation 콘솔 모드에서 Infrastructure Composer 콘솔, AWS Toolkit for Visual Studio Code 확장 및의 카드를 사용하는 데 적용됩니다. Lambda 관련 카드(Lambda 함수 및 Lambda 계층)에는 CloudFormation 콘솔 모드의 Infrastructure Composer에서 사용할 수 없는 코드 빌드 및 패키징 솔루션이 필요합니다. 자세한 내용은 CloudFormation 콘솔 모드에서 Infrastructure Composer 사용 단원을 참조하십시오.

주제

Infrastructure Composer의 향상된 구성 요소 카드

향상된 구성 요소 카드를 구성하기 위해 Infrastructure Composer는 리소스 속성 패널에 양식을 제공합니다. 이 양식은 각 향상된 구성 요소 카드를 구성하는 과정을 안내하기 위해 고유하게 큐레이션됩니다. 양식을 작성하면 Infrastructure Composer가 인프라 코드를 수정합니다.

일부 향상된 구성 요소 카드에는 추가 기능이 있습니다. 이 섹션에서는 향상된 구성 요소 카드 사용의 기본 사항을 검토하고 추가 기능이 있는 카드에 대한 세부 정보를 제공합니다.

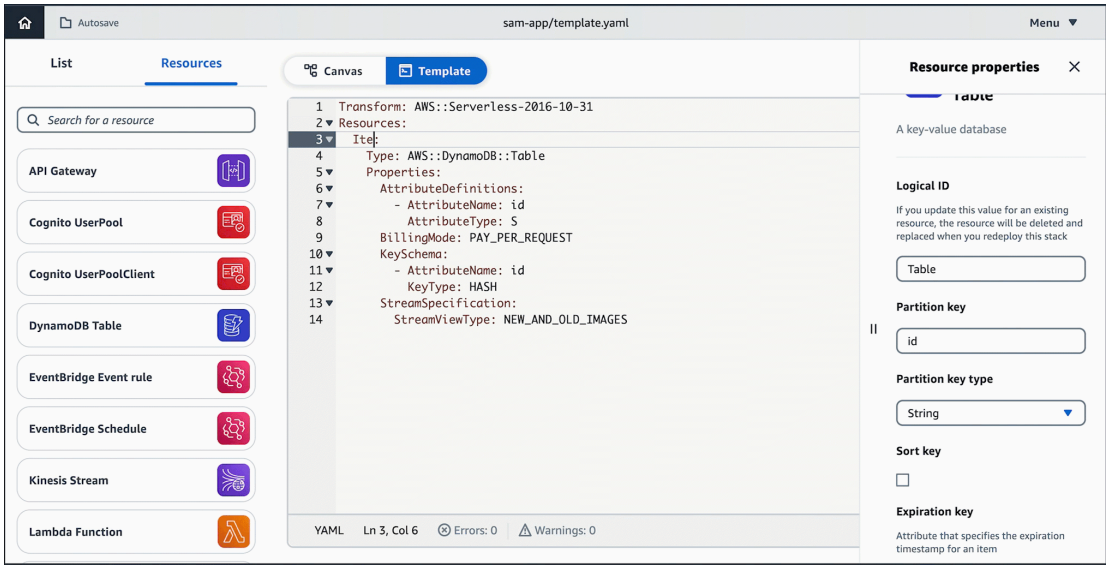
향상된 구성 요소 카드에 대한 자세한 내용은 Infrastructure Composer의 향상된 구성 요소 카드 및 섹션을 참조하세요. Infrastructure Composer의 향상된 구성 요소 카드

절차

리소스 속성 패널은 구성을 간소화하고 카드 구성을 간소화하는 가이드레일을 추가합니다. 이 패널을 사용하려면 다음 단계를 수행합니다.

1. 카드를 두 번 클릭하여 리소스 속성 패널을 불러옵니다.
2. 카드를 클릭하고 세부 정보를 선택하여 리소스 속성 패널을 불러옵니다.
3. 의 Infrastructure Composer에서 템플릿을 AWS Management Console선택하여 애플리케이션 코드를 표시합니다. 여기에서 직접를 구성합니다.

   다음 이미지는이 작업을 수행하는 방법을 보여줍니다.



Amazon Relational Database Service(RDS)에서 Infrastructure Composer 사용

AWS Infrastructure Composer 는 Amazon Relational Database Service(RDS)와의 통합을 지원합니다. Infrastructure Composer에서 RDS 데이터베이스(외부) 향상된 구성 요소 카드를 사용하여 애플리케이션을 다른 CloudFormation 또는 AWS Serverless Application Model (AWS SAM) 템플릿에 정의된 Amazon RDS DB 클러스터, 인스턴스 및 프록시에 연결할 수 있습니다.

RDS 데이터베이스(외부) 향상된 구성 요소 카드는 다른 템플릿에 정의된 Amazon RDS 리소스를 나타냅니다. 여기에는 다음이 포함됩니다.

- 다른 템플릿에 정의된 Amazon RDS DB 클러스터 또는 인스턴스
- Amazon RDS DB 프록시

RDS 데이터베이스(외부) 향상된 구성 요소 카드는 리소스 팔레트에서 사용할 수 있습니다.

이 카드를 사용하려면 Infrastructure Composer 캔버스로 끌어서 구성하고 다른 리소스에 연결합니다.

Lambda 함수를 통해 애플리케이션을 외부 Amazon RDS DB 클러스터 또는 인스턴스에 연결할 수 있습니다.

요구 사항

이 기능을 사용하려면 다음 요구 사항을 충족해야 합니다.

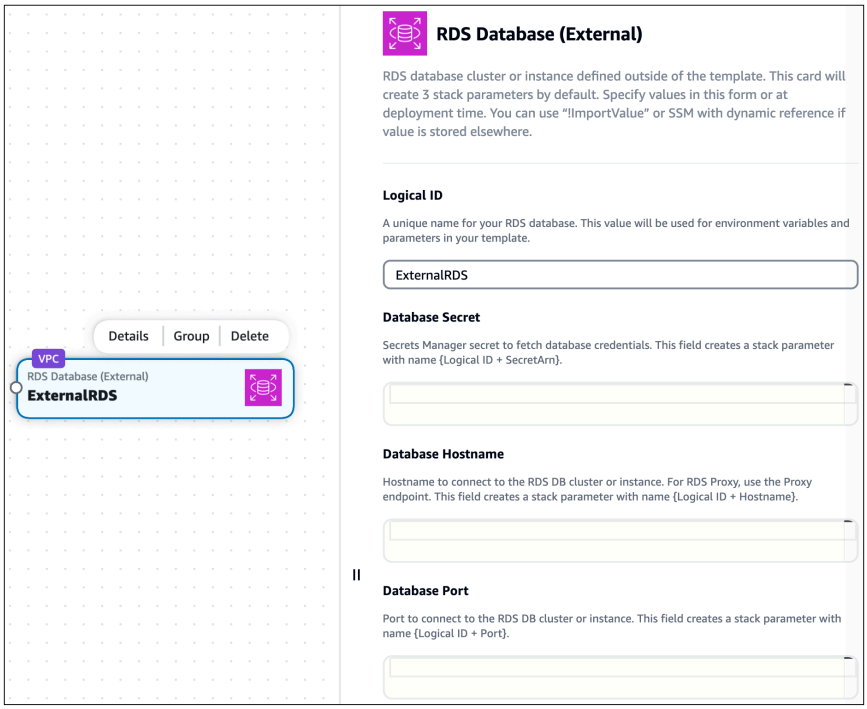
1. 외부 Amazon RDS DB 클러스터, 인스턴스 또는 프록시는를 사용하여 사용자 암호를 AWS Secrets Manager 관리해야 합니다. 자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS를 사용한 암호 관리 및 AWS Secrets Manager 섹션을 참조하세요.
2. Infrastructure Composer의 애플리케이션은 새 프로젝트이거나 Infrastructure Composer에서 원래 생성되었어야 합니다.

절차

1단계: 외부 RDS 데이터베이스 카드 구성

리소스 팔레트에서 RDS 데이터베이스(외부) 향상된 구성 요소 카드를 캔버스로 드래그합니다.

카드를 선택하고 세부 정보를 선택하거나 카드를 두 번 클릭하여 리소스 속성 패널을 불러옵니다. 카드의 리소스 속성 패널이 나타납니다.



여기에서 다음을 구성할 수 있습니다.

- 논리적 ID - 외부 Amazon RDS DB 클러스터, 인스턴스 또는 프록시의 고유한 이름입니다. 이 ID는 외부 Amazon RDS DB 리소스의 논리적 ID 값과 일치할 필요가 없습니다.

- 데이터베이스 보안 암호 - Amazon RDS DB 클러스터, 인스턴스 또는 프록시와 연결된 AWS Secrets Manager 보안 암호의 식별자입니다. 이 필드는 다음 값을 허용합니다.
  - 정적 값 - 보안 암호 ARN과 같은 데이터베이스 보안 암호의 고유 식별자입니다. 예를 들면, `arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-secret-name-1a2b3c`입니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 개념을 참조하세요.
  - 출력 값 - Secrets Manager 보안 암호가 배포되면 AWS CloudFormation출력 값이 생성됩니다. `Fn::ImportValue` 내장 함수를 사용하여 여기에서 출력 값을 지정할 수 있습니다. 예를 들어 `!ImportValue MySecret`입니다.
  - SSM 파라미터 스토어의 값 - SSM 파라미터 스토어에 보안 암호를 저장하고 동적 참조를 사용하여 해당 값을 지정할 수 있습니다. 예를 들어 `{{resolve:ssm:MySecret}}`입니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 SSM 파라미터를 참조하세요.
- 데이터베이스 호스트 이름 - Amazon RDS DB 클러스터, 인스턴스 또는 프록시에 연결하는 데 사용할 수 있는 호스트 이름입니다. 이 값은 Amazon RDS 리소스를 정의하는 외부 템플릿에 지정됩니다. 허용되는 값은 다음과 같습니다.
  - 정적 값 - 엔드포인트 주소와 같은 데이터베이스 호스트 이름의 고유 식별자입니다. 예를 들면, `mystack-mydb-1apw1j4phylrk.cg034hpkmmjt.us-east-2.rds.amazonaws.com`입니다.
  - 출력 값 - 배포된 Amazon RDS DB 클러스터, 인스턴스 또는 프록시의 출력 값입니다. `Fn::ImportValue` 내장 함수를 사용하여 출력 값을 지정할 수 있습니다. 예를 들어 `!ImportValue myStack-myDatabase-abcd1234`입니다.
  - SSM 파라미터 스토어의 값 - SSM 파라미터 스토어에 데이터베이스 호스트 이름을 저장하고 동적 참조를 사용하여 값을 지정할 수 있습니다. 예를 들어 `{{resolve:ssm:MyDatabase}}`입니다.
- 데이터베이스 포트 - Amazon RDS DB 클러스터, 인스턴스 또는 프록시에 연결하는 데 사용할 수 있는 포트 번호입니다. 이 값은 Amazon RDS 리소스를 정의하는 외부 템플릿에 지정됩니다. 허용되는 값은 다음과 같습니다.
  - 정적 값 - 데이터베이스 포트입니다. 예를 들어 `3306`입니다.
  - 출력 값 - 배포된 Amazon RDS DB 클러스터, 인스턴스 또는 프록시의 출력 값입니다. 예를 들어 `!ImportValue myStack-MyRDSInstancePort`입니다.
  - SSM 파라미터 스토어의 값 - SSM 파라미터 스토어에 데이터베이스 호스트 이름을 저장하고 동적 참조를 사용하여 값을 지정할 수 있습니다. 예를 들어 `{{resolve:ssm:MyRDSInstancePort}}`입니다.

> **Note**
>
> 논리적 ID 값만 여기에서 구성해야 합니다. 원하는 경우 배포 시 다른 속성을 구성할 수 있습니다.

2단계: Lambda 함수 카드 연결

리소스 팔레트에서 Lambda 함수 향상된 구성 요소 카드를 캔버스로 드래그합니다.

Lambda 함수 카드의 왼쪽 포트를 RDS 데이터베이스(외부) 카드의 오른쪽 포트에 연결합니다.

Infrastructure Composer는이 연결을 용이하게 하기 위해 템플릿을 프로비저닝합니다.

연결을 생성하기 위해 Infrastructure Composer가 수행하는 작업

위에 나열된 절차를 완료하면 Infrastructure Composer는 Lambda 함수를 데이터베이스에 연결하는 특정 작업을 수행합니다.

외부 Amazon RDS DB 클러스터, 인스턴스 또는 프록시를 지정하는 경우

RDS 데이터베이스(외부) 카드를 캔버스로 드래그하면 Infrastructure Composer는 필요에 따라 템플릿의 `Metadata` 및 `Parameters` 섹션을 업데이트합니다. 다음은 예제입니다.

```
Metadata:
  AWS::Composer::ExternalResources:
    ExternalRDS:
      Type: externalRDS
      Settings:
        Port: !Ref ExternalRDSPort
        Hostname: !Ref ExternalRDSHostname
        SecretArn: !Ref ExternalRDSSecretArn
Parameters:
  ExternalRDSPort:
    Type: Number
  ExternalRDSHostname:
    Type: String
  ExternalRDSSecretArn:
    Type: String
```

메타데이터는 CloudFormation 템플릿에 대한 세부 정보를 저장하는 데 사용되는 템플릿 섹션입니다. Infrastructure Composer와 관련된 메타데이터는 `AWS::Composer::ExternalResources` 메타데이터 키 아래에 저장됩니다. 여기서 Infrastructure Composer는 Amazon RDS DB 클러스터, 인스턴스 또는 프록시에 대해 지정한 값을 저장합니다.

CloudFormation 템플릿의 파라미터 섹션은 배포 시 템플릿 전체에 삽입할 수 있는 사용자 지정 값을 저장하는 데 사용됩니다. 제공하는 값의 유형에 따라 Infrastructure Composer는 Amazon RDS DB 클러스터, 인스턴스 또는 프록시에 대한 값을 여기에 저장하고 템플릿 전체에서 지정할 수 있습니다.

`Metadata` 및 `Parameters` 섹션의 문자열 값은 RDS 데이터베이스(외부) 카드에 지정한 논리적 ID 값을 사용합니다. 논리적 ID를 업데이트하면 문자열 값이 변경됩니다.

Lambda 함수를 데이터베이스에 연결하는 경우

Lambda 함수 카드를 RDS 데이터베이스(외부) 카드에 연결하면 Infrastructure Composer는 환경 변수 및 AWS Identity and Access Management (IAM) 정책을 프로비저닝합니다. 다음은 예제입니다.

```
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      ...
      Environment:
        Variables:
          EXTERNALRDS_PORT: !Ref ExternalRDSPort
          EXTERNALRDS_HOSTNAME: !Ref ExternalRDSHostname
          EXTERNALRDS_SECRETARN: !Ref ExternalRDSSecretArn
      Policies:
        - AWSSecretsManagerGetSecretValuePolicy:
            SecretArn: !Ref ExternalRDSSecretArn
```

환경 변수는 런타임 시 함수에서 사용할 수 있는 변수입니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 환경 변수 사용을 참조하세요.

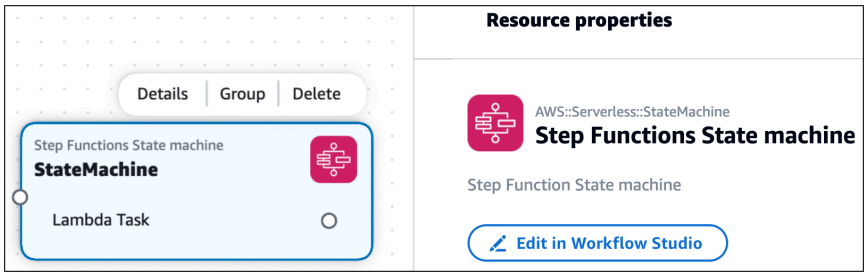
정책은 함수에 대한 권한을 프로비저닝합니다. 여기서 Infrastructure Composer는 함수에서 Secrets Manager로의 읽기 액세스를 허용하는 정책을 생성하여 Amazon RDS DB 클러스터, 인스턴스 또는 프록시에 액세스하기 위한 암호를 가져옵니다.

AWS Infrastructure Composer 와 함께 사용 AWS Step Functions

AWS Infrastructure Composer 는 와의 통합을 지원합니다AWS Step Functions Workflow Studio. Infrastructure Composer를 사용하여 다음을 수행합니다.

- Infrastructure Composer 내에서 Workflow Studio 직접 Step Functions를 시작합니다.
- 새 워크플로를 생성 및 관리하거나 기존 워크플로를 Infrastructure Composer로 가져옵니다.
- Infrastructure Composer 캔버스를 사용하여 워크플로를 다른 AWS 리소스와 통합합니다.

다음 이미지는 Step Functions 상태 시스템 카드입니다.



Infrastructure ComposerWorkflow Studio의 Step Functions를 사용하면 두 개의 강력한 시각적 디자이너의 이점을 한 곳에서 사용할 수 있습니다. 워크플로와 애플리케이션을 설계할 때 Infrastructure Composer는 코드형 인프라(IaC)를 생성하여 배포를 안내합니다.

주제

- IAM 정책
- Infrastructure ComposerWorkflow Studio에서 Step Functions 시작하기
- Infrastructure ComposerWorkflow Studio에서 Step Functions 사용
- 자세히 알아보기

IAM 정책

워크플로의 작업을 리소스에 연결하면 Infrastructure Composer는 리소스 간의 상호 작용을 승인하는 데 필요한 AWS Identity and Access Management (IAM) 정책을 자동으로 생성합니다. 다음은 예제입니다.

```
Transform: AWS::Serverless-2016-10-31
Resources:
  StockTradingStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      ...
      Policies:
        - LambdaInvokePolicy:
            FunctionName: !Ref CheckStockValue
```

```
    ...
CheckStockValue:
  Type: AWS::Serverless::Function
  ...
```

필요한 경우 템플릿에 IAM 정책을 더 추가할 수 있습니다.

Infrastructure ComposerWorkflow Studio에서 Step Functions 시작하기

시작하려면 새 워크플로를 생성하거나 기존 워크플로를 가져올 수 있습니다.

새 워크플로를 생성하려면

1. 리소스 팔레트에서 Step Functions 상태 시스템 향상된 구성 요소 카드를 캔버스로 드래그합니다.

   Step Functions 상태 시스템 카드를 캔버스로 드래그하면 Infrastructure Composer가 다음을 생성합니다.

   - 상태 시스템을 정의하는 `AWS::Serverless::StateMachine` 리소스입니다. 기본적으로 Infrastructure Composer는 표준 워크플로를 생성합니다. Express 워크플로를 생성하려면 템플릿의 `Type` 값을에서 `STANDARD`로 변경합니다`EXPRESS`.
   - 상태 시스템에 대한 Amazon CloudWatch 로그 그룹을 정의하는 `AWS::Logs::LogGroup` 리소스입니다.
2. 카드의 리소스 속성 패널을 열고 Workflow Studio에서 편집을 선택하여 Infrastructure Composer Workflow Studio 내에서 엽니다.

   Step Functions가 디자인 모드에서 Workflow Studio 열립니다. 자세한 내용은 AWS Step Functions 개발자 안내서의 디자인 모드를 참조하세요.

   > **Note**
   >
   > Infrastructure Composer를 수정하여 상태 시스템 정의를 외부 파일에 저장할 수 있습니다. 자세한 내용은 외부 파일 작업를 참조하세요.
3. 워크플로를 생성하고 저장을 선택합니다. 를 종료하려면 인프라 컴포저로 돌아가기를 Workflow Studio선택합니다.

Infrastructure Composer는 `AWS::Serverless::StateMachine` 리소스의 `Defintion` 속성을 사용하여 워크플로를 정의합니다.

4. 다음 중 하나를 수행하여 워크플로를 수정할 수 있습니다.

   - 를 Workflow Studio 다시 열고 워크플로를 수정합니다.
   - 콘솔에서 Infrastructure Composer의 경우 애플리케이션의 템플릿 보기를 열고 템플릿을 수정할 수 있습니다. 로컬 동기화를 사용하는 경우 로컬 IDE에서 워크플로를 수정할 수 있습니다. Infrastructure Composer는 Infrastructure Composer에서 변경 사항을 감지하고 워크플로를 업데이트합니다.
   - Toolkit for VS Code의 Infrastructure Composer의 경우 템플릿을 직접 수정할 수 있습니다. Infrastructure Composer는 Infrastructure Composer에서 변경 사항을 감지하고 워크플로를 업데이트합니다.

기존 워크플로를 가져오려면

AWS Serverless Application Model (AWS SAM) 템플릿을 사용하여 정의된 애플리케이션에서 워크플로를 가져올 수 있습니다. `AWS::Serverless::StateMachine` 리소스 유형으로 정의된 상태 시스템을 사용하면를 시작하는 데 사용할 수 있는 Step Functions 상태 시스템 향상된 구성 요소 카드로 시각화됩니다Workflow Studio.

`AWS::Serverless::StateMachine` 리소스는 다음 속성 중 하나를 사용하여 워크플로를 정의할 수 있습니다.

- `Definition` - 워크플로는 AWS SAM 템플릿 내에서 객체로 정의됩니다.
- `DefinitionUri` - 워크플로는 Amazon States Language를 사용하여 외부 파일에 정의됩니다. 그러면 파일의 로컬 경로가이 속성으로 지정됩니다.

정의 속성

콘솔의 Infrastructure Composer

`Definition` 속성을 사용하여 정의된 워크플로의 경우 단일 템플릿 또는 전체 프로젝트를 가져올 수 있습니다.

- 템플릿 - 템플릿 가져오기에 대한 지침은 섹션을 참조하세요Infrastructure Composer 콘솔에서 기존 프로젝트 템플릿 가져오기. Infrastructure Composer 내에서 변경한 내용을 저장하려면 템플릿을 내보내야 합니다.

- 프로젝트 - 프로젝트를 가져올 때 로컬 동기화를 활성화해야 합니다. 변경 사항은 로컬 시스템에 자동으로 저장됩니다. 프로젝트 가져오기에 대한 지침은 섹션을 참조하세요Infrastructure Composer 콘솔에서 기존 프로젝트 폴더 가져오기.

Toolkit for VS Code의 Infrastructure Composer

`Definition` 속성을 사용하여 정의된 워크플로의 경우 템플릿에서 Infrastructure Composer를 열 수 있습니다. 지침은 에서 Infrastructure Composer에 액세스 AWS Toolkit for Visual Studio Code 섹션을 참조하세요.

DefinitionUri 속성

콘솔의 Infrastructure Composer

`DefinitionUri` 속성을 사용하여 정의된 워크플로의 경우 프로젝트를 가져오고 로컬 동기화를 활성화해야 합니다. 프로젝트 가져오기에 대한 지침은 섹션을 참조하세요Infrastructure Composer 콘솔에서 기존 프로젝트 폴더 가져오기.

Toolkit for VS Code의 Infrastructure Composer

`DefinitionUri` 속성을 사용하여 정의된 워크플로의 경우 템플릿에서 Infrastructure Composer를 열 수 있습니다. 지침은 에서 Infrastructure Composer에 액세스 AWS Toolkit for Visual Studio Code 섹션을 참조하세요.

Infrastructure ComposerWorkflow Studio에서 Step Functions 사용

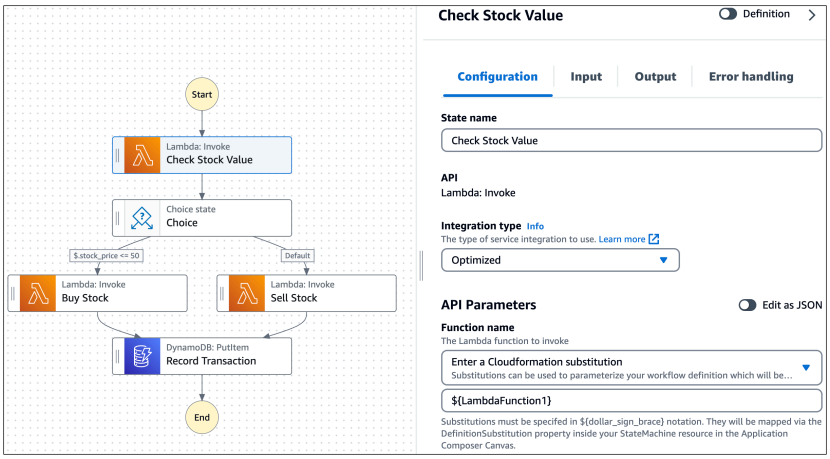
빌드 워크플로

Infrastructure Composer는 정의 대체를 사용하여 워크플로 작업을 애플리케이션의 리소스에 매핑합니다. 정의 대체에 대한 자세한 내용은 AWS Serverless Application Model 개발자 안내서 `DefinitionSubstitutions`의 섹션을 참조하세요.

에서 작업을 생성할 때 각 작업에 대한 정의 대체를 Workflow Studio지정합니다. 그런 다음 Infrastructure Composer 캔버스의 리소스에 작업을 연결할 수 있습니다.

에서 정의 대체를 지정하려면 Workflow Studio

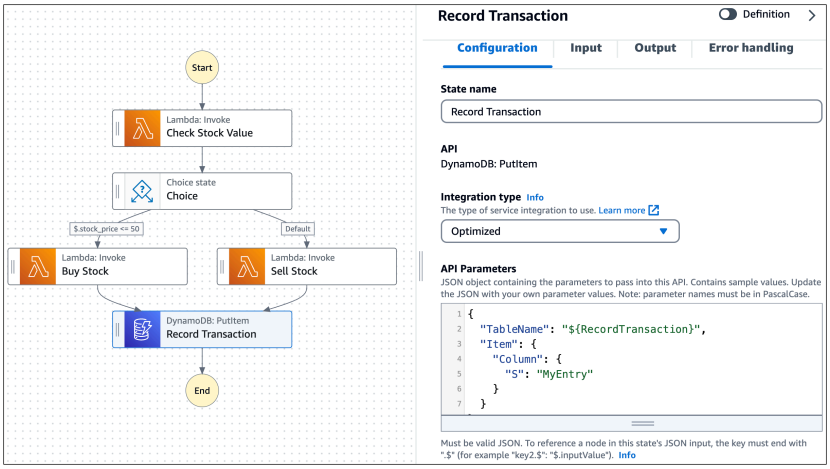
1. 작업의 구성 탭을 열고 API 파라미터 필드를 찾습니다.



2. API 파라미터 필드에 드롭다운 옵션이 있는 경우 CloudFormation 대체 항목 입력을 선택합니다. 그런 다음 고유한 이름을 입력합니다.

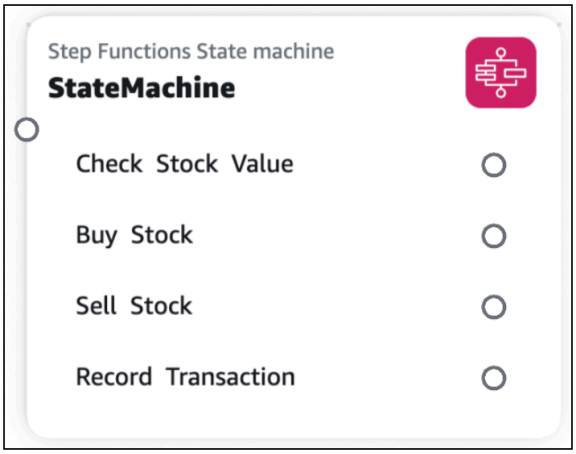
   동일한 리소스에 연결하는 작업의 경우 각 작업에 대해 동일한 정의 대체를 지정합니다. 기존 정의 대체를 사용하려면 CloudFormation 대체 선택을 선택하고 사용할 대체를 선택합니다.

3. API 파라미터 필드에 JSON 객체가 포함된 경우 정의 대체를 사용할 리소스 이름을 지정하는 항목을 수정합니다. 다음 예제에서는를 `"MyDynamoDBTable"`로 변경합니다`"${RecordTransaction}"`.



4. 저장을 선택하고 인프라 컴포저로 돌아가기를 선택합니다.

워크플로의 작업은 Step Functions 상태 시스템 카드에 시각화됩니다.



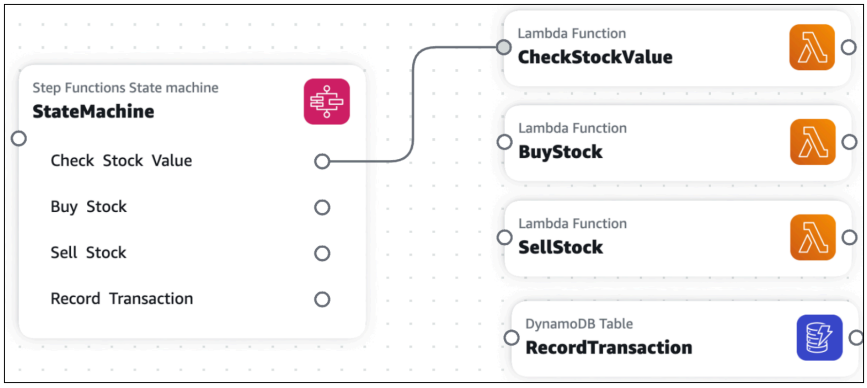
워크플로 작업에 리소스 연결

Infrastructure Composer에서 지원되는 워크플로 작업과 지원되는 Infrastructure Composer 카드 간의 연결을 생성할 수 있습니다.

- 지원되는 워크플로 작업 - Step Functions에 최적화된에 대한 작업 AWS 서비스 입니다. 자세한 내용은 AWS Step Functions 개발자 안내서의 Step Functions에 최적화된 통합을 참조하세요.
- 지원되는 Infrastructure Composer 카드 - 향상된 구성 요소 카드가 지원됩니다. Infrastructure Composer의 카드에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer에서 카드 구성 및 수정.

연결을 생성할 때 작업과 카드 AWS 서비스 의가 일치해야 합니다. 예를 들어 Lambda 함수를 호출하는 워크플로 태스크를 Lambda 함수 향상된 구성 요소 카드에 연결할 수 있습니다.

연결을 생성하려면 작업 포트를 클릭하고 향상된 구성 요소 카드의 왼쪽 포트로 끕니다.



Infrastructure Composer는 연결을 정의하기 위해 `DefinitionSubstitution` 값을 자동으로 업데이트합니다. 다음은 예제입니다.

```
Transform: AWS::Serverless-2016-10-31
Resources:
  StateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      Definition:
        StartAt: Check Stock Value
        States:
          Check Stock Value:
            Type: Task
            Resource: arn:aws:states:::lambda:invoke
            Parameters:
              Payload.$: $
              FunctionName: ${CheckStockValue}
            Next: Choice
          ...
      DefinitionSubstitutions:
        CheckStockValue: !GetAtt CheckStockValue.Arn
        ...
  CheckStockValue:
    Type: AWS::Serverless::Function
    Properties:
      ...
```

외부 파일 작업

Step Functions 상태 시스템 카드에서 워크플로를 생성하면 Infrastructure Composer는 `Definition` 속성을 사용하여 템플릿 내에 상태 시스템 정의를 저장합니다. 상태 시스템 정의를 외부 파일에 저장하도록 Infrastructure Composer를 구성할 수 있습니다.

> **Note**
>
> 의 Infrastructure Composer에서이 기능을 사용하려면 로컬 동기화가 활성화되어 있어야 AWS Management Console합니다. 자세한 내용은 Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장 단원을 참조하십시오.

상태 시스템 정의를 외부 파일에 저장하려면

1. Step Functions 상태 시스템 카드의 리소스 속성 패널을 엽니다.
2. 상태 시스템 정의에 외부 파일 사용 옵션을 선택합니다.

3. 상태 시스템 정의 파일의 상대 경로와 이름을 제공합니다.
4. 저장을 선택합니다.

Infrastructure Composer는 다음을 수행합니다.

1. 상태 시스템 정의를 `Definition` 필드에서 외부 파일로 이동합니다.
2. Amazon States Language를 사용하여 상태 시스템 정의를 외부 파일에 저장합니다.
3. `DefinitionUri` 필드를 사용하여 외부 파일을 참조하도록 템플릿을 수정합니다.

자세히 알아보기

Infrastructure Composer의 Step Functions에 대한 자세한 내용은 다음을 참조하세요.

- AWS Step Functions 개발자 안내서Workflow Studio의 Infrastructure Composer에서 사용.
- AWS Step Functions 개발자 안내서의 AWS SAM 템플릿의 DefinitionSubstitutions.

Infrastructure Composer의 표준 카드

모든 CloudFormation 리소스는 리소스 팔레트에서 표준 IaC 리소스 카드로 사용할 수 있습니다. 시각적 캔버스로 끌면 표준 IaC 리소스 카드가 표준 구성 요소 카드가 됩니다. 이는 카드가 하나 이상의 표준 IaC 리소스임을 의미합니다. 추가 예제 및 세부 정보는이 섹션의 주제를 참조하세요.

템플릿 보기와 리소스 속성 창을 통해 인프라 코드를 수정할 수 있습니다. 예를 들어 다음은 `Alexa::ASK::Skill` 표준 IaC 리소스의 시작 템플릿 예제입니다.

```
Resources:
  Skill:
    Type: Alexa::ASK::Skill
    Properties:
      AuthenticationConfiguration:
        RefreshToken: <String>
        ClientSecret: <String>
        ClientId: <String>
      VendorId: <String>
      SkillPackage:
        S3Bucket: <String>
        S3Key: <String>
```

표준 IaC 리소스 카드 시작 템플릿은 다음으로 구성됩니다.

- CloudFormation 리소스 유형입니다.
- 필수 또는 일반적으로 사용되는 속성입니다.
- 각 속성에 제공할 값의 필수 유형입니다.

> **Note**
>
> Amazon Q를 사용하여 표준 리소스 카드에 대한 인프라 코드 제안을 생성할 수 있습니다. 자세한 내용은 AWS Infrastructure Composer 와 함께 사용 Amazon Q Developer를 참조하세요.

절차

리소스 속성 패널을 통해 표준 구성 요소 카드의 각 리소스에 대한 인프라 코드를 수정할 수 있습니다.

표준 구성 요소 카드를 수정하려면

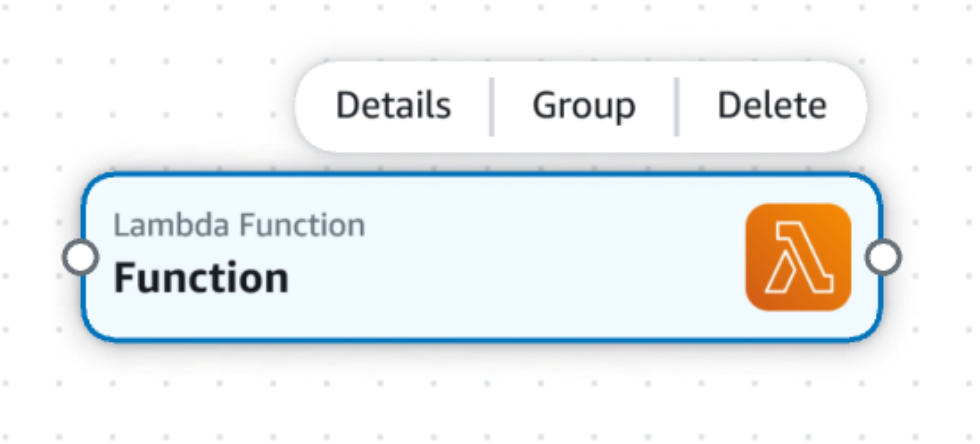
1. 표준 IaC 구성 요소 카드의 리소스 속성 패널을 엽니다.
2. 편집 필드의 드롭다운 목록에서 편집할 표준 IaC 리소스를 선택합니다.
3. 인프라 코드를 수정하고 저장합니다.

Infrastructure Composer에서 카드 삭제

이 섹션에서는에서 카드를 삭제하는 지침을 제공합니다 AWS Infrastructure Composer.

향상된 구성 요소 카드

향상된 구성 요소 카드를 삭제하려면 시각적 캔버스에 배치한 카드를 선택합니다. 카드 작업 메뉴에서 삭제를 선택합니다.



표준 구성 요소 카드

표준 구성 요소 카드를 삭제하려면 템플릿에서 각 CloudFormation 리소스의 인프라 코드를 수동으로 제거해야 합니다. 다음은 이를 수행하는 간단한 방법입니다.

1. 삭제할 리소스의 논리적 ID를 기록해 둡니다.
2. 템플릿의 `Resources` 또는 `Outputs` 섹션에서 논리적 ID로 리소스를 찾습니다.
3. 템플릿에서 리소스를 삭제합니다. 여기에는 리소스 논리적 ID와 `Type` 및와 같은 중첩된 값이 포함됩니다`Properties`.
4. Canvas 뷰를 확인하여 리소스가 캔버스에서 제거되었는지 확인합니다.

Infrastructure Composer에서 Change Inspector로 코드 업데이트 보기

Infrastructure Composer 콘솔에서 설계하면 인프라 코드가 자동으로 생성됩니다. Change Inspector를 사용하여 템플릿 코드 업데이트를 보고 Infrastructure Composer가 생성하는 내용을 알아봅니다.

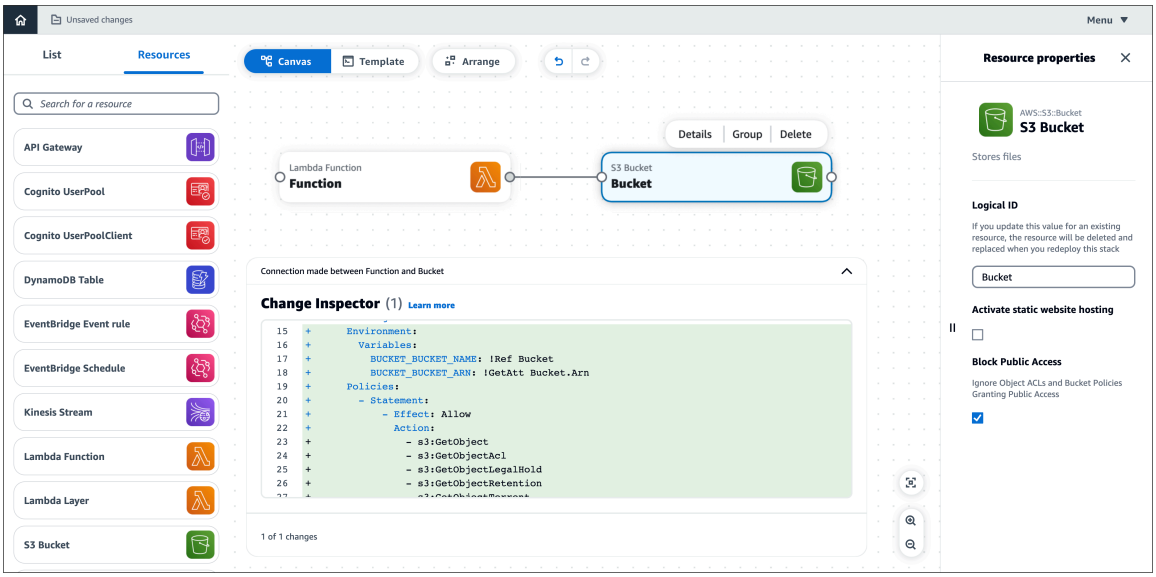
이 주제에서는 AWS Management Console 또는 AWS Toolkit for Visual Studio Code 확장에서 Infrastructure Composer를 사용하는 방법을 다룹니다.

Change Inspector는 최근 코드 업데이트를 보여주는 Infrastructure Composer 내의 시각적 도구입니다.

- 애플리케이션을 설계할 때 시각적 캔버스 하단에 메시지가 표시됩니다. 이러한 메시지는 수행 중인 작업에 대한 설명을 제공합니다.

- 지원되는 경우 메시지를 확장하여 Change Inspector를 볼 수 있습니다.
- Change Inspector에는 가장 최근 상호 작용의 코드 변경 사항이 표시됩니다.

다음 예제에서는 변경 검사기의 작동 방식을 보여줍니다.



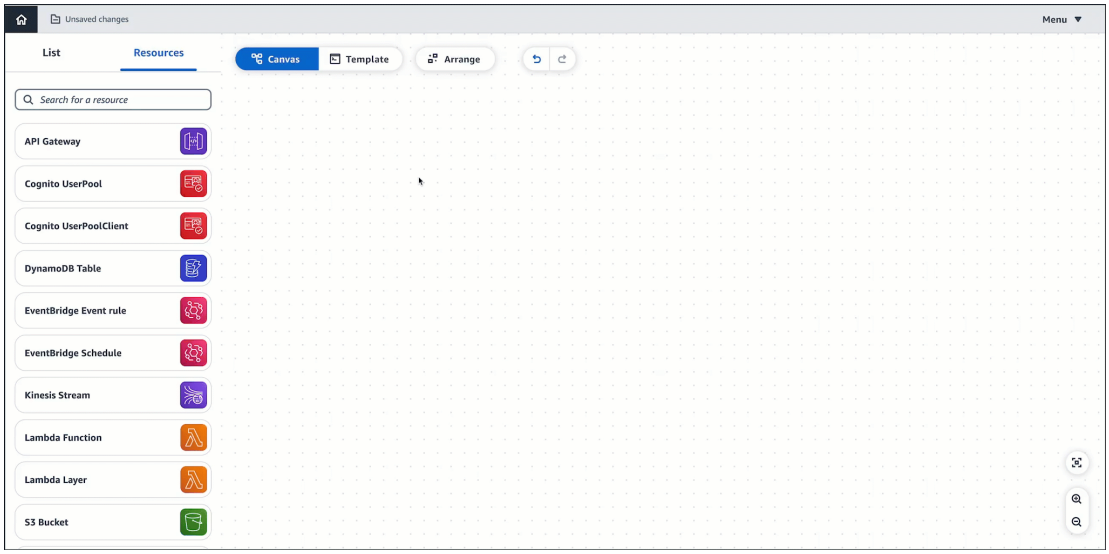
Change Inspector의 이점

Change Inspector는 Infrastructure Composer가 생성하는 템플릿 코드를 볼 수 있는 좋은 방법입니다. 또한 인프라 코드를 작성하는 방법을 배울 수 있는 좋은 방법입니다. Infrastructure Composer에서 애플리케이션을 설계할 때 Change Inspector에서 코드 업데이트를 보고 설계를 프로비저닝하는 데 필요한 코드에 대해 알아봅니다.

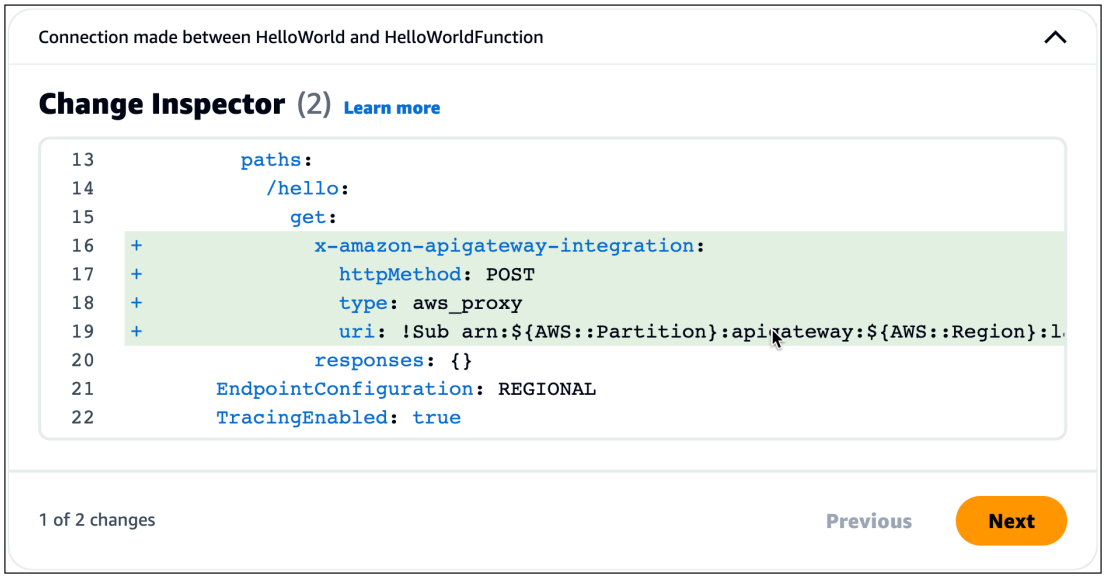
절차

Change Inspector를 사용하려면

1. 메시지를 확장하여 Change Inspector를 불러옵니다.



2. 자동으로 구성된 코드를 확인합니다.



   a. 녹색으로 강조 표시된 코드는 새로 추가된 코드를 나타냅니다.

   b. 빨간색으로 강조 표시된 코드는 새로 제거된 코드를 나타냅니다.

   c. 행 번호는 템플릿 내의 위치를 나타냅니다.

3. 템플릿의 여러 섹션이 업데이트되면 Change Inspector가 이를 구성합니다. 모든 변경 사항을 보려면 이전 및 다음 버튼을 선택합니다.

> **Note**
>
> 콘솔의 Infrastructure Composer의 경우 템플릿 보기를 사용하여 전체 템플릿의 컨텍스트에서 코드 변경 사항을 볼 수 있습니다. Infrastructure Composer를 로컬 IDE와 동기화하고 로컬 시스템에서 전체 템플릿을 볼 수도 있습니다. 자세한 내용은 Infrastructure Composer 콘솔을 로컬 IDE에 연결를 참조하세요.

자세히 알아보기

Infrastructure Composer가 생성하는 코드에 대한 자세한 내용은 다음을 참조하세요.

- Infrastructure Composer의 카드 연결.

Infrastructure Composer에서 외부 파일 참조

외부 파일을 AWS Serverless Application Model (AWS SAM) 템플릿과 함께 사용하여 반복된 코드를 재사용하고 프로젝트를 구성할 수 있습니다. 예를 들어 OpenAPI 사양에 설명된 Amazon API Gateway REST API 리소스가 여러 개 있을 수 있습니다. 템플릿에서 OpenAPI 사양 코드를 복제하는 대신 하나의 외부 파일을 생성하고 각 리소스에 대해 참조할 수 있습니다.

AWS Infrastructure Composer 는 다음과 같은 외부 파일 사용 사례를 지원합니다.

- 외부 OpenAPI 사양 파일에서 정의한 API Gateway REST API 리소스입니다.

- AWS Step Functions 외부 상태 시스템 정의 파일로 정의된 상태 시스템 리소스입니다.

지원되는 리소스에 대한 외부 파일 구성에 대한 자세한 내용은 다음을 참조하세요.

- `DefinitionBody`(`AWS::Serverless::Api`일 때)
- `DefinitionUri`(`AWS::Serverless::StateMachine`일 때)

> **Note**
>
> Infrastructure Composer 콘솔에서 Infrastructure Composer를 사용하여 외부 파일을 참조하려면 로컬 동기화 모드에서 Infrastructure Composer를 사용해야 합니다. 자세한 내용은 Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장 단원을 참조하십시오.

주제

Infrastructure Composer 외부 참조 파일의 모범 사례

로컬 IDE에서 Infrastructure Composer 사용

로컬 동기화 모드에서 로컬 IDE와 함께 Infrastructure Composer를 사용하는 경우 로컬 IDE를 사용하여 외부 파일을 보고 수정할 수 있습니다. 템플릿에서 참조되는 지원되는 외부 파일의 콘텐츠는 Infrastructure Composer 캔버스에서 자동으로 업데이트됩니다. 자세한 내용은 Infrastructure Composer 콘솔을 로컬 IDE에 연결를 참조하세요.

프로젝트의 상위 디렉터리 내에 외부 파일 유지

프로젝트의 상위 디렉터리 내에 하위 디렉터리를 생성하여 외부 파일을 구성할 수 있습니다. Infrastructure Composer는 프로젝트의 상위 디렉터리 외부의 디렉터리에 저장된 외부 파일에 액세스할 수 없습니다.

를 사용하여 애플리케이션 배포 AWS SAM CLI

애플리케이션을에 배포할 때 AWS 클라우드로컬 외부 파일을 먼저 Amazon Simple Storage Service(Amazon S3)와 같은 액세스 가능한 위치에 업로드해야 합니다. AWS SAM CLI를 사용하여이 프로세스를 자동으로 촉진할 수 있습니다. 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 배포 시 로컬 파일 업로드를 참조하세요.

Infrastructure Composer에서 외부 파일 참조 생성

지원되는 리소스의 리소스 속성 패널에서 외부 파일 참조를 생성할 수 있습니다.

외부 파일 참조를 생성하려면

1. API Gateway 또는 Step Functions 향상된 구성 요소 카드에서 세부 정보를 선택하여 리소스 속성 패널을 불러옵니다.
2. 외부 파일 사용 옵션을 찾아 선택합니다.
3. 외부 파일의 상대 경로를 지정합니다. 파일에서 외부 `template.yaml` 파일까지의 경로입니다.

   예를 들어 다음 프로젝트의 구조에서 `api-spec.yaml` 외부 파일을 참조하려면를 상대 경로`./api-spec.yaml`로 지정합니다.

   ```
   demo
   ### api-spec.yaml
   ### src
   # ### Function
   # ### index.js
   # ### package.json
   ### template.yaml
   ```

   > **Note**
   >
   > 외부 파일과 지정된 경로가 없는 경우 Infrastructure Composer에서 생성합니다.

4. 변경 사항을 저장합니다.

Infrastructure Composer에서 외부 파일 참조가 있는 프로젝트 로드

이 페이지에 나열된 단계에 따라 외부 파일 참조가 있는 Infrastructure Composer 프로젝트를 로드합니다.

Infrastructure Composer 콘솔에서

1. 에 나열된 단계를 완료합니다Infrastructure Composer 콘솔에서 기존 프로젝트 템플릿 가져오기.
2. Infrastructure Composer가 프로젝트의 루트 폴더에 연결하라는 메시지를 표시하는지 확인합니다.

브라우저가 파일 시스템 액세스 API를 지원하는 경우 Infrastructure Composer는 프로젝트의 루트 폴더에 연결하라는 메시지를 표시합니다. Infrastructure Composer는 프로젝트를 로컬 동기화 모드로 열어 외부 파일을 지원합니다. 참조된 외부 파일이 지원되지 않는 경우 오류 메시지가 표시됩니다. 오류 메시지에 대한 자세한 내용은 섹션을 참조하세요문제 해결.

Toolkit for VS Code에서

1. 에 나열된 단계를 완료합니다에서 Infrastructure Composer에 액세스 AWS Toolkit for Visual Studio Code.
2. Infrastructure Composer에서 보려는 템플릿을 엽니다.

템플릿에서 Infrastructure Composer에 액세스하면 Infrastructure Composer가 외부 파일을 자동으로 감지합니다. 참조된 외부 파일이 지원되지 않는 경우 오류 메시지가 표시됩니다. 오류 메시지에 대한 자세한 내용은 섹션을 참조하세요문제 해결.

Infrastructure Composer에서 외부 파일을 참조하는 애플리케이션 생성

이 예제에서는를 AWS SAM CLI 사용하여 상태 시스템 정의에 대한 외부 파일을 참조하는 애플리케이션을 생성합니다. 그런 다음 외부 파일을 올바르게 참조하여 Infrastructure Composer에 프로젝트를 로드합니다.

예제

1. 먼저 AWS SAM CLI sam init 명령을 사용하여 라는 새 애플리케이션을 초기화합니다demo. 대화형 흐름 중에 다단계 워크플로 빠른 시작 템플릿을 선택합니다.

```
$ sam init

...

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1
```

```
Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        ...
Template: 2

Which runtime would you like to use?
        1 - dotnet6
        2 - dotnetcore3.1
        ...
        15 - python3.7
        16 - python3.10
        17 - ruby2.7
Runtime: 16

Based on your selections, the only Package type available is Zip.
We will proceed to selecting the Package type as Zip.

Based on your selections, the only dependency manager available is pip.
We will proceed copying the template using pip.

Would you like to enable X-Ray tracing on the function(s) in your application?  [y/
N]: ENTER

Would you like to enable monitoring using CloudWatch Application Insights?
For more info, please view https://docs.aws.amazon.com/AmazonCloudWatch/latest/
monitoring/cloudwatch-application-insights.html [y/N]: ENTER

Project name [sam-app]: demo

    -----------------------
    Generating application:
    -----------------------
    Name: demo
    Runtime: python3.10
    Architectures: x86_64
    Dependency Manager: pip
    Application Template: step-functions-sample-app
    Output Directory: .
    Configuration file: demo/samconfig.toml
```

```
    Next steps can be found in the README file at demo/README.md

...
```

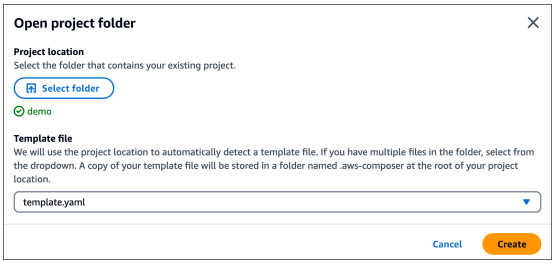
이 애플리케이션은 상태 시스템 정의에 대한 외부 파일을 참조합니다.

```
...
Resources:
  StockTradingStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
...
```

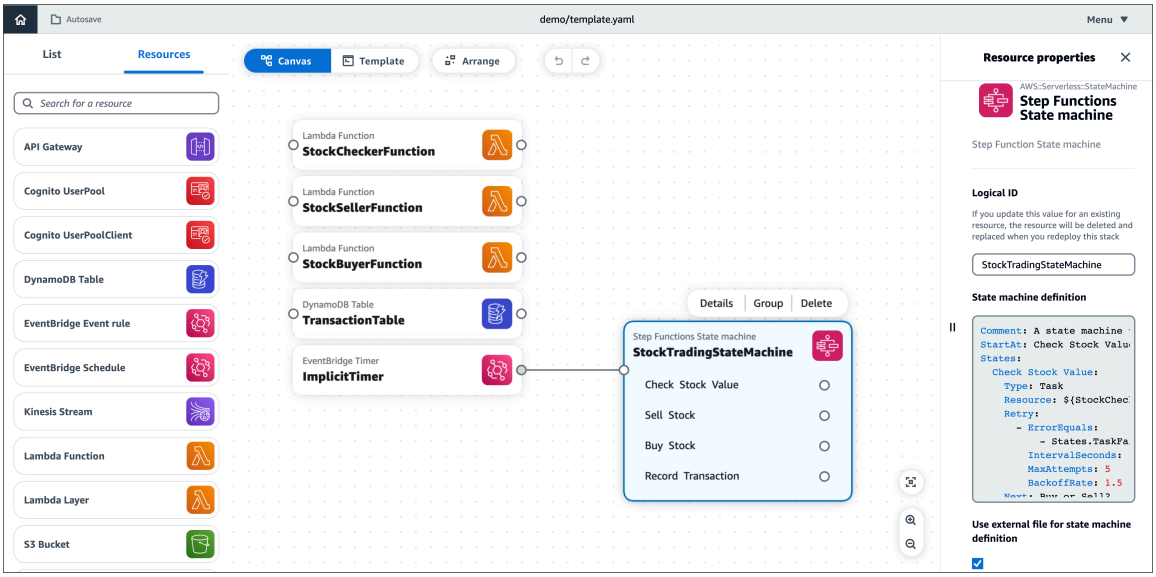
외부 파일은 애플리케이션의 `statemachine` 하위 디렉터리에 있습니다.

```
demo
### README.md
### __init__.py
### functions
#   ### __init__.py
#   ### stock_buyer
#   ### stock_checker
#   ### stock_seller
### samconfig.toml
### statemachine
#   ### stock_trader.asl.json
### template.yaml
### tests
```

2. 그런 다음 콘솔에서 Infrastructure Composer에 애플리케이션을 로드합니다. Infrastructure Composer 홈 페이지에서 CloudFormation 템플릿 로드를 선택합니다.
3. demo 프로젝트 폴더를 선택하고 프롬프트가 파일을 볼 수 있도록 허용합니다. `template.yaml` 파일을 선택하고 생성을 선택합니다. 메시지가 표시되면 변경 사항 저장을 선택합니다.



Infrastructure Composer는 외부 상태 시스템 정의 파일을 자동으로 감지하여 로드합니다. StockTradingStateMachine 리소스를 선택하고 세부 정보를 선택하여 리소스 속성 패널을 표시합니다. 여기서 Infrastructure Composer가 외부 상태 시스템 정의 파일에 자동으로 연결되었음을 확인할 수 있습니다.



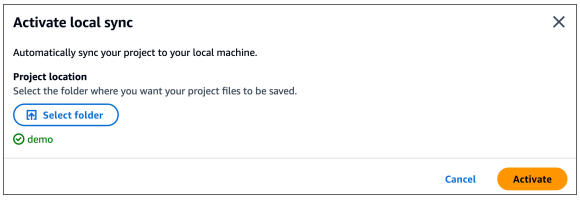
상태 시스템 정의 파일에 대한 모든 변경 사항은 Infrastructure Composer에 자동으로 반영됩니다.

Infrastructure Composer를 사용하여 OpenAPI 사양 외부 파일 참조

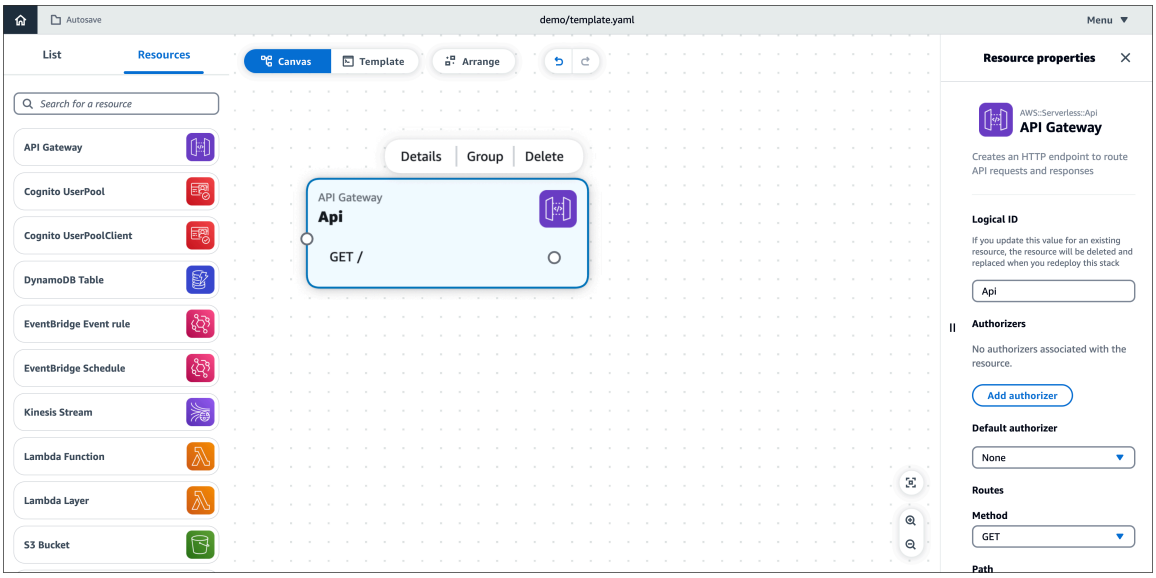
이 예제에서는 콘솔의 Infrastructure Composer를 사용하여 API Gateway를 정의하는 외부 OpenAPI 사양 파일을 참조합니다REST API.

먼저 Infrastructure Composer 홈 페이지에서 새 프로젝트를 생성합니다.

그런 다음 메뉴에서 로컬 동기화 활성화를 선택하여 로컬 동기화를 활성화합니다. 라는 새 폴더를 생성하고 demo프롬프트가 파일을 볼 수 있도록 허용한 다음 활성화를 선택합니다. 메시지가 표시되면 변경 사항 저장을 선택합니다.

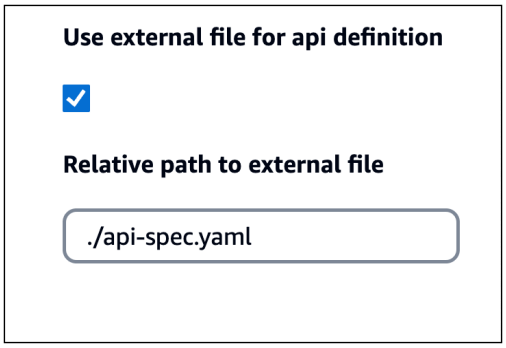


그런 다음 Amazon API Gateway 카드를 캔버스로 드래그합니다. 세부 정보를 선택하여 리소스 속성 패널을 불러옵니다.



리소스 속성 패널에서 다음을 구성하고 저장합니다.

- API 정의에 외부 파일 사용 옵션을 선택합니다.
- 외부 파일의 상대 경로`./api-spec.yaml`로 입력



이렇게 하면 로컬 시스템에 다음 디렉터리가 생성됩니다.

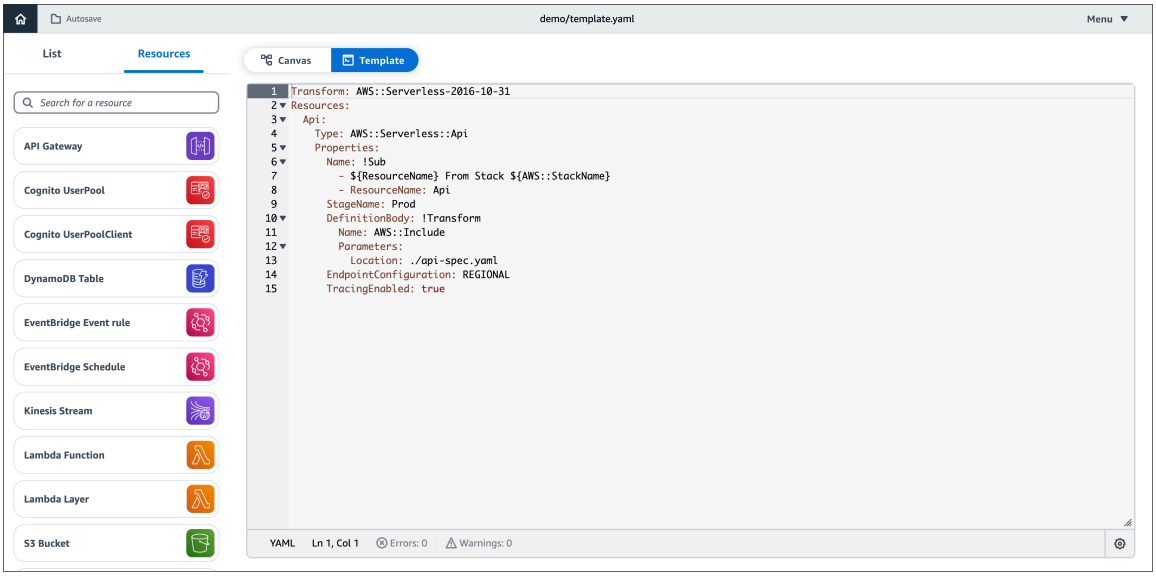
```
demo
### api-spec.yaml
```

이제 로컬 시스템에서 외부 파일을 구성할 수 있습니다. IDE를 사용하여 프로젝트 폴더에 `api-spec.yaml` 있는를 엽니다. 내용을 다음과 같이 바꿉니다.

```
openapi: '3.0'
info: {}
paths:
```

```
  /:
    get:
      responses: {}
    post:
      x-amazon-apigateway-integration:
        credentials:
          Fn::GetAtt:
            - ApiQueuesendmessageRole
            - Arn
        httpMethod: POST
        type: aws
        uri:
          Fn::Sub: arn:${AWS::Partition}:apigateway:${AWS::Region}:sqs:path/
${AWS::AccountId}/${Queue.QueueName}
        requestParameters:
          integration.request.header.Content-Type: '''application/x-www-form-
urlencoded'''
        requestTemplates:
          application/json: Action=SendMessage&MessageBody={"data":$input.body}
        responses:
          default:
            statusCode: 200
      responses:
        '200':
          description: 200 response
```

Infrastructure Composer 템플릿 보기에서 Infrastructure Composer가 외부 파일을 참조하도록 템플릿을 자동으로 업데이트했음을 확인할 수 있습니다.



Infrastructure Composer를 Amazon Virtual Private Cloud(Amazon VPC)와 통합

AWS Infrastructure Composer 는 Amazon Virtual Private Cloud(Amazon VPC) 서비스와의 통합을 지원합니다. Infrastructure Composer를 사용하여 다음을 수행할 수 있습니다.

- 시각적 VPC 태그를 통해 VPC에 있는 캔버스의 리소스를 식별합니다.
- 외부 템플릿에서 VPCs를 사용하여 AWS Lambda 함수를 구성합니다.

다음 이미지는 VPC로 구성된 Lambda 함수가 있는 애플리케이션의 예입니다.

Amazon VPC에 대해 자세히 알아보려면 Amazon VPC 사용 설명서의 Amazon VPC란 무엇입니까?를 참조하세요.

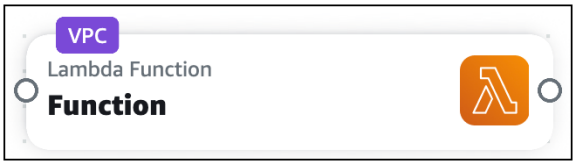
주제

- VPC에서 Infrastructure Composer 리소스 및 관련 정보 식별
- Infrastructure Composer에서 외부 VPCs 사용하여 Lambda 함수 구성
- Infrastructure Composer를 사용하는 외부 VPC에 대해 가져온 템플릿의 파라미터
- Infrastructure Composer를 사용하여 가져온 템플릿에 새 파라미터 추가
- Infrastructure Composer를 사용하여 Lambda 함수와 다른 템플릿에 정의된 VPC 구성

VPC에서 Infrastructure Composer 리소스 및 관련 정보 식별

Infrastructure Composer를 Amazon VPC와 통합하려면 먼저 VPC의 리소스와 통합을 완료하는 데 필요한 정보를 식별해야 합니다. 여기에는 보안 그룹, 서브넷 식별자, 파라미터 유형, SSM 유형, 정적 값 유형과 관련된 구성 정보도 포함됩니다.

Infrastructure Composer는 VPC 태그를 사용하여 VPC의 리소스를 시각화합니다. 이 태그는 캔버스의 카드에 적용됩니다. 다음은 VPC 태그가 있는 Lambda 함수의 예입니다.



VPC 태그는 다음을 수행할 때 캔버스의 카드에 적용됩니다.

- Infrastructure Composer에서 VPC를 사용하여 Lambda 함수를 구성합니다.
- VPC로 구성된 리소스가 포함된 템플릿을 가져옵니다.

보안 그룹 및 서브넷 식별자

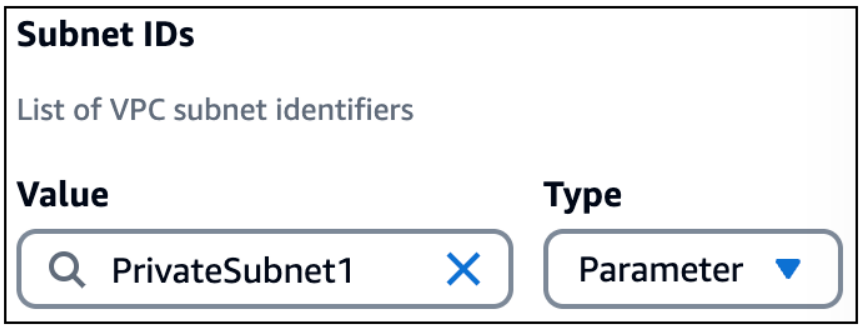
Lambda 함수는 여러 보안 그룹 및 서브넷으로 구성할 수 있습니다. Lambda 함수에 대한 보안 그룹 또는 서브넷을 구성하려면 값과 유형을 제공합니다.

- 값 - 보안 그룹 또는 서브넷의 식별자입니다. 허용되는 값은 유형에 따라 달라집니다.
- 유형 - 다음과 같은 유형의 값이 허용됩니다.
  - 파라미터 이름
  - AWS Systems Manager (SSM) 파라미터 스토어
  - 정적 값

파라미터 유형

AWS CloudFormation 템플릿의 `Parameters` 섹션을 사용하여 여러 템플릿에 리소스 정보를 저장할 수 있습니다. 파라미터에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 파라미터를 참조하세요.

파라미터 유형의 경우 파라미터 이름을 제공할 수 있습니다. 다음 예제에서는 `PrivateSubnet1` 파라미터 이름 값을 제공합니다.



파라미터 이름을 제공하면 Infrastructure Composer가 템플릿의 `Parameters` 섹션에서 파라미터 이름을 정의합니다. 그런 다음 Infrastructure Composer는 Lambda 함수 리소스의 파라미터를 참조합니다. 다음은 예제입니다.

```
...
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SubnetIds:
          - !Ref PrivateSubnet1
Parameters:
  PrivateSubnet1:
    Type: AWS::EC2::Subnet::Id
    Description: Parameter is generated by Infrastructure Composer
```

SSM 유형

SSM Parameter Store는 구성 데이터 관리 및 보안 암호 관리를 위한 안전한 계층적 스토리지를 제공합니다. 자세한 내용을 알아보려면 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store를 참조하세요.

SSM 유형의 경우 다음 값을 제공할 수 있습니다.

- SSM 파라미터 스토어의 값에 대한 동적 참조입니다.
- 템플릿에 정의된 `AWS::SSM::Parameter` 리소스의 논리적 ID입니다.

동적 참조

형식의 동적 참조를 사용하여 SSM 파라미터 스토어에서 값을 참조할 수 있습니다`{{resolve:ssm:reference-key}}`. 자세한 내용은 AWS CloudFormation 사용 설명서의 SSM 파라미터를 참조하세요.

Infrastructure Composer는 SSM 파라미터 스토어의 값으로 Lambda 함수를 구성하는 인프라 코드를 생성합니다. 다음은 예제입니다.

```
...
Resources:
  Function:
```

```
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds:
          - '{{resolve:ssm:demo-app/sg-0b61d5c742dc2c773}}'
  ...
```

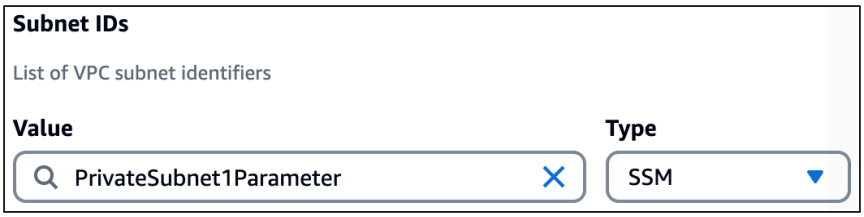
Logical ID

논리적 ID로 동일한 템플릿의 AWS::SSM::Parameter 리소스를 참조할 수 있습니다.

다음은의 서브넷 ID를 PrivateSubnet1Parameter 저장하는 라는 AWS::SSM::Parameter 리소스의 예입니다PrivateSubnet1.

```
...
Resources:
  PrivateSubnet1Parameter:
    Type: AWS::SSM::Parameter
    Properties:
      Name: /MyApp/VPC/SubnetIds
      Description: Subnet ID for PrivateSubnet1
      Type: String
      Value: subnet-04df123445678a036
```

다음은 Lambda 함수의 논리적 ID로 제공되는이 리소스 값의 예입니다.



Infrastructure Composer는 SSM 파라미터를 사용하여 Lambda 함수를 구성하는 인프라 코드를 생성합니다.

```
...
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
```

```
        SubnetIds:
          - !Ref PrivateSubnet1Parameter
  ...
  PrivateSubnet1Parameter:
    Type: AWS::SSM::Parameter
    Properties:
      ...
```

정적 값 유형

보안 그룹 또는 서브넷이 배포되면 CloudFormation ID 값이 생성됩니다. 이 ID를 정적 값으로 제공할 수 있습니다.

정적 값 유형의 경우 유효한 값은 다음과 같습니다.

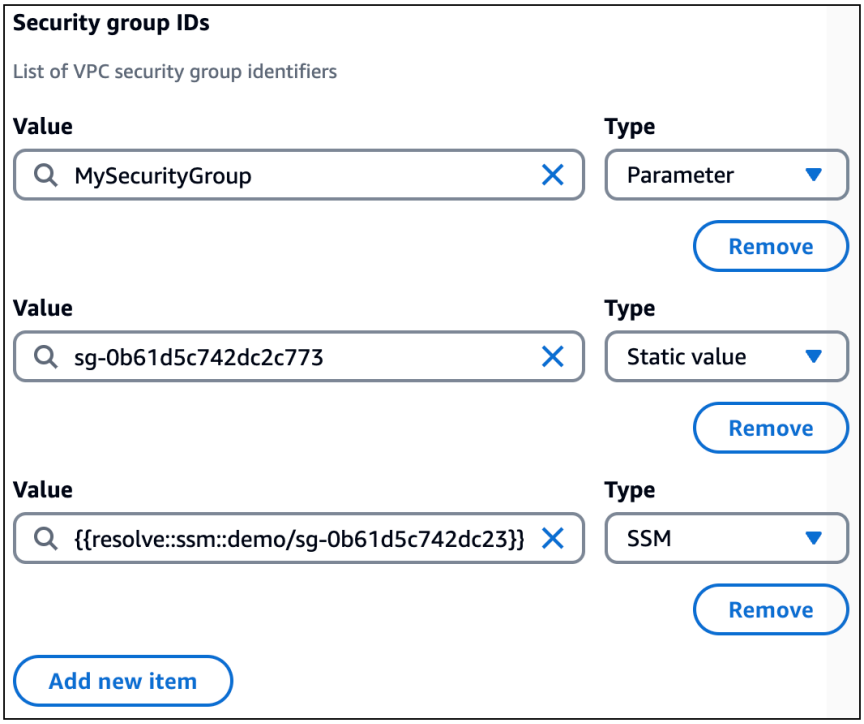
- 보안 그룹의 경우를 제공합니다`GroupId`. 자세한 내용은 AWS CloudFormation 사용 설명서의 값 반환을 참조하세요. 예를 들면, `sg-0b61d5c742dc2c773`입니다.
- 서브넷의 경우를 제공합니다`SubnetId`. 자세한 내용은 AWS CloudFormation 사용 설명서의 값 반환을 참조하세요. 예를 들면, `subnet-01234567890abcdef`입니다.

Infrastructure Composer는 정적 값으로 Lambda 함수를 구성하는 인프라 코드를 생성합니다. 다음은 예제입니다.

```
...
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds:
          - subnet-01234567890abcdef
        SubnetIds:
          - sg-0b61d5c742dc2c773
  ...
```

여러 유형 사용

보안 그룹 및 서브넷의 경우 여러 유형을 함께 사용할 수 있습니다. 다음은 다양한 유형의 값을 제공하여 Lambda 함수에 대한 세 가지 보안 그룹을 구성하는 예제입니다.



Infrastructure Composer는 `SecurityGroupIds` 속성 아래의 세 값을 모두 참조합니다.

```
...
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds:
          - !Ref MySecurityGroup
          - sg-0b61d5c742dc2c773
          - '{{resolve::ssm::demo/sg-0b61d5c742dc23}}'
      ...
Parameters:
  MySecurityGroup:
    Type: AWS::EC2::SecurityGroup::Id
    Description: Parameter is generated by Infrastructure Composer
```

Infrastructure Composer에서 외부 VPCs 사용하여 Lambda 함수 구성

다른 템플릿에 정의된 VPC로 Lambda 함수 구성을 시작하려면 Lambda 함수 향상된 구성 요소 카드를 사용합니다. 이 카드는 AWS Serverless Application Model (AWS SAM) `AWS::Serverless::Function` 리소스 유형을 사용하는 Lambda 함수를 나타냅니다.

외부 템플릿의 VPC를 사용하여 Lambda 함수를 구성하려면

1. Lambda 함수 리소스 속성 패널에서 VPC 설정(고급) 드롭다운 섹션을 확장합니다.
2. 외부 VPC에 할당을 선택합니다.
3. Lambda 함수에 대해 구성할 보안 그룹 및 서브넷의 값을 제공합니다. 세부 정보는 보안 그룹 및 서브넷 식별자 섹션을 참조하세요.
4. 변경 사항을 저장합니다.

Infrastructure Composer를 사용하는 외부 VPC에 대해 가져온 템플릿의 파라미터

외부 VPC의 보안 그룹 및 서브넷에 대해 정의된 파라미터가 있는 기존 템플릿을 가져올 때 Infrastructure Composer는 파라미터를 선택할 수 있는 드롭다운 목록을 제공합니다.

다음은 가져온 템플릿의 `Parameters` 섹션에 대한 예입니다.

```
...
Parameters:
  VPCSecurityGroups:
    Description: Security group IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::SecurityGroup::Id>
  VPCSubnets:
    Description: Subnet IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::Subnet::Id>
  VPCSubnet:
    Description: Subnet Id generated by Infrastructure Composer
    Type: AWS::EC2::Subnet::Id
...
```

캔버스에서 새 Lambda 함수에 대해 외부 VPC를 구성할 때 드롭다운 목록에서 이러한 파라미터를 사용할 수 있습니다. 다음은 예제입니다.

목록 파라미터 유형 가져오기 시 제한 사항

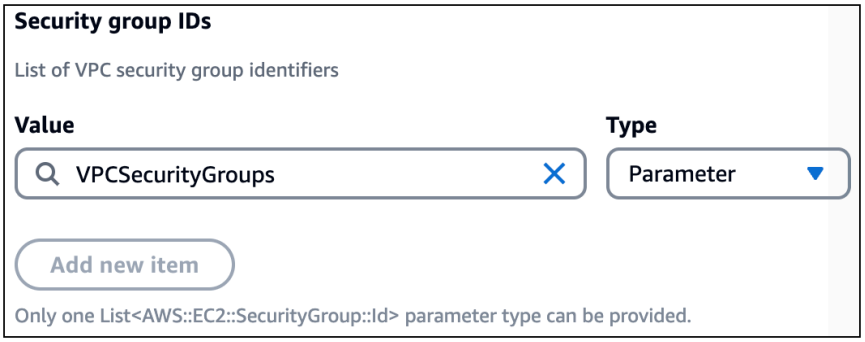
일반적으로 각 Lambda 함수에 대해 여러 보안 그룹 및 서브넷 식별자를 지정할 수 있습니다. 기존 템플릿에 `List<AWS::EC2::SecurityGroup::Id>` 또는와 같은 목록 파라미터 유형이 포함된 경우 식별자를 하나만 지정할 `List<AWS::EC2::Subnet::Id>`수 있습니다.

파라미터 목록 유형에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 지원되는 AWS특정 파라미터 유형을 참조하세요.

다음은를 목록 파라미터 유형`VPCSecurityGroups`으로 정의하는 템플릿의 예입니다.

```
...
Parameters:
  VPCSecurityGroups:
    Description: Security group IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::SecurityGroup::Id>
...
```

Infrastructure Composer에서 `VPCSecurityGroups` 값을 Lambda 함수의 보안 그룹 식별자로 선택하면 다음 메시지가 표시됩니다.



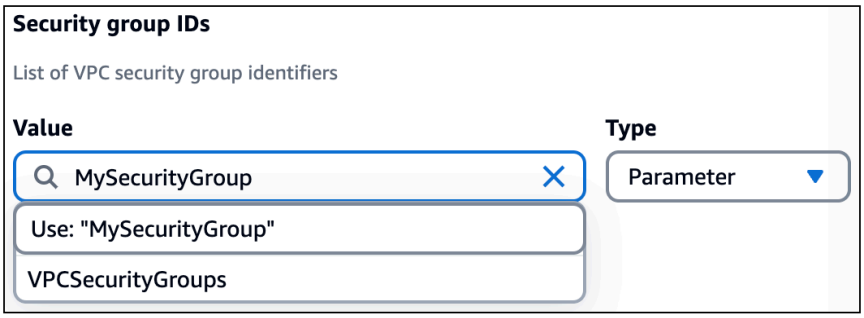
이 제한은 `AWS::Lambda::Function VpcConfig` 객체의 `SecurityGroupIds` 및 `SubnetIds` 속성 모두 문자열 값 목록만 허용하기 때문에 발생합니다. 단일 목록 파라미터 유형에는 문자열 목록이 포함되어 있으므로 지정 시 제공되는 유일한 객체일 수 있습니다.

목록 파라미터 유형의 경우 다음은 Lambda 함수로 구성된 경우 템플릿에 정의된 방법의 예입니다.

```
...
Parameters:
  VPCSecurityGroups:
    Description: Security group IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::SecurityGroup::Id>
  VPCSubnets:
    Description: Subnet IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::Subnet::Id>
Resources:
  ...
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds: !Ref VPCSecurityGroups
        SubnetIds: !Ref VPCSubnets
```

Infrastructure Composer를 사용하여 가져온 템플릿에 새 파라미터 추가

파라미터가 정의된 기존 템플릿을 가져올 때 새 파라미터를 생성할 수도 있습니다. 드롭다운 목록에서 기존 파라미터를 선택하는 대신 새 유형과 값을 제공합니다. 다음은 라는 새 파라미터를 생성하는 예제입니다`MySecurityGroup`.



Lambda 함수의 리소스 속성 패널에서 제공하는 모든 새 값에 대해 Infrastructure Composer는 Lambda 함수의 `SecurityGroupIds` 또는 `SubnetIds` 속성 아래의 목록에서 해당 값을 정의합니다. 다음은 예제입니다.

```
...
Resources:
  MyFunction:
```

```
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds:
          - sg-94b3a1f6
        SubnetIds:
          - !Ref SubnetParameter
          - !Ref VPCSubnet
```

외부 템플릿에서 목록 파라미터 유형의 논리적 ID를 참조하려면 템플릿 보기를 사용하고 템플릿을 직접 수정하는 것이 좋습니다. 목록 파라미터 유형의 논리적 ID는 항상 단일 값으로 제공되고 유일한 값으로 제공되어야 합니다.

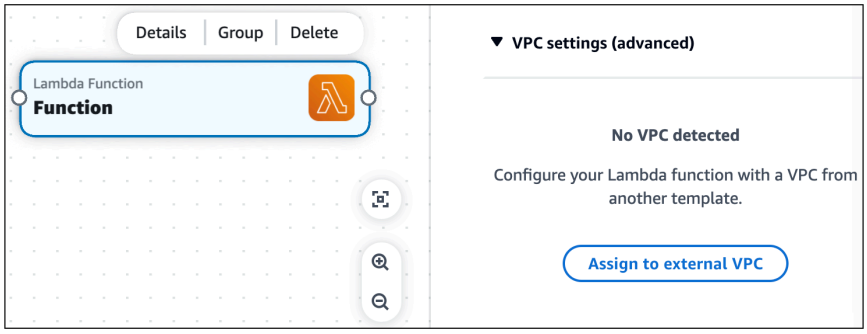
```
...
Parameters:
  VPCSecurityGroups:
    Description: Security group IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::SecurityGroup::Id>
  VPCSubnets:
    Description: Subnet IDs generated by Infrastructure Composer
    Type: List<AWS::EC2::Subnet::Id>
Resources:
  ...
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      ...
      VpcConfig:
        SecurityGroupIds: !Ref VPCSecurityGroups # Valid syntax
        SubnetIds:
          - !Ref VPCSubnets # Not valid syntax
```

Infrastructure Composer를 사용하여 Lambda 함수와 다른 템플릿에 정의된 VPC 구성

이 예제에서는 다른 템플릿에 정의된 VPC를 사용하여 Infrastructure Composer에서 Lambda 함수를 구성합니다.

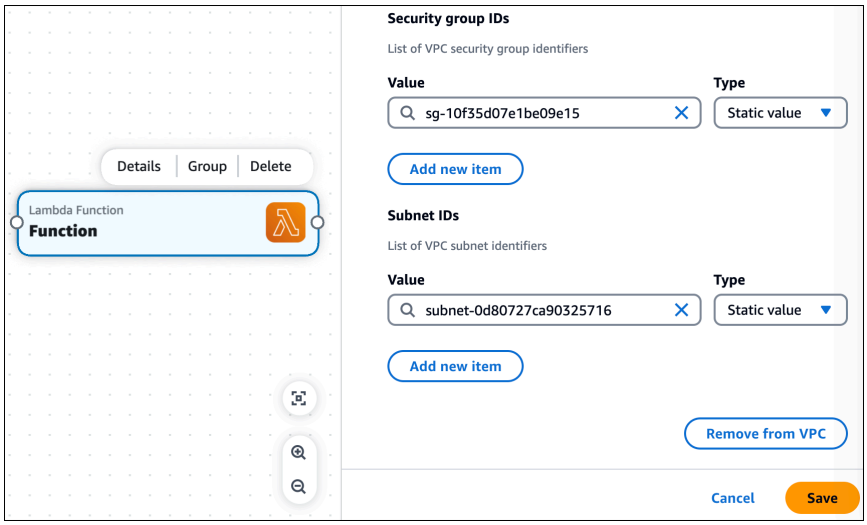
먼저 Lambda 함수 향상된 구성 요소 카드를 캔버스로 드래그합니다.

다음으로 카드의 리소스 속성 패널을 열고 VPC 설정(고급) 드롭다운 섹션을 확장합니다.

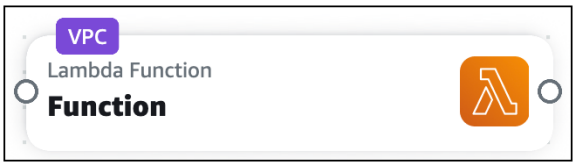


다음으로 외부 VPC에 할당을 선택하여 외부 템플릿에서 VPC 구성을 시작합니다.

이 예제에서는 보안 그룹 ID와 서브넷 ID를 참조합니다. 이러한 값은 VPC를 정의하는 템플릿이 배포될 때 생성됩니다. 정적 값 유형을 선택하고 IDs 값을 입력합니다. 완료되면 저장을 선택합니다.



이제 Lambda 함수가 VPC로 구성되었으므로 VPC 태그가 카드에 표시됩니다.



Infrastructure Composer는 외부 VPC의 보안 그룹 및 서브넷으로 Lambda 함수를 구성하는 인프라 코드를 생성했습니다.

```
Transform: AWS::Serverless-2016-10-31
Resources:
  Function:
    Type: AWS::Serverless::Function
    Properties:
      Description: !Sub
        - Stack ${AWS::StackName} Function ${ResourceName}
        - ResourceName: Function
      CodeUri: src/Function
      Handler: index.handler
      Runtime: nodejs18.x
      MemorySize: 3008
      Timeout: 30
      Tracing: Active
      VpcConfig:
        SecurityGroupIds:
          - sg-10f35d07e1be09e15
        SubnetIds:
          - subnet-0d80727ca90325716
  FunctionLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    Properties:
      LogGroupName: !Sub /aws/lambda/${Function}
```

Infrastructure Composer 서버리스 애플리케이션을 AWS 클라우드에 배포

AWS Infrastructure Composer 를 사용하여 배포 가능한 서버리스 애플리케이션을 설계합니다. 배포하려면 AWS CloudFormation 호환되는 서비스를 사용합니다. AWS Serverless Application Model (AWS SAM)를 사용하는 것이 좋습니다.

AWS SAM 는 서버리스 애플리케이션을 구축하고 실행하기 위한 개발자 도구를 제공하는 오픈 소스 프레임워크입니다 AWS. 개발자는 AWS SAM의 간편 구문을 사용하여 배포 중에 인프라로 변환되는 CloudFormation 리소스와 특수 서버리스 리소스를 선언합니다.

중요 AWS SAM 개념

사용하기 전에 몇 가지 기본 개념에 익숙해지는 AWS SAM것이 중요합니다.

- AWS SAM 작동 방식: AWS Serverless Application Model 개발자 안내서의이 주제에서는 서비스 없는 애플리케이션을 생성하는 데 사용하는 기본 구성 요소인 AWS SAM CLI, AWS SAM 프로젝트 및 AWS SAM 템플릿에 대한 중요한 정보를 제공합니다.
- 사용 방법 AWS Serverless Application Model (AWS SAM): AWS Serverless Application Model 개발자 안내서에 있는이 주제에서는 애플리케이션을 AWS 클라우드에 배포하는 AWS SAM 데 사용하기 위해 완료해야 하는 단계에 대한 개략적인 개요를 제공합니다.

Infrastructure Composer에서 애플리케이션을 설계할 때 sam sync 명령을 사용하여가 로컬 변경 사항을 AWS SAM CLI 자동으로 감지하고 해당 변경 사항을에 배포하도록 할 수 있습니다 CloudFormation. 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 sam sync 사용을 참조하세요.

다음 단계

애플리케이션 배포를 준비및 Infrastructure Composer를 AWS SAM CLI 사용하여 배포하도록 설정하려면 섹션을 참조하세요.

및 Infrastructure Composer를 AWS SAM CLI 사용하여 배포하도록 설정

를 사용하여 애플리케이션을 배포하려면 AWS SAM먼저 및를 AWS CLI 설치하고 액세스해야 합니다 AWS SAM CLI. 이 섹션의 주제에서는 이에 대한 세부 정보를 제공합니다.

AWS CLI 설치

를 AWS CLI 설치하기 전에를 설치하고 설정하는 것이 좋습니다 AWS SAM CLI. 지침은 AWS Command Line Interface 사용 설명서의 의 최신 버전 설치 또는 업데이트를 AWS CLI 참조하세요.

> **Note**
>
> 를 설치 AWS CLI한 후 AWS 자격 증명을 구성해야 합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 빠른 설정을 참조하세요.

AWS SAM CLI 설치

를 설치하려면 AWS Serverless Application Model 개발자 안내서의 설치를 AWS SAM CLI AWS SAM CLI참조하세요.

에 액세스 AWS SAM CLI

에서 Infrastructure Composer를 사용하는 경우를 사용할 AWS Management Console수 있는 다음과 같은 옵션이 있습니다 AWS SAM CLI.

로컬 동기화 모드 활성화

로컬 동기화 모드를 사용하면 AWS SAM 템플릿을 포함한 프로젝트 폴더가 로컬 시스템에 자동으로 저장됩니다. Infrastructure Composer는가 AWS SAM 인식하는 방식으로 프로젝트 디렉터리를 구성합니다. 프로젝트의 루트 디렉터리에서를 실행할 AWS SAM CLI 수 있습니다.

로컬 동기화 모드에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer 콘솔에서 프로젝트를 로컬로 동기화하고 저장.

템플릿 내보내기

템플릿을 로컬 시스템으로 내보낼 수 있습니다. 그런 다음 템플릿이 포함된 상위 폴더에서를 실행합니다 AWS SAM CLI. `--template-file` 옵션을 모든 AWS SAM CLI 명령과 함께 사용하고 템플릿 경로를 제공할 수도 있습니다.

에서 Infrastructure Composer 사용 AWS Toolkit for Visual Studio Code

Toolkit for VS Code에서 Infrastructure Composer를 사용하여 Infrastructure Composer를 로컬 시스템으로 가져올 수 있습니다. 그런 다음 Infrastructure Composer와 VS Code의를 AWS SAM CLI 사용합니다.

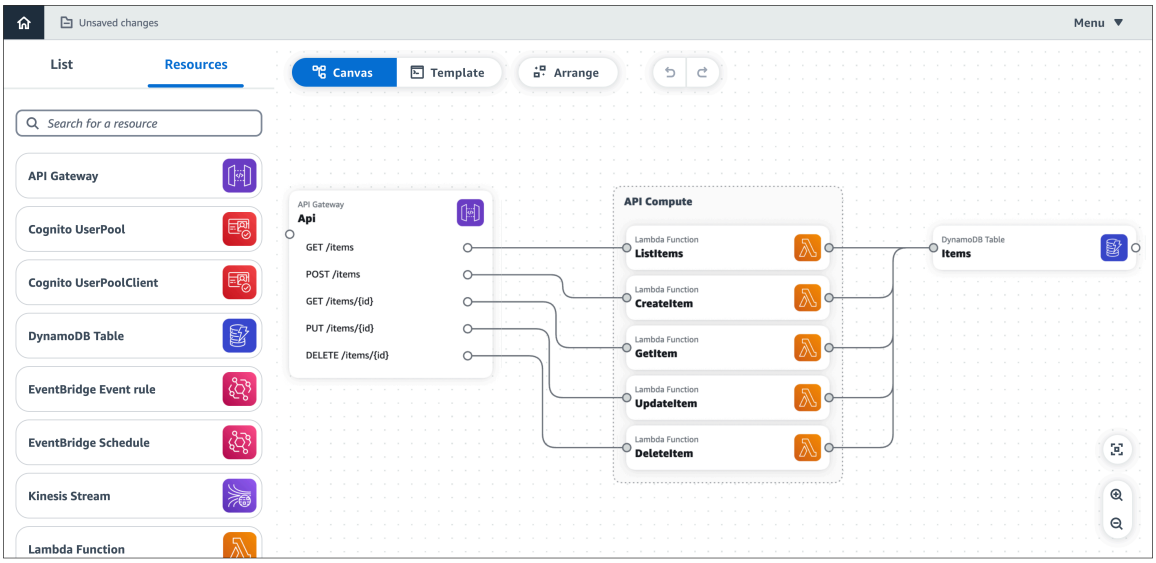
다음 단계

애플리케이션을 배포하려면 섹션을 참조하세요와 함께 Infrastructure Composer AWS SAM 를 사용하여 빌드 및 배포.

와 함께 Infrastructure Composer AWS SAM 를 사용하여 빌드 및 배포

이제를 완료했으므로 AWS SAM 및 Infrastructure Composer를 사용하여 애플리케이션을 배포할 및 Infrastructure Composer를 AWS SAM CLI 사용하여 배포하도록 설정수 있습니다. 이 섹션에서는 이를 수행하는 방법을 자세히 설명하는 예제를 제공합니다. 를 사용하여 애플리케이션을 배포하는 방법에 대한 지침은 개발자 안내서의를 사용하여 애플리케이션 및 리소스 AWS SAM 배포를 참조할 수도 있습니다 AWS SAM. AWS Serverless Application Model

이 예제에서는 Infrastructure Composer 데모 애플리케이션을 빌드하고 배포하는 방법을 보여줍니다. 데모 애플리케이션에는 다음과 같은 리소스가 있습니다.



> **Note**
>
> - 데모 애플리케이션에 대한 자세한 내용은 섹션을 참조하세요Infrastructure Composer 데모 프로젝트 로드 및 수정.
> - 이 예제에서는 로컬 동기화가 활성화된 Infrastructure Composer를 사용합니다.

1. sam build 명령을 사용하여 애플리케이션을 빌드합니다.

```
$ sam build
...
Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=========================
[*] Validate SAM template: sam validate
[*] Invoke Function: sam local invoke
[*] Test Function in the Cloud: sam sync --stack-name {{stack-name}} --watch
[*] Deploy: sam deploy --guided
```

는 AWS SAM CLI 프로젝트 폴더에 `./aws-sam` 디렉터리를 생성합니다. 이 디렉터리에는 애플리케이션의 Lambda 함수에 대한 빌드 아티팩트가 포함되어 있습니다. 다음은 프로젝트 디렉터리의 출력입니다.

```
.
### README.md
### samconfig.toml
### src
#   ### CreateItem
#   #   ### index.js
#   #   ### package.json
#   ### DeleteItem
#   #   ### index.js
#   #   ### package.json
#   ### GetItem
#   #   ### index.js
#   #   ### package.json
#   ### ListItems
#   #   ### index.js
#   #   ### package.json
#   ### UpdateItem
#       ### index.js
#       ### package.json
### template.yaml
```

2. 이제 애플리케이션을 배포할 준비가 되었습니다. 를 사용합니다`sam deploy --guided`. 이렇게 하면 일련의 프롬프트를 통해 애플리케이션을 배포할 준비가 됩니다.

```
$ sam deploy --guided
...
Configuring SAM deploy
======================

    Looking for config file [samconfig.toml] :  Found
    Reading default arguments  :  Success

    Setting default arguments for 'sam deploy'
    =========================================
    Stack Name [aws-app-composer-basic-api]: AWS Region [us-west-2]:
    #Shows you resources changes to be deployed and require a 'Y' to initiate
 deploy
    Confirm changes before deploy [y/N]:
```

```
    #SAM needs permission to be able to create roles to connect to the resources in
 your template
    Allow SAM CLI IAM role creation [Y/n]:
    #Preserves the state of previously provisioned resources when an operation
 fails
    Disable rollback [y/N]:
    ListItems may not have authorization defined, Is this okay? [y/N]: y
    CreateItem may not have authorization defined, Is this okay? [y/N]: y
    GetItem may not have authorization defined, Is this okay? [y/N]: y
    UpdateItem may not have authorization defined, Is this okay? [y/N]: y
    DeleteItem may not have authorization defined, Is this okay? [y/N]: y
    Save arguments to configuration file [Y/n]:
    SAM configuration file [samconfig.toml]:
    SAM configuration environment [default]:
```

에는 AWS SAM CLI 배포할 항목에 대한 요약이 표시됩니다.

```
Deploying with following values
    ===============================
    Stack name                   : aws-app-composer-basic-api
    Region                       : us-west-2
    Confirm changeset            : False
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-
demo-1b3x26zbcdkqr
    Capabilities                 : ["CAPABILITY_IAM"]
    Parameter overrides          : {}
    Signing Profiles             : {}
```

는 AWS SAM CLI 먼저 CloudFormation 변경 세트를 생성하여 애플리케이션을 배포합니다.

```
Initiating deployment
=====================
Uploading to aws-app-composer-basic-api/4181c909ee2440a728a7a129dafb83d4.template
 7087 / 7087  (100.00%)

Waiting for changeset to be created..
CloudFormation stack changeset
-----------------------------------------------------------------------------------------
Operation                               LogicalResourceId
 ResourceType                            Replacement
-----------------------------------------------------------------------------------------
```

```
+ Add                                ApiDeploymentcc153d135b
 AWS::ApiGateway::Deployment          N/A
+ Add                                ApiProdStage
 AWS::ApiGateway::Stage               N/A
+ Add                                Api
 AWS::ApiGateway::RestApi             N/A
+ Add                                CreateItemApiPOSTitemsPermissionP
 AWS::Lambda::Permission              N/A
                                     rod
+ Add                                CreateItemRole
 AWS::IAM::Role                       N/A
+ Add                                CreateItem
 AWS::Lambda::Function                N/A
+ Add                                DeleteItemApiDELETEitemsidPermiss
 AWS::Lambda::Permission              N/A
                                     ionProd
+ Add                                DeleteItemRole
 AWS::IAM::Role                       N/A
+ Add                                DeleteItem
 AWS::Lambda::Function                N/A
+ Add                                GetItemApiGETitemsidPermissionPro
 AWS::Lambda::Permission              N/A
                                     d
+ Add                                GetItemRole
 AWS::IAM::Role                       N/A
+ Add                                GetItem
 AWS::Lambda::Function                N/A
+ Add                                Items
 AWS::DynamoDB::Table                 N/A
+ Add                                ListItemsApiGETitemsPermissionPro
 AWS::Lambda::Permission              N/A
                                     d
+ Add                                ListItemsRole
 AWS::IAM::Role                       N/A
+ Add                                ListItems
 AWS::Lambda::Function                N/A
+ Add                                UpdateItemApiPUTitemsidPermission
 AWS::Lambda::Permission              N/A
                                     Prod
+ Add                                UpdateItemRole
 AWS::IAM::Role                       N/A
+ Add                                UpdateItem
 AWS::Lambda::Function                N/A
-------------------------------------------------------------------------------------------
```

```
Changeset created successfully. arn:aws:cloudformation:us-
west-2:513423067560:changeSet/samcli-deploy1677472539/967ab543-f916-4170-b97d-
c11a6f9308ea
```

그런 다음는 AWS SAM CLI 애플리케이션을 배포합니다.

```
CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------
ResourceStatus                      ResourceType
 LogicalResourceId                   ResourceStatusReason
---------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS                  AWS::DynamoDB::Table               Items
                          -
CREATE_IN_PROGRESS                  AWS::DynamoDB::Table               Items
                          Resource creation Initiated
CREATE_COMPLETE                     AWS::DynamoDB::Table               Items
                          -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 DeleteItemRole                      -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 ListItemsRole                       -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 UpdateItemRole                      -
CREATE_IN_PROGRESS                  AWS::IAM::Role                     GetItemRole
                          -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 CreateItemRole                      -
CREATE_IN_PROGRESS                  AWS::IAM::Role
 DeleteItemRole                      Resource creation Initiated
CREATE_IN_PROGRESS                  AWS::IAM::Role
 ListItemsRole                       Resource creation Initiated
CREATE_IN_PROGRESS                  AWS::IAM::Role                     GetItemRole
                          Resource creation Initiated
CREATE_IN_PROGRESS                  AWS::IAM::Role
 UpdateItemRole                      Resource creation Initiated
CREATE_IN_PROGRESS                  AWS::IAM::Role
 CreateItemRole                      Resource creation Initiated
CREATE_COMPLETE                     AWS::IAM::Role
 DeleteItemRole                      -
CREATE_COMPLETE                     AWS::IAM::Role
 ListItemsRole                       -
```

```
CREATE_COMPLETE                    AWS::IAM::Role                    GetItemRole
                        -
CREATE_COMPLETE                    AWS::IAM::Role
 UpdateItemRole                     -
CREATE_COMPLETE                    AWS::IAM::Role
 CreateItemRole                     -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             DeleteItem
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             CreateItem
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             ListItems
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             UpdateItem
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             DeleteItem
                       Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::Lambda::Function             GetItem
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Function             ListItems
                       Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::Lambda::Function             CreateItem
                       Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::Lambda::Function             UpdateItem
                       Resource creation Initiated
CREATE_IN_PROGRESS                 AWS::Lambda::Function             GetItem
                       Resource creation Initiated
CREATE_COMPLETE                    AWS::Lambda::Function             DeleteItem
                        -
CREATE_COMPLETE                    AWS::Lambda::Function             ListItems
                        -
CREATE_COMPLETE                    AWS::Lambda::Function             CreateItem
                        -
CREATE_COMPLETE                    AWS::Lambda::Function             UpdateItem
                        -
CREATE_COMPLETE                    AWS::Lambda::Function             GetItem
                        -
CREATE_IN_PROGRESS                 AWS::ApiGateway::RestApi          Api
                        -
CREATE_IN_PROGRESS                 AWS::ApiGateway::RestApi          Api
                       Resource creation Initiated
CREATE_COMPLETE                    AWS::ApiGateway::RestApi          Api
                        -
CREATE_IN_PROGRESS                 AWS::Lambda::Permission
 GetItemApiGETitemsidPermissionPro   -
```

```
                                                                    d
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 ListItemsApiGETitemsPermissionPro   -
                                                                    d
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 DeleteItemApiDELETEitemsidPermiss   -
                                                                    ionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ApiDeploymentcc153d135b             -
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 UpdateItemApiPUTitemsidPermission   -
                                                                    Prod
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 CreateItemApiPOSTitemsPermissionP   -
                                                                    rod
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 GetItemApiGETitemsidPermissionPro   Resource creation Initiated
                                                                    d
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 UpdateItemApiPUTitemsidPermission   Resource creation Initiated
                                                                    Prod
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 CreateItemApiPOSTitemsPermissionP   Resource creation Initiated
                                                                    rod
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 ListItemsApiGETitemsPermissionPro   Resource creation Initiated
                                                                    d
CREATE_IN_PROGRESS                  AWS::Lambda::Permission
 DeleteItemApiDELETEitemsidPermiss   Resource creation Initiated
                                                                    ionProd
CREATE_IN_PROGRESS                  AWS::ApiGateway::Deployment
 ApiDeploymentcc153d135b             Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::Deployment
 ApiDeploymentcc153d135b             -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ApiProdStage                        -
CREATE_IN_PROGRESS                  AWS::ApiGateway::Stage
 ApiProdStage                        Resource creation Initiated
CREATE_COMPLETE                     AWS::ApiGateway::Stage
 ApiProdStage                        -
CREATE_COMPLETE                     AWS::Lambda::Permission
 CreateItemApiPOSTitemsPermissionP   -
                                                                    rod
```

```
CREATE_COMPLETE                     AWS::Lambda::Permission
 UpdateItemApiPUTitemsidPermission   -
                                                                    Prod
CREATE_COMPLETE                     AWS::Lambda::Permission
 ListItemsApiGETitemsPermissionPro   -
                                                                    d
CREATE_COMPLETE                     AWS::Lambda::Permission
 DeleteItemApiDELETEitemsidPermiss   -
                                                                    ionProd
CREATE_COMPLETE                     AWS::Lambda::Permission
 GetItemApiGETitemsidPermissionPro   -
                                                                    d
CREATE_COMPLETE                     AWS::CloudFormation::Stack      aws-app-
composer-basic-api          -
-------------------------------------------------------------------------------------------------
```

마지막으로 배포에 성공했음을 알리는 메시지가 표시됩니다.

```
Successfully created/updated stack - aws-app-composer-basic-api in us-west-2
```

에서 Infrastructure Composer AWS SAM 를 사용하여 스택 삭제

이 예제에서는 sam delete 명령을 사용하여 CloudFormation 스택을 삭제하는 방법을 보여줍니다.

sam delete에 AWS SAM CLI 명령을 입력하고 스택과 템플릿을 삭제할지 확인합니다.

```
$ sam delete
 Are you sure you want to delete the stack aws-app-composer-basic-api in the region us-
west-2 ? [y/N]: y
 Do you want to delete the template file 30439348c0be6e1b85043b7a935b34ab.template in
 S3? [y/N]: y
 - Deleting S3 object with key eb226ca86d1bc4e9914ad85eb485fed8
 - Deleting S3 object with key 875e4bcf4b10a6a1144ad83158d84b6d
 - Deleting S3 object with key 20b869d98d61746dedd9aa33aa08a6fb
 - Deleting S3 object with key c513cedc4db6bc184ce30e94602741d6
 - Deleting S3 object with key c7a15d7d8d1c24b77a1eddf8caebc665
 - Deleting S3 object with key e8b8984f881c3732bfb34257cdd58f1e
 - Deleting S3 object with key 3185c59b550594ee7fca7f8c36686119.template
 - Deleting S3 object with key 30439348c0be6e1b85043b7a935b34ab.template
 - Deleting Cloudformation stack aws-app-composer-basic-api
```

```
Deleted successfully
```

AWS Infrastructure Composer 문제 해결

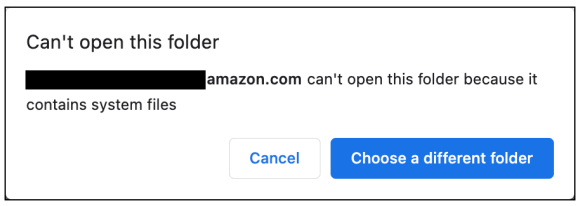
이 섹션의 주제에서는 사용 시 오류 메시지 문제 해결에 대한 지침을 제공합니다 AWS Infrastructure Composer.

주제

오류 메시지

"이 폴더를 열 수 없습니다"

오류 메시지 예:



가능한 원인: Infrastructure Composer가 로컬 동기화 모드를 사용하여 민감한 디렉터리에 액세스할 수 없습니다.

> 이 오류에 대한 자세한 내용은 섹션을 참조하세요Data Infrastructure Composer는에 액세스할 수 있습니다..
>
> 다른 로컬 디렉터리에 연결하거나 로컬 동기화가 비활성화된 Infrastructure Composer를 사용해 보세요.

“호환되지 않는 템플릿”

예제 오류: Infrastructure Composer에서 새 프로젝트를 로드할 때 다음이 표시됩니다.

가능한 원인: 프로젝트에 Infrastructure Composer에서 지원되지 않는 외부 참조 파일이 포함되어 있습니다.

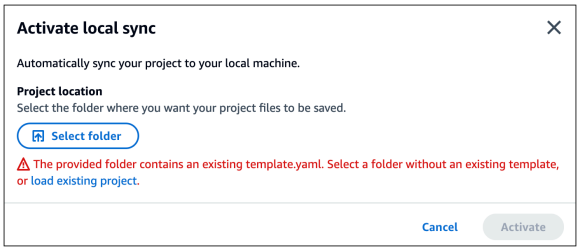
> Infrastructure Composer에서 지원되는 외부 파일에 대한 자세한 내용은 섹션을 참조하세요외부 파일 참조.

가능한 원인: 프로젝트가 다른 로컬 디렉터리의 외부 파일에 연결됩니다.

외부에서 참조한 파일을 Infrastructure Composer 로컬 동기화 모드와 함께 사용하도록 선택한 디렉터리의 하위 디렉터리로 이동합니다.

“제공된 폴더에 기존 template.yaml이 포함되어 있습니다.”

로컬 동기화를 활성화하려고 하면 다음 오류가 표시됩니다.



가능한 원인: 선택한 폴더에 template.yaml 파일이 이미 포함되어 있습니다.

애플리케이션 템플릿이 포함되지 않은 다른 디렉터리를 선택하거나 새 디렉터리를 생성합니다.

"브라우저에 해당 폴더에 프로젝트를 저장할 수 있는 권한이 없습니다..."

가능한 원인: Infrastructure Composer가 로컬 동기화 모드를 사용하여 민감한 디렉터리에 액세스할 수 없습니다.

이 오류에 대한 자세한 내용은 섹션을 참조하세요Data Infrastructure Composer는에 액세스할 수 있습니다..

다른 로컬 디렉터리에 연결하거나 로컬 동기화가 비활성화된 Infrastructure Composer를 사용하세요.

의 보안 AWS Infrastructure Composer

의 클라우드 보안 AWS 이 최우선 순위입니다. 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 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 제공 범위 내 서비스를 AWS Infrastructure Composer참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Infrastructure Composer를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 Infrastructure Composer를 구성하는 방법을 보여줍니다. 또한 Infrastructure Composer 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- 의 데이터 보호 AWS Infrastructure Composer
- AWS Identity and Access Management 용 AWS Infrastructure Composer
- 에 대한 규정 준수 검증 AWS Infrastructure Composer
- 의 복원력 AWS Infrastructure Composer

의 데이터 보호 AWS Infrastructure Composer

AWS 공동 책임 모델 AWS Infrastructure Composer의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은

데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Infrastructure Composer 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

> **Note**
>
> Infrastructure Composer에 입력하는 모든 데이터는 Infrastructure Composer 내에서 기능을 제공하고 시스템에 로컬로 저장된 프로젝트 파일 및 디렉터리를 생성하는 목적으로만 사용됩니다. Infrastructure Composer는이 데이터를 저장, 저장 또는 전송하지 않습니다.

데이터 암호화

Infrastructure Composer는 데이터가 저장, 저장 또는 전송되지 않으므로 고객 콘텐츠를 암호화하지 않습니다.

저장 시 암호화

Infrastructure Composer는 데이터가 저장, 저장 또는 전송되지 않으므로 고객 콘텐츠를 암호화하지 않습니다.

전송 중 암호화

Infrastructure Composer는 데이터가 저장, 저장 또는 전송되지 않으므로 고객 콘텐츠를 암호화하지 않습니다.

키 관리

Infrastructure Composer는 고객 콘텐츠가 저장, 저장 또는 전송되지 않으므로 키 관리를 지원하지 않습니다.

인터네트워크 트래픽 개인 정보 보호

Infrastructure Composer는 온프레미스 클라이언트 및 애플리케이션을 사용하여 트래픽을 생성하지 않습니다.

AWS Identity and Access Management 용 AWS Infrastructure Composer

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 Infrastructure Composer 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

- AWS Infrastructure Composer 에서 IAM을 사용하는 방법

대상

Infrastructure Composer에는 최소한에 대한 읽기 전용 액세스 권한이 필요합니다 AWS Management Console. 이 권한이 있는 모든 사용자는 Infrastructure Composer의 모든 기능을 사용할 수 있습니다. Infrastructure Composer의 특정 기능에 대한 세분화된 액세스는 지원되지 않습니다.

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 사용자에게 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 사용하여에 Directory Service 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.

액세스 제어 목록(ACL)

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.

Amazon S3 AWS WAF 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

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.

AWS Infrastructure Composer 에서 IAM을 사용하는 방법

AWS Infrastructure Composer 에서는 최소한에 대한 읽기 전용 액세스 권한이 필요합니다 AWS Management Console. 이 권한이 있는 모든 사용자는 Infrastructure Composer의 모든 기능을 사용할 수 있습니다. Infrastructure Composer의 특정 기능에 대한 세분화된 액세스는 지원되지 않습니다.

프로젝트 템플릿과 파일을에 배포할 때 AWS CloudFormation필요한 권한이 있어야 합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management를 사용한 액세스 제어를 참조하세요.

다음 표에는 사용할 수 있는 IAM 기능이 나와 있습니다 AWS Infrastructure Composer.

IAM 특성	Infrastructure Composer 지원
자격 증명 기반 정책	아니요
리소스 기반 정책	아니요
정책 작업	아니요
정책 리소스	아니요
정책 조건 키	아니요
ACL	아니요
ABAC(정책 내 태그)	아니요
임시 보안 인증	예
위탁자 권한	아니요
서비스 역할	아니요

IAM 특성	Infrastructure Composer 지원
서비스 연결 역할	아니요

Infrastructure Composer 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Infrastructure Composer의 자격 증명 기반 정책

자격 증명 기반 정책 지원: 아니요

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.

Infrastructure Composer 내의 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 또는이 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.

Infrastructure Composer에 대한 정책 작업

정책 작업 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.

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.

Infrastructure Composer 작업 목록을 보려면 서비스 승인 참조의 AWS Infrastructure Composer에서 정의한 작업을 참조하세요.

Infrastructure Composer에 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": "*"
```

Infrastructure Composer 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의 AWS Infrastructure Composer에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Infrastructure Composer에서 정의한 작업을 참조하세요.

Infrastructure Composer에 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.

Infrastructure Composer 조건 키 목록을 보려면 서비스 승인 참조의 AWS Infrastructure Composer에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 AWS Infrastructure Composer에서 정의한 작업을 참조하세요.

Infrastructure ComposerACLs

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.

Infrastructure Composer를 사용한 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.

Infrastructure Composer에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 스위치 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 작업하는AWS 서비스 섹션을 참조하세요.

임시 자격 증명을 사용하여를 통해 Infrastructure Composer에 액세스할 수 있습니다 AWS Management Console. 예제는 IAM 사용 설명서의 AWS 콘솔에 대한 사용자 지정 자격 증명 브로커 액세스 활성화를 참조하세요.

Infrastructure Composer에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 아니요

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

Infrastructure Composer의 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 Infrastructure Composer 기능이 중단될 수 있습니다. Infrastructure Composer가 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Infrastructure Composer의 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 `Yes`가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

에 대한 규정 준수 검증 AWS Infrastructure Composer

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 제공 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

의 복원력 AWS Infrastructure Composer

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 많으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Infrastructure Composer에 입력하는 모든 데이터는 Infrastructure Composer 내에서 기능을 제공하고 시스템에 로컬로 저장된 프로젝트 파일 및 디렉터리를 생성하는 목적으로만 사용됩니다. Infrastructure Composer는이 데이터를 저장하거나 저장하지 않습니다.

Infrastructure Composer의 문서 기록

다음 표에서는 Infrastructure Composer의 중요한 설명서 릴리스를 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최신 설명서 업데이트: 2023년 11월 30일

변경 사항	설명	날짜
개발자 안내서 전체에서 콘텐츠 재구성 및 업데이트	검색 편의성과 사용성을 개선하기 위해 안내서를 개편하고 재구성했습니다. 제목을 업데이트하고 개선했습니다. 주제 및 개념을 소개할 때 세부 정보를 추가했습니다.	2024년 8월 1일
CloudFormation 콘솔 모드에서 Infrastructure Composer를 사용하기 위한 설명서를 추가하고 Infrastructure Composer 개발자 안내서를 재구성했습니다.	AWS Infrastructure Composer 이제 CloudFormation 콘솔 모드에서를 사용할 수 있습니다. 자세한 내용은 CloudFormation 콘솔 모드에서 Infrastructure Composer 사용을 참조하세요. 또한 사용 설명서의 대부분의 콘텐츠가 간소화된 환경을 만들기 위해 재구성되었습니다.	2024년 3월 28일
CodeWhisperer와의 Infrastructure Composer 통합에 대한 설명서 추가	AWS Infrastructure Composer Toolkit for VS Code의는 Amazon CodeWhisperer와의 통합을 제공합니다. 자세한 내용은 Amazon CodeWhisperer AWS Infrastructure Composer에서 사용을 참조하세요.	2023년 11월 30일
에서 Infrastructure Composer를 사용하여 애플리케이션을	Infrastructure Composer 캔버스의 동기화 버튼을 사용하여	2023년 11월 30일

배포하기 위한 설명서 추가 AWS Toolkit for Visual Studio Code	애플리케이션을에 배포합니다 AWS 클라우드. 자세한 내용은 sam sync를 사용하여 애플리케이션 배포를 참조하세요.	
의 Infrastructure Composer에 대한 설명서 추가 AWS Toolkit for Visual Studio Code	이제 VS Code의 Infrastructure Composer를와 함께 사용할 수 있습니다 AWS Toolkit for Visual Studio Code. 자세한 내용은 AWS Infrastructure Composer 에서 사용을 참조하세요 AWS Toolkit for Visual Studio Code.	2023년 11월 30일
Step Functions Workflow Studio 통합 추가	Infrastructure Composer 캔버스에서 Step Functions Workflow Studio를 시작합니다. 자세한 내용은 AWS Infrastructure Composer 와 함께 사용을 AWS Step Functions 참조하세요.	2023년 11월 27일
Lambda 콘솔 및 Infrastructure Composer 통합 추가	Lambda 콘솔에서 Infrastructure Composer 캔버스를 시작합니다. 자세한 내용은 AWS Lambda 콘솔 AWS Infrastructure Composer 에서 사용을 참조하세요.	2023년 11월 14일

Infrastructure Composer의 추천 서비스로 Amazon VPC 추가	Infrastructure Composer는 VPC 태그를 도입하여 VPC로 구성된 리소스를 시각화합니다. 외부 템플릿에 정의된 VPCs를 사용하여 Lambda 함수를 구성할 수도 있습니다. 자세한 내용은 Amazon VPC에서 Infrastructure Composer 사용을 참조하세요.	2023년 10월 17일
Infrastructure Composer를 사용하여 Amazon RDS를 추천 서비스로 추가	Infrastructure Composer 애플리케이션을 외부 템플릿에 정의된 Amazon RDS DB 클러스터 또는 인스턴스에 연결합니다. 자세한 내용은 Amazon RDS에서 Infrastructure Composer 사용을 참조하세요.	2023년 10월 17일
모든 CloudFormation 리소스로 설계할 수 있도록 Infrastructure Composer 지원 추가	CloudFormation 리소스 팔레트에서 애플리케이션을 설계할 리소스를 선택합니다. 자세한 내용은 모든 리소스 작업을 참조하세요 CloudFormation.	2023년 9월 26일
Infrastructure Composer의 카드에 대한 설명서 추가	Infrastructure Composer는 애플리케이션을 설계하고 빌드하는 데 사용할 수 있는 여러 유형의 카드를 지원합니다. 자세한 내용은 Infrastructure Composer에서 카드로 설계를 참조하세요.	2023년 9월 20일

실행 취소 및 다시 실행 기능에 대한 설명서 추가	Infrastructure Composer 캔버스에서 실행 취소 및 다시 실행 버튼을 사용합니다. 자세한 내용은 실행 취소 및 다시 실행을 참조하세요.	2023년 8월 1일
로컬 동기화 모드에 대한 설명서 추가	로컬 동기화 모드를 사용하여 프로젝트를 자동으로 동기화하고 로컬 시스템에 저장합니다. 자세한 내용은 로컬 동기화 모드를 참조하세요.	2023년 8월 1일
캔버스 내보내기 기능에 대한 설명서 추가	캔버스 내보내기 기능을 사용하여 애플리케이션의 캔버스를 로컬 시스템으로 이미지로 내보냅니다. 자세한 내용은 캔버스 내보내기를 참조하세요.	2023년 8월 1일
외부 파일 참조에 대한 Infrastructure Composer 지원	Infrastructure Composer에서 지원되는 리소스에 대한 외부 파일을 참조하세요. 자세한 내용은 외부 파일을 참조하는 템플릿 작업을 참조하세요.	2023년 5월 17일
리소스 연결에 대한 새로운 설명서	리소스를 함께 연결하여 애플리케이션의 리소스 간에 이벤트 기반 관계를 정의합니다. 자세한 내용은 Infrastructure Composer 시각적 캔버스를 사용하여 리소스 함께 연결을 참조하세요.	2023년 3월 7일

새로운 Change Inspector 기능	Change Inspector를 사용하여 템플릿 코드 업데이트를 보고 Infrastructure Composer가 생성하는 내용을 알아봅니다. 자세한 내용은 Change Inspector로 코드 업데이트 보기를 참조하세요.	2023년 3월 7일
이제 Infrastructure Composer를 정식 버전으로 사용할 수 있습니다.	AWS Infrastructure Composer 이제를 정식 버전으로 사용할 수 있습니다. 자세한 내용은 AWS Infrastructure Composer 이제 정식 출시 - 서버리스 애플리케이션을 빠르게 시각적으로 빌드를 참조하세요.	2023년 3월 7일
연결 모드 사용의 이점에 대해 확장됨	로컬 IDE와 연결된 모드에서 Infrastructure Composer를 사용하여 개발 속도를 높입니다. 자세한 내용은 로컬 IDE에서 Infrastructure Composer 사용을 참조하세요.	2023년 3월 7일
다른 AWS 서비스를 사용하여 애플리케이션을 배포하는 방법에 대한 주제 업데이트	Infrastructure Composer를 사용하여 배포 가능한 서버리스 애플리케이션을 설계합니다. AWS SAM 를 사용하여 서버리스 애플리케이션을 배포합니다. 자세한 내용은 및에서 CloudFormation Infrastructure Composer 사용을 AWS SAM 참조하세요.	2023년 3월 3일

서버리스 개념 섹션 추가	Infrastructure Composer를 사용하기 전에 기본 서버리스 개념에 대해 알아봅니다. 자세한 내용은 서버리스 개념을 참조하세요.	2023년 3월 2일
공개 릴리스	Infrastructure Composer의 최초 공개 릴리스입니다.	2022년 12월 1일

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