

Implementation Guide

# Modern Data Architecture Accelerator



# Modern Data Architecture Accelerator: Implementation Guide

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

|   |           |
|---|-----------|
| <b>Solution overview .....</b>  | <b>1</b>  |
| Use cases .....   | 2         |
| Concepts and definitions .....  | 3         |
| Solution Structure .....  | 4         |
| The Module Concept .....  | 4         |
| How Modules Work Together .....   | 5         |
| MDAA Starter Packages .....   | 5         |
| <b>Architecture overview .....</b>  | <b>11</b> |
| Architecture diagram .....  | 11        |
| Sample configurations .....   | 13        |
| MDAA Configuration Structure .....  | 14        |
| Module Configurations .....   | 16        |
| <b>Plan your deployment .....</b>   | <b>19</b> |
| Supported AWS Regions .....   | 19        |
| Cost .....  | 19        |
| Example cost tables .....   | 20        |
| Security .....  | 21        |
| Security Controls and Compliance .....  | 22        |
| Quotas .....  | 23        |
| Quotas for AWS services in this solution .....  | 23        |
| AWS CloudFormation quotas .....   | 24        |
| AWS Lambda quotas .....   | 24        |
| Amazon VPC quotas .....   | 24        |
| <b>Deploy the solution .....</b>  | <b>25</b> |
| Deployment using installer CloudFormation template .....                                  | 25        |
| Prerequisites .....   | 25        |
| Source of the Solution Code .....   | 28        |
| Launch the installer stack .....  | 28        |
| Await initial environment deployment .....  | 31        |
| Terminology .....   | 32        |
| Prerequisites for CDK Deployment .....  | 32        |
| Manual Bootstrap (Single Deployment Source/Target Account) .....                          | 32        |
| Manual Bootstrap (Single Deployment Source Account and One or More Target Accounts) ..... | 32        |

|  |           |
|--|-----------|
| Deployment .....   | 33        |
| Deployment Overview .....                                  | 33        |
| Deployment Patterns .....                                  | 33        |
| Deployment Preparation .....                               | 33        |
| Deploying starter packages .....                           | 37        |
| AI/ML Starter Package .....                                | 38        |
| Datalake Starter Package .....                             | 43        |
| GenAI Accelerator Starter Package .....                    | 48        |
| Governed Lakehouse Starter Package .....                   | 53        |
| <b>Update the solution .....</b>                           | <b>58</b> |
| <b>Troubleshooting .....</b>                               | <b>59</b> |
| Known issue resolution .....                               | 59        |
| Failed to upload data in S3 bucket .....                   | 59        |
| Data Science Configuration Deployment Failure .....        | 59        |
| Lake Formation Data Lake Deployment Issues .....           | 60        |
| Failed to resolve context: vpc_id .....                    | 61        |
| Contact AWS Support .....                                  | 62        |
| Create case .....  | 62        |
| How can we help? .....                                     | 63        |
| Additional information .....                               | 63        |
| Help us resolve your case faster .....                     | 63        |
| Solve now or contact us .....                              | 63        |
| <b>Uninstall the solution .....</b>                        | <b>64</b> |
| <b>Use the solution .....</b>                              | <b>65</b> |
| Using configuration files .....                            | 65        |
| Configuration File Structures .....                        | 65        |
| Single Domain, Shared CDK Apps Configs Across Envs .....   | 65        |
| Single Domain, Separate CDK Apps Configs Across Envs ..... | 66        |
| Working with dynamic references .....                      | 66        |
| Configuration sharing and composition .....                | 66        |
| Configuration Merging Rules .....                          | 66        |
| Customizing the solution .....                             | 67        |
| <b>Developer guide .....</b>                               | <b>68</b> |
| Setting up MDAA Dev Environment .....                      | 68        |
| Testing .....  | 68        |
| Testing Overview .....                                     | 68        |

|                                     |           |
|-------------------------------------|-----------|
| Testing Constructs and Stacks ..... | 68        |
| Testing Apps .....                  | 70        |
| <b>Reference .....</b>              | <b>71</b> |
| Anonymized data collection .....    | 71        |
| Contributors .....                  | 72        |
| <b>Notices .....</b>                | <b>73</b> |

# Accelerate the Deployment of Secure and Compliant Modern Data Architectures for Advanced Analytics and AI

Publication date: August 2025. For updates, refer to [CHANGELOG.md](#) file in the MDAA Developer Guide.

The Modern Data Architecture Accelerator (MDAA) on AWS helps customers rapidly deploy and manage sophisticated data platform architectures on AWS. This solution provides a flexible framework that can adapt to most common analytics platform architectures, including basic Data Lakes and Data Warehouses, Lake House architectures, complex Data Mesh implementations, and generative AI development environments. The solution helps you establish a modern data foundation with built-in security, governance, and operational capabilities. Through a simplified configuration approach, you can:

- Deploy generative AI development environments with integrated AWS services
- Configure and manage AI/ML workloads including generative AI solutions
- Deploy data environments across multiple domains and AWS accounts
- Implement AWS reference architectures like Modern Data Architecture (Lake House)
- Configure and manage data mesh nodes for distributed data platforms
- Manage data governance controls and security services
- Define and deploy purpose-built analytics services for specific use cases
- Deploy machine learning workload environments
- Customise deployments through infrastructure-as-code using AWS CDK
- Configure data ingestion patterns, storage layers, and processing capabilities

MDAA is provided as an open-source solution that can be deployed from locally cloned source code or published NPM packages. You pay only for AWS services enabled to set up your data platform and operate your workloads.

## Key Benefits

- Accelerated Time to Value: Deploy a production-ready modern data platform in weeks instead of months

- **Built-in Data Governance:** Implement data security, privacy, and compliance controls from day one
- **Standardised Architecture:** Ensure consistent data handling patterns and practices across the organisation
- **Analytics-Ready Infrastructure:** Pre-configured analytics services and data processing pipelines, including integrated components for developing and deploying generative AI solutions
- **Cost Optimisation:** Built-in cost management and data lifecycle optimisation features

The intended audience for using this solution’s features and capabilities in their environment are data platform engineers, data architects, analytics teams and cloud operations professionals.

Use this navigation table to quickly find answers to these questions:

This implementation guide describes architectural considerations and configuration steps for deploying MDAA. It includes links to AWS CloudFormation templates synthesised from AWS CDK that launch and configure the AWS services required to deploy this solution using AWS best practices for security and availability

| If you want to . . .  | Read . . .                            |
|---|---------------------------------------|
| Know the cost for running this solution.<br><br>The cost will depend on the modules and other custom features you want to deploy. | <a href="#">Cost</a>                  |
| Know which AWS Regions support this solution.   | <a href="#">Supported AWS Regions</a> |
| Access the source code.   | <a href="#">GitHub repository</a>     |

## Use cases

- **Centralised Data Platforms** - Deploy and manage centralized data lakes and warehouses from a single account
- **Distributed Data Mesh** - Create autonomous data mesh nodes for individual business units while maintaining unified governance

- **Hub and Spoke Architecture** - Implement hybrid models with centralized enterprise data assets and distributed business unit autonomy
- **Analytics and ML Platforms** - Build platforms supporting analytics, data science, and AI/ML workloads
- **Custom Data Architectures** - Adapt and extend the framework to implement custom data platform architectures

## Concepts and definitions

### **Analytics Data Lake**

A centralised repository that allows you to store structured and unstructured data at any scale. It enables you to break down data silos and combine different types of analytics to gain insights and guide better business decisions.

### **Data Mesh**

A decentralised socio-technical approach to data management and analytics that treats data as a product and applies domain-oriented, self-serve design to distribute data ownership and architecture.

### **Data Product**

A reusable dataset with clear ownership, documentation, and service-level objectives that can be easily discovered and consumed by authorised users across the organisation.

### **Federated Access Control**

A security mechanism that enables centralised management of user identities and access permissions across multiple systems and domains while maintaining consistent security policies.

### **Data Governance**

The overall management of data availability, usability, integrity, and security in an enterprise system. It includes policies, procedures, and standards that ensure data is managed consistently and used appropriately.

### **Data Quality**

The measure of data's condition and its fitness to serve its intended purpose in a given context. This includes accuracy, completeness, consistency, timeliness, and validity of the data.



## Self-Service Analytics

A form of business intelligence where users can access and analyse data without requiring assistance from IT or data specialists, enabling faster decision-making and reducing bottlenecks.

## Data Catalog

A centralised metadata repository that helps organisations discover, understand, and manage their data assets. It provides a searchable inventory of data assets across the data platform.

## Data Pipeline

A series of automated steps that extract data from various sources, transform it according to business rules, and load it into target systems for analysis and reporting.

## Data Domain

A logical grouping of related data assets and processes managed by a specific business unit or team, typically aligned with organisational functions or business capabilities.

### Note

For a general reference of AWS terms, see the [AWS Glossary](#).

## Solution Structure

MDAA is a comprehensive solution built using a modular approach. Think of it as a sophisticated building kit for creating secure and scalable data infrastructure on AWS. Just as a building needs a foundation, walls, and utilities, MDAA provides all the necessary components to build your data/AI platform.

## The Module Concept

Think of modules as specialized building blocks. For example:

- If you want to deploy raw and transformation buckets for your data lake, there's a datalake module that creates encrypted S3 buckets with proper access controls, sets up fine-grained lifecycle policies for cost optimization and configures bucket policies and cross-account access if needed

- If you need to query data using Amazon Athena, there's a module that sets up Athena workgroups with resource controls, configures query result locations, connects with your datalake and establishes necessary IAM permissions for query execution
- If you want to add AWS Lake Formation settings to your tables, there's a module that configures Lake Formation permissions and security settings, sets up database and table-level access controls, etc.

## How Modules Work Together

Consider this practical scenario: You want to build a secure data lake for financial data.

- Start with the roles module to create necessary IAM roles and policies
- Add the datalake module to create encrypted storage
- Add the Glue module to catalog your data
- Implement Lake Formation module for compliance
- Configure Athena module for analysts to query
- Add audit modules for security

## MDAA Starter Packages

### Overview

Modern Data Architecture Accelerator (MDAA) provides a comprehensive set of pre-configured starter packages, each designed to accelerate your journey in building enterprise-grade secure and compliant data platforms on AWS. These packages eliminate the complexity of starting from scratch by providing production-ready configurations, security controls, and infrastructure templates.

### Available Starter Packages

#### 1. Basic Data Lake Package

| Purpose      | Basic data lake foundation  |
|--------------|---|
| Key Features | <ul style="list-style-type: none"><li>• Encrypted S3 buckets for raw and processed data</li></ul> |

| Purpose   | Basic data lake foundation  |
|-----------|---|
|           | <ul style="list-style-type: none"> <li>• Glue Catalog configuration for data discovery</li> <li>• Basic Athena setup for SQL querying</li> <li>• Foundation security controls and IAM roles</li> <li>• Data lifecycle management policies</li> <li>• Audit capabilities through CloudTrail for complete data access tracking</li> </ul> |
| Ideal For | <ul style="list-style-type: none"> <li>• Organizations who want to centralize data storage with appropriate security controls</li> <li>• Implementing governance and compliance requirements</li> <li>• Enable self-service data access for various user roles</li> <li>• Fine-grained roles access for admins and users</li> </ul>     |

## 2. AI/ML Platform Package

| Purpose      | Enterprise-grade machine learning infrastructure  |
|--------------|---|
| Key Features | <ul style="list-style-type: none"> <li>• All Basic Data Lake features, plus:</li> <li>• SageMaker Studio domains with pre-configured user profiles and permissions</li> <li>• IAM roles with least-privilege permissions for data science operations</li> <li>• S3 buckets for storing training data, model artifacts, and results</li> <li>• AWS Glue Data Catalog integration for comprehensive metadata management</li> <li>• Lake Formation configuration for fine-grained access control</li> <li>• Audit capabilities through CloudTrail for complete model development tracking</li> </ul> |
| Best For     | <ul style="list-style-type: none"> <li>• Data science and ML teams</li> <li>• Data science teams to rapidly develop and deploy ML models.</li> <li>• Organization which requires governed access to data and model resources.</li> </ul>  |

### 3. GenAI Accelerator Starter Package

| Purpose      | Generative AI development and deployment platform   |
|--------------|---|
| Key Features | <ul style="list-style-type: none"> <li>• All Basic Datalake features, plus:</li> <li>• Pre-configured Amazon Bedrock access and model permissions</li> <li>• Aurora Vector database setup for RAG (Retrieval Augmented Generation) applications</li> <li>• Knowledge base integration for document processing</li> <li>• Autosync capability to sync documents to knowledge bases from S3 bucket</li> <li>• Prompt engineering and agent customization capabilities</li> <li>• Custom Transform function for Knowledge bases</li> <li>• Includes customer support assistant example implementation</li> </ul> |
| Best For     | <ul style="list-style-type: none"> <li>• Organizations building generative AI applications</li> <li>• Teams developing Agentic AI solutions</li> <li>• Enterprises requiring secure and governed GenAI Applications</li> </ul>  |

### 4. Governed Lakehouse Package

| Purpose      | Enterprise lakehouse with comprehensive data governance using DataZone   |
|--------------|--|
| Key Features | <ul style="list-style-type: none"> <li>• All Basic Data Lake features, plus:</li> <li>• DataZone domains for data product management and discovery</li> <li>• Fine-grained access control using Lake Formation</li> <li>• Multi-team data producer and consumer support</li> <li>• KMS-encrypted S3 buckets with proper access controls</li> <li>• Glue crawlers for automated data cataloging</li> <li>• IAM roles with data-admin and data-user permissions</li> <li>• Structured data consumption primarily via Athena</li> </ul> |
| Best For     | <ul style="list-style-type: none"> <li>• Enterprises requiring fine-grained data access control</li> </ul>   |

| Purpose | Enterprise lakehouse with comprehensive data governance using DataZone   |
|---------|--|
|         | <ul style="list-style-type: none"><li>• Multi-team environments with data producers and consumers</li><li>• Organizations building data product marketplaces</li><li>• Teams working primarily with structured data via Athena</li></ul> |

## Package Benefits

### Time to Market

- Reduce implementation time by 60-70%
- Avoid common architectural pitfalls
- Start with proven configurations

### Cost Optimization

- Pre-configured resource optimization
- Built-in cost control measures
- Efficient resource utilization patterns

### Security & Compliance

- Security controls aligned with AWS best practices
- Built-in compliance frameworks
- Automated security monitoring

### Scalability

- Designed for growth
- Flexible architecture
- Easy module addition/removal

## Best Practices

### Security

- Enable all recommended security features
- Implement proper encryption
- Regular security assessments
- Continuous monitoring

### Operations

- Follow GitOps practices
- Implement proper tagging
- Regular backup testing
- Disaster recovery planning

### Cost Management

- Enable cost allocation tags
- Set up budget alerts
- Regular cost reviews
- Resource optimization

## Support and Maintenance

### Regular Updates

- Security patches
- Feature updates
- Performance improvements
- Best practice updates

**Note**

All packages are regularly updated to incorporate the latest AWS features and security best practices.

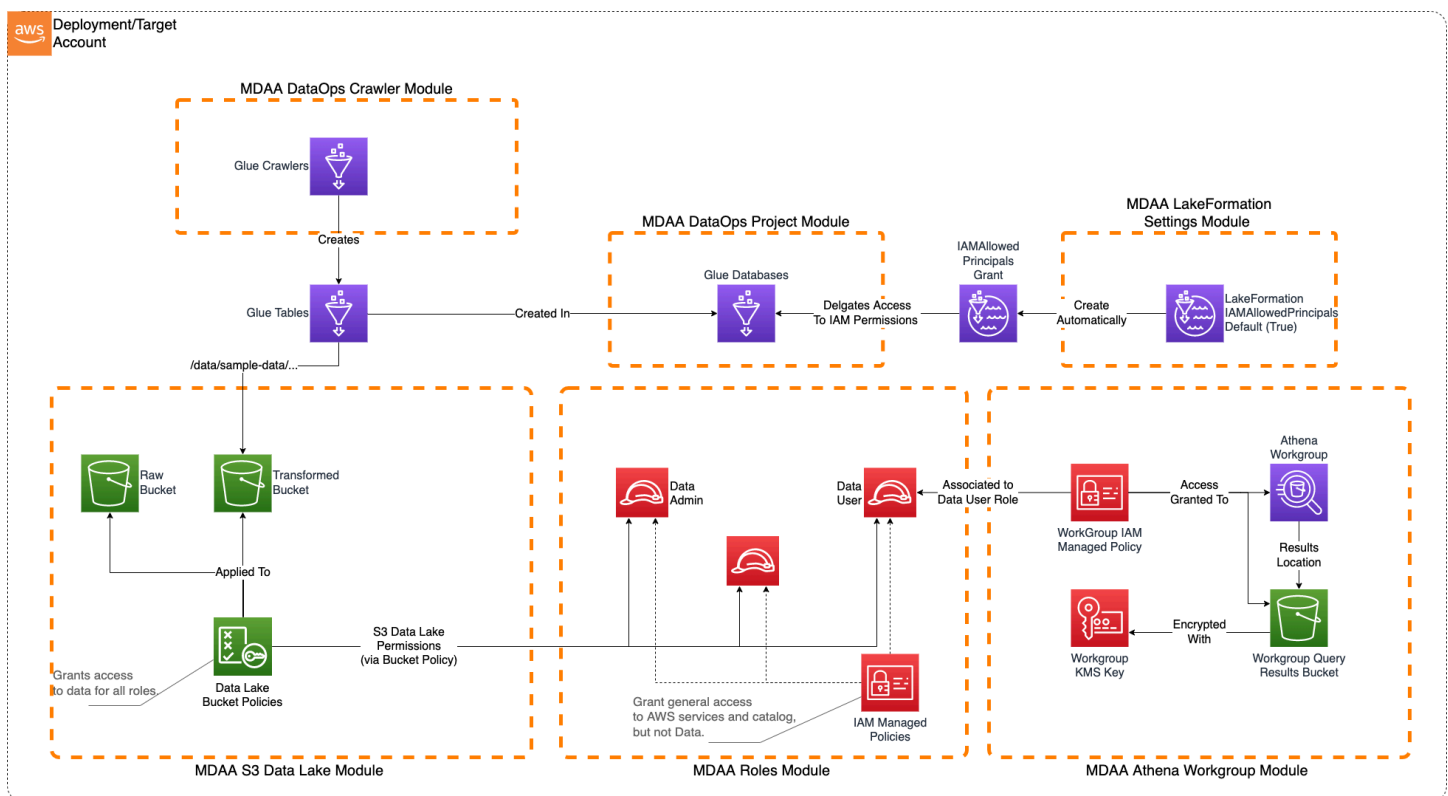
# Architecture overview

This section provides a high-level overview of how MDAA enables flexible deployment of data platforms on AWS. MDAA supports multiple architectural patterns that can evolve as your organization's needs change.

## Architecture diagram

The first step of building an analytics platform with MDAA is to decide on an initial architecture. MDAA is extremely flexible, able to be adapted to most common analytics platform architectures on AWS. These include basic Data Lakes and Data Warehouses, Lake House, and complex Data Mesh architectures. Note that the initial architecture does not need to be the target end-state architecture for the platform, as these architectures often build on each other, and a MDAA configuration/deployment can be adapted through iteration from one architectural state to another.

As an example, the following is the architecture for the Basic Datalake.





## Basic Datalake on AWS architecture

1. You use AWS CloudFormation to install MDAA into your environment. Your environment must meet prerequisites before deploying the solution. (See [PREDEPLOYMENT](#).) The provided CloudFormation template deploys an AWS CodePipeline that contains the MDAA installation engine for building analytics platforms.
2. The Modern Data Architecture (Lake House) framework functions as the core architecture pattern. This way, you can establish a flexible, scalable foundation for solving virtually any data problem—using analytics, data science, or AI/ML—on AWS. The architecture remains fully open and interoperable with data capabilities both inside and outside of AWS.
3. An S3-based data lake serves as the core component, wrapped with a unified data governance layer. The solution deploys AWS Glue and AWS Lake Formation to provide comprehensive data cataloging and access controls. Additionally, the solution implements a DataOps layer to facilitate seamless data movement between the core data lake and purpose-built analytics services on the perimeter.
4. The solution deploys purpose-built analytics services that you can select based on specific use cases. These services support various analytical workloads including traditional BI, data science, and machine learning. The solution maintains flexibility to add or modify analytics services as requirements evolve.
5. For organizations requiring distributed data architecture, MDAA supports deployment of Data Mesh patterns. Each business unit can operate an autonomous data mesh node, typically implementing an individual Lake House architecture. The solution enables producer/consumer relationships between nodes while maintaining unified governance.

### Supported Architecture Patterns

- Modern Data Architecture (Lake House) - The AWS reference architecture centered around an S3-based data lake with unified governance and analytics capabilities
- Data Mesh - A distributed architecture pattern providing autonomy to business units while maintaining centralized governance
- Hub and Spoke - A hybrid model combining centralized enterprise data assets with distributed business unit capabilities

### Core Platform Functions

MDAA implements these key platform capabilities:

- Data Ingest
- S3 Data Lake/Persistence
- Governance
- Processing/Curation (DataOps)
- Analytics, Query, and Consumption
- Data Science, AI/ML
- Visualization

## Additional notes

Initial deployment establishes the core data lake, governance framework, and basic analytics capabilities. Additional components can be added iteratively based on specific requirements and use cases.

The solution maintains flexibility to evolve from basic data lake architectures to more sophisticated patterns like Data Mesh. The unified governance model ensures consistent controls even as the architecture grows in complexity. When implementing Data Mesh patterns, each node maintains autonomy while adhering to organization-wide governance standards.

We provide guidance on selecting appropriate architecture patterns based on organizational maturity, use cases, and compliance requirements. The solution documentation includes detailed deployment procedures for each supported pattern.

## Sample configurations

Modern Data Architecture Accelerator (MDAA) on AWS includes example sample configurations that allow you to quickly deploy analytics workloads and data platforms across your organisation. The repository includes sample configurations and detailed documentation that provide guidance for implementing various modules including analytics platform, ML workloads and data architectures across different AWS Regions.

MDAA is designed to deploy data environments across multiple domains and environments. Each domain/environment is constituted by one or more configured MDAA modules. Each MDAA module references a CDK app and corresponding configuration. During deployment, MDAA executes a module's underlying CDK application, providing all necessary configuration details as CDK context.

We built these sample configurations based on AWS best practices and common analytics use cases across industries. This solution provides automated deployment of analytics infrastructure while maintaining flexibility to customise based on your specific data requirements, security needs, and compliance standards

These sample MDAA configurations are provided as a starting point for common analytics platform architectures.

- Basic DataLake with Glue - A basic S3 Data Lake with Glue database and crawler
- Basic Terraform DataLake - A basic S3 Data Lake built with the MDAA Terraform module
- Fine-grained Access Control DataLake - An S3 Data Lake with fine-grained access control using LakeFormation
- Data Warehouse - A standalone Redshift Data Warehouse
- Lakehouse - A full LakeHouse implementation, with Data Lake, Data Ops Layers (using NYC taxi data), and a Redshift data warehouse
- Data Science Platform - A standalone SageMaker Studio Data Science Platform
- GenAI Platform - A standalone GenAI Accelerator Platform

Additional customization of the baselines will likely be required to align with your organization's specific analytics needs and compliance requirements

## MDAA Configuration Structure

MDAA is designed to deploy data environments across multiple domains and environments. Each domain/environment is constituted by one or more configured MDAA modules. Each MDAA module references a CDK app and corresponding configuration. During deployment, MDAA executes a module's underlying CDK application, providing all necessary configuration details as CDK context.

- Domain - A data environment can be organized into one or more domains, which may align to organizational units such as line of business, directorate, etc. Domains may be spread across one or more accounts. When spread across multiple accounts, each domain becomes a potential node in a data mesh architecture.
- Environment - An domain can be deployed across multiple environments (such as DEV/TEST/PROD). Each environment may deployed in a separate account.

- **Module** - A module specifies which CDK App and corresponding configuration will be deployed within an data environment domain/environment. During deployment, modules will be deployed in stages according to dependencies between modules.
- **CDK App** - A CDK App is built, executed, and deployed using the AWS CDK framework. The CDK app will be forked from the MDAA orchestrator and executed as a regular CDK application. Each CDK produces one or more CloudFormation stacks, which in turn deploy the cloud resources which will constitute the data environment. Alternatively, instead of deploying resources directly to the environment, they can instead be published as Service Catalog products, to be deployed on a self-service basis by users within the accounts.

**MDAA Config File/Folder Layouts** MDAA is configured using a set of YAML configuration files. The main CLI configuration file (typically 'mdaa.yaml') specifies the global, domain, environment, and modules to be deployed. Module (CDK App) configurations are specified in separate YAML files, or can be configured inline in the CLI config itself. Module (CDK App) configurations are documented in detail in their respective READMEs. Terraform modules are configured directly using HCL configurations next to mdaa.yaml. MDAA configuration layouts are very flexible. Configurations for an entire org can be concentrated into a single MDAA config file, or can be spread out across multiple config files by domain, line of business, environment, etc.

### **Single Domain, Shared CDK Apps Configs Across Envs**

In this scenario, a MDAA config contains a single domain, with CDK App configs shared across dev/test/prod. In this case, the shared configs likely make heavy use of SSM parameters to achieve portability across environments.

### **Single Domain, Separate CDK Apps Configs Across Envs**

In this scenario, a MDAA config contains a single domain, with separate CDK App configs across dev/test/prod.

### **Multiple domains, single MDAA config**

In this scenario, multiple domains are in the same MDAA Config/

### **Multiple domains, Multiple MDAA config**

In this scenario, each domain is in its own MDAA config.

## Module Configurations

Each MDAA Module/CDK App has its own configuration schema, which is documented in their respective READMEs. There are some common configuration behaviours and capabilities, however, which can be used across all MDAA Module configs.

### Dynamic References

MDAA allows use of Dynamic References in configuration files. These build on the concept of CloudFormation Dynamic References.

```
# Example Config File w/Dynamic References

# Will be passed through to CloudFormation as a CFN Dynamic Reference and will be
  resolved at deployment time
vpcId: "{{resolve:ssm:/path/to/ssm/param}}"
sensitive_value: "{{resolve:ssm-secure:parameter-name:version}}"
db_username: "{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}",
db_password: "{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}"

# Will be resolved at synth time to the CDK context value passed from the MDAA CLI
  config or directly in CDK context
subnetId: "{{context:some_context_key}}"

# Will be resolved at synth time to environment variable values
subnetId: "{{env_var:some_env_variable_name}}"

# Will be resolved at synth time to the values passed for org/domain/env/account/region
  from the MDAA CLI config via CDK context
# Identical to org: "{{context:org}}"
org: "{{org}}"
domain: "{{domain}}"
env: "{{env}}"
module_name: "{{module_name}}"
partition: "{{partition}}"
account: "{{account}}"
region: "{{region}}"

# Dynamic references can also be embedded inline in config values:
key_arn: arn:{{partition}}:kms:{{region}}:{{account}}:key/{{context:key_id}}
```

### Configuration Sharing Across Domains, Envs, Modules

MDAA modules may share identical config files across multiple domains, envs, and modules. Because MDAA automatically injects the domain/env/module names into resource naming, each resulting deployment will result in uniquely named resources but with otherwise identical behaviours.

```
# Example MDAA Config With Shared Configs
domains:
  domain1:
    environments:
      dev:
        modules:
          test_datalake:
            module_path: "@aws-mdaa/datalake"
            module_configs:
              - ./shared/datalake.yaml
  domain2:
    environments:
      dev:
        modules:
          test_datalake:
            module_path: "@aws-mdaa/datalake"
            module_configs:
              - ./shared/datalake.yaml
```

Both datalakes will have identical configurations, but named according to their domain.

## Configuration Composition

Each MDAA module accepts one or more configuration files, which are merged into an effective config, which is then validated and parsed by the app. This allows for configs to be composed of common base configs shared across multiple modules, environments, or domains, with only the differentiating config values to be applied on top. In general, config files will be merged according to the following rules:

- Lists on same config key will be merged across config files
- Objects on same config key will be concatenated
- Scalar values will be overridden, with config files higher on list taking precedence

```
# Example MDAA CLI Module Specification With Multiple Configs
domains:
```

```
domain1:
  environments:
    dev:
      modules:
        roles1:
          module_path: "@aws-mdaa/roles"
          module_configs:
            - ./domain1/roles1.yaml
            - ./shared/roles_base.yaml
domain2:
  environments:
    dev:
      modules:
        roles2:
          module_path: "@aws-mdaa/roles"
          module_configs:
            - ./domain2/roles2.yaml
            - ./shared/roles_base.yaml
```

# Plan your deployment

This section describes the Region, cost, security, quota, and other considerations for planning your deployment.

## Supported AWS Regions

This solution uses numerous services, which aren't currently available in all AWS Regions. We recommend using AWS Control Tower and AWS Organizations when launching this solution in an AWS Region where these services are available. For the most current availability of AWS services by Region, refer to the [AWS Regional Services List](#).

## Cost

You are responsible for the cost of AWS services used while running this solution. As of August 2025, costs primarily depend on the resources used, data processed, transferred, and stored.

S3 costs vary based on storage class, data volume, request types, data retrieval, transfer rates, and additional features. IAM is provided at no additional cost. For KMS, costs depend on the encryption type: SSE-S3 (default encryption) incurs no additional charge, while SSE-KMS incurs both a monthly fee (\$1/month per key) and per-request charges (\$0.03 per 10,000 requests). If using SSE-KMS, enabling S3 Bucket Keys can reduce KMS costs by up to 99%.

For detailed pricing information and to estimate costs for your specific implementation, we recommend using the [AWS Pricing Calculator](#).

### Note

The cost for running the Modern Data Architecture Accelerator in the AWS Cloud depends on the deployment configuration you choose. The following examples provide cost breakdown for some of the sample configurations deployed in the US East (N. Virginia) Region. AWS services listed in the example tables below are billed on a monthly basis.

We recommend creating a [budget](#) through [AWS Cost Explorer](#) to help manage costs. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.



## Example cost tables

The following samples are based on the options from the installer CloudFormation template.

### Basic Data Lake

| Resources  | Monthly cost [USD] |
|--|--------------------|
| <b>Glue Catalog</b> - Configures the Encryption at Rest settings for Glue Catalog at the account level. Additionally, configures Glue catalogs for cross account access required by a Data Mesh architecture.  | \$0.90             |
| <b>Audit</b> - Configures and deploys Audit resources to use as target for audit data and for querying audit data via Athena   | \$0.90             |
| <b>Datalake KMS and Buckets</b> - Configures and deploys a set of encrypted data lake buckets and bucket policies. Bucket policies are suitable for direct access via IAM and/or federated roles, as well as indirect access via LakeFormation/Athena. | \$0.90             |
| <b>Athena Workgroup</b> - Configures and deploys Athena Workgroups for use on the Data Lake  | \$0.90             |
| <b>Data Science Team/Project</b> - Deploys resource to support a team's Data Science activities  | \$0.90             |
| <b>Total</b>   | <b>\$4.50*</b>     |

## Data Science setup

| Resources  | Monthly cost [USD] |
|--|--------------------|
| <b>Glue Catalog</b> - Configures the Encryption at Rest settings for Glue Catalog at the account level. Additionally, configures Glue catalogs for cross account access required by a Data Mesh architecture.  | \$0.90             |
| <b>Audit</b> - Configures and deploys Audit resources to use as target for audit data and for querying audit data via Athena   | \$0.90             |
| <b>Datalake KMS and Buckets</b> - Configures and deploys a set of encrypted data lake buckets and bucket policies. Bucket policies are suitable for direct access via IAM and/or federated roles, as well as indirect access via LakeFormation/Athena. | \$0.90             |
| <b>Athena Workgroup</b> - Configures and deploys Athena Workgroups for use on the Data Lake  | \$0.90             |
| <b>Data Science Team/Project</b> - Deploys resource to support a team's Data Science activities  | \$0.90             |
| <b>Total</b>   | <b>\$5.40*</b>     |

\*Your final cost depends the usage of the resources. Estimates above does not account for customer's workloads.

## Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared responsibility model](#) reduces your operational burden because AWS operates,

manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit the [AWS Security Center](#).

## Security Controls and Compliance

MDAA implements multiple security controls and compliance measures:

- Compliance with multiple AWS CDK Nag rulesets:
- AWS Solutions ruleset
- NIST 800-53 Rev 5 ruleset
- HIPAA ruleset
- Adherence to ITSG-33 PBMM Security Control Requirements
- Implementation of security best practices across all deployed resources

## Encryption

MDAA enforces comprehensive encryption measures:

- Ubiquitous encryption at rest for all data storage components
- Mandatory encryption in transit for all data transfers
- Integration with AWS KMS for key management

## Access Control

The solution implements the principle of least privilege:

- Least-privileged permissions by default for all deployed resources
- Role-based access control (RBAC) implementation
- Secure self-service deployments through AWS Service Catalog (optional)

## Governance Controls

MDAA provides several governance mechanisms:

- AWS CloudFormation as the single deployment mechanism through CDK

- Consistent resource naming conventions across all deployments
- Standardized tagging strategy for all generated resources
- Centralized change management through Infrastructure as Code

## Resource Management

Security is enforced through:

- Consistent deployment patterns across all MDAA modules
- Standardized SSM parameter publication for secure resource reference
- Compliant resource configurations by default

## Monitoring and Metrics

The solution includes:

- Anonymous operational metrics collection (with opt-out capability)
- Integration with AWS native security monitoring services
- Compliance validation capabilities

## Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

### Quotas for AWS services in this solution

Make sure you have sufficient quota for the services to be deployed by your configuration. For more information, see [AWS service quotas](#).

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the [Service endpoints and quotas](#) page in the PDF instead.

[Amplify](#)

[Amazon ECR](#)

|                            |                                    |
|----------------------------|------------------------------------|
| <a href="#">Athena</a>     | <a href="#">Lambda</a>             |
| <a href="#">CloudFront</a> | <a href="#">OpenSearch Service</a> |
| <a href="#">Cognito</a>    | <a href="#">Neptune</a>            |
| <a href="#">Config</a>     | <a href="#">Amazon S3</a>          |
| <a href="#">Amazon ECS</a> |                                    |

## AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when launching this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, see [AWS CloudFormation quotas](#) in the *AWS CloudFormation User's Guide*.

## AWS Lambda quotas

Your account has an AWS Lambda concurrent execution quota of 1000. If the solution is used in an account where there are other workloads running and using Lambda, then set this quota to an appropriate value. This value is adjustable; for more information, see [AWS Lambda quotas](#) in the *AWS Lambda User's Guide*.

### Note

This solution requires 150 executions from the concurrent execution quota to be available in the account to which the solution is being deployed. If there are fewer than 150 executions available in that account, the CloudFormation deployment will fail.

## Amazon VPC quotas

Your AWS account can contain five VPCs and two Elastic IPs (EIPs). If the solution is used in an account with other VPCs or EIPs, this could prevent you from deploying this solution successfully. If you are at risk of reaching this quota, you may provide your own VPC for deployment by providing it in your configuration. For more information, see [Amazon VPC quotas](#) in the *Amazon VPC User's Guide*.

# Deploy the solution

The solution is mainly deployed as an AWS CDK application. Customers can also use a CloudFormation template to deploy the solution based on some sample configurations.

Before you deploy, review the [cost](#), [architecture](#), and other considerations discussed earlier in this guide.

## Deployment using installer CloudFormation template

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template specifies the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources described in the template.

**Time to deploy:** Approximately five minutes for the installer CloudFormation stack and up to 45 minutes for the sample configuration you choose to deploy with.

## Prerequisites

Before you deploy the Modern Data Architecture Accelerator, ensure you have the following prerequisites in place:

### AWS Account Requirements

- An AWS account with permissions to create the required resources
- IAM permissions to create roles, policies, and CloudFormation stacks
- Sufficient service quotas for the resources that will be deployed (varies based on the sample configuration you choose)

### Networking Requirements

- If you plan to deploy resources that require VPC connectivity (such as Amazon Redshift):
  - An existing VPC with at least one subnet
  - The subnet must have internet access for the deployment process
  - Security groups configured to allow necessary traffic

## GitHub Integration (if using GitHub as source)

- A GitHub account with access to the repository containing the solution code
- A GitHub CodeConnect connection established between AWS and GitHub (instructions provided in the next section)

## S3 Integration (if using S3 as source)

- An S3 bucket where you've uploaded the solution code zip file
- Appropriate permissions to access the S3 bucket from your AWS account

## Technical Knowledge

- Basic understanding of AWS services and CloudFormation
- Familiarity with data architecture concepts
- Understanding of the AWS Management Console

## Browser Requirements

- A modern web browser (Chrome, Firefox, Safari, or Edge)
- JavaScript enabled

### Tip

Before proceeding with deployment, we recommend reviewing the architecture overview and cost sections of this guide to understand the solution components and potential costs.

## AWS Account Requirements

- An AWS account with permissions to create the required resources
- IAM permissions to create roles, policies, and CloudFormation stacks
- Sufficient service quotas for the resources that will be deployed (varies based on the sample configuration you choose)

## Networking Requirements

- If you plan to deploy resources that require VPC connectivity (such as Amazon Redshift):
  - An existing VPC with at least one subnet
  - The subnet must have internet access for the deployment process
  - Security groups configured to allow necessary traffic

## GitHub Integration (if using GitHub as source)

- A GitHub account with access to the repository containing the solution code
- A GitHub CodeConnect connection established between AWS and GitHub (instructions provided in the next section)

## S3 Integration (if using S3 as source)

- An S3 bucket where you've uploaded the solution code zip file
- Appropriate permissions to access the S3 bucket from your AWS account

## Technical Knowledge

- Basic understanding of AWS services and CloudFormation
- Familiarity with data architecture concepts
- Understanding of the AWS Management Console

## Browser Requirements

- A modern web browser (Chrome, Firefox, Safari, or Edge)
- JavaScript enabled

### Tip

Before proceeding with deployment, we recommend reviewing the architecture overview and cost sections of this guide to understand the solution components and potential costs.



## Source of the Solution Code

The CloudFormation stack will deploy a CodePipeline pipeline. That pipeline needs to know where the solution source code is. Customers can have the source in a zipped file uploaded to S3 or point the stack to a GitHub repository. For GitHub repository, which is the default, follow these steps.

### Create a GitHub CodeConnect connection

The CloudFormation installer stack requires a GitHub CodeConnect connection to access the Modern Data Architecture Accelerator code on GitHub. For detailed instructions on creating a GitHub connection, see [Working with GitHub connections](#) in the AWS CodePipeline User Guide.

Follow these steps to create a connection:

1. Navigate to the AWS Developer Tools console and select **Settings > Connections**
2. Choose **Create connection**
3. Select **GitHub** as the provider and enter a connection name (e.g., "MDAA-GitHub-Connection")
4. Choose **Connect to GitHub**
5. When prompted, sign in to your GitHub account and authorize the connection
6. After authorization, the connection will be established automatically when created through the console
7. Copy the ARN of the connection for use in the CloudFormation stack

#### Note

If you create the GitHub connection using the AWS CLI instead of the console, the connection will be in a "Pending" status and you'll need to complete additional steps to update the pending connection through the console.

## Launch the installer stack

This section provides instructions for deploying the Modern Data Architecture Accelerator CloudFormation template.

### Step 1. Launch the stack

1. Sign in to the [AWS CloudFormation console](#).

3. Select the button to launch the Modern Data Architecture Accelerator CloudFormation template.

A blue button with rounded corners and a white border, containing the text "Launch solution" in white.

3. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.
4. On the **Create stack** page, verify that the correct template URL is shown in the **Amazon S3 URL** text box and choose **Next**.

## Step 2. Specify stack details


5. On the **Specify stack details** page, assign a name to your solution stack.

### Important

The stack name must be unique within your AWS account. Consider including the sample name in the stack name if you plan to deploy multiple samples, for example, `mdaa-basicdatalake-<accountname>-<region>`.

6. Under **Parameters**, review and modify the parameters for this solution template as needed. The following table describes each parameter.

| Parameter               | Default value      | Description   |
|-------------------------|--------------------|---|
| <b>OrgName</b>          | <requires input>   | The organization name. This will be prefixed to all stacks and generated resources.   |
| <b>Source</b>           | github             | Specify the source of the solution code. Valid options: * github - Use a GitHub repository * S3 - Use a zipped file in an S3 bucket |
| <b>Repository Owner</b> | aws                | The owner of the GitHub repository hosting the solution code.   |
| <b>Repository Name</b>  | modern-data-archit | The name of the GitHub repository hosting the solution code.  |

| Parameter                                | Default value  | Description  |
|--|--|--|
|  | ecture-accelerator   |  |
| <b>Branch Name</b>                       | main   | The name of the git branch to use for installation. <div><div> <b>Note</b><br/>The Branch Name parameter defaults to the latest release branch name. To determine the branch name, navigate to the Modern Data Architecture Accelerator on AWS GitHub branches page and choose the release branch you want to deploy. Release branch names align with the semantic versioning of our GitHub releases.</div></div> |
| <b>GitHub CodeConnect Connection ARN</b> | <requires input><br>(required when using GitHub as source) | The ARN of the GitHub CodeConnect connection you created in the prerequisites step.  |
| <b>Repository Bucket Name</b>            | <requires input><br>(required when using S3 as source)     | The name of the S3 bucket where the zip file of the source is stored.  |
| <b>Repository Bucket Object</b>          | release/latest.zip<br>(required when using S3 as source)   | The key of the S3 object containing the zipped source code.  |

| Parameter          | Default value     | Description  |
|--------------------|-------------------|--|
| <b>Sample Name</b> | basic<br>datalake | Sample configuration to deploy with the solution. Available options include: * basic<br>datalake * data mesh * data warehouse<br>* genai accelerator |
| <b>Subnet Id</b>   | (optional)        | The subnet ID for resources that require VPC connectivity, such as Amazon Redshift.  |
| <b>VPC Id</b>      | (optional)        | The VPC ID for resources that require VPC connectivity, such as Amazon Redshift.   |

### Step 3. Configure stack options

1. Choose **Next**.
2. On the **Configure stack options** page, choose **Next**.
3. On the **Review and create** page, review and confirm the settings. Select the box acknowledging that the template might create IAM resources.
4. Choose **Submit** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE\_COMPLETE status in approximately five minutes.

## Await initial environment deployment

Use the following procedure to ensure the Modern Data Architecture Accelerator deploys the configuration to your environment.

1. Sign in to the AWS Management Console and navigate to the **AWS CodePipeline** console. The pipeline with the name you designated should show a status of either In Progress or Complete. If In Progress, wait for the pipeline to complete.
2. Go to CloudFormation page again and you will see new Stacks corresponding to the modules of the solution starting to appear one after another.
3. The pipeline takes approximately 45 minutes to complete, though some sample configurations such as basic datalake will take less.

4. After all the stacks complete successfully, the installer pipeline will finish with a COMPLETE status.

## Terminology

- **Deployment Account** - The AWS account where deployment activities will be occurring. This includes source control for MDAA, MDAA artifact building and publishing, and MDAA execution.
- **Target Account** - The AWS account where Data Analytics Environment resources will ultimately be deployed by MDAA. Note that MDAA can execute in one account and deploy to another, or can execute and deploy to the same account.

## Prerequisites for CDK Deployment

### Manual Bootstrap (Single Deployment Source/Target Account)

1. Obtain AWS credentials for the account and place them into your credentials file or populate appropriate environment variables. These credentials should have sufficient permissions to create the following resources in the target account: \* IAM Roles \* SSM Parameters \* S3 Buckets
2. Run the CDK bootstrap command. Multiple regions can be specified simultaneously:

```
export CDK_NEW_BOOTSTRAP=1
cdk bootstrap aws://<AWS Account Number>/<Target Region>...
```

For example:

```
export CDK_NEW_BOOTSTRAP=1
cdk bootstrap aws://123456789012/ca-central-1 aws://123456789012/us-east-1
```

### Manual Bootstrap (Single Deployment Source Account and One or More Target Accounts)

This procedure should be executed for each target account. This procedure bootstraps the target account while establishing trust with the source account (where deployments will be triggered)

Obtain AWS credentials for the target account and place them into your credentials file or populate appropriate environment variables.

These credentials should have sufficient permissions to create the following resources in the target account: \* IAM Roles \* SSM Parameters \* S3 Buckets

Run the CDK bootstrap command. Multiple regions can be specified simultaneously:

```
export CDK_NEW_BOOTSTRAP=1
cdk bootstrap --cloudformation-execution-policies <CDK Deployment IAM Policy Arns> --
trust <Source Account Number> aws://<Target AWS Account Number>/<Target Region>...
```

Note that the permissions specified with cloudformation-execution-policies are granted to CloudFormation during deployment into the account. These permissions should be sufficient to deploy MDAA resources, but not overly permissive.

## Deployment

### Deployment Overview

The following are procedures which can be executed in order to manually deploy MDAA to target accounts. These procedures assume that the appropriate preparations have been made within the organizations accounts.

### Deployment Patterns

MDAA may be deployed using a number of patterns:

- Same Deployment Source and Target Account (Centralized Data Environment)
- Single Deployment Source account, One or More Separate Target Accounts (Centralized deployment governance, decentralized Data Environments)

### Deployment Preparation

#### Node Installation

Install a version of Node.js using a method appropriate to your system. MDAA requires nodejs 22.x and npm/npx version 10.x or greater.

## Environment Setup

Ensure your credentials are populated either in your environment or in your `~/.aws/credentials` file. Also, ensure your AWS region is specified either in your environment or in your `~/.aws/config` file:

```
[default]
region=ca-central-1
```

## Deployment from Locally Cloned Source Code

As of MDAA 0.40, deployment from locally cloned MDAA source code is the preferred deployment mode, as it avoids requiring MDAA NPM packages to be published. As of 0.43, specification of the `-l` flag is no longer required to force local execution mode. Modules which are available in locally cloned MDAA source code will be used. Otherwise, the required packages will be installed via NPM.

1. Clone MDAA repo.
2. Run `<path_to_cloned_repo>/bin/mdaa -c <path_to_mdaa_yaml> <cdk action>`
  - MDAA will run `npm install` at the root of the cloned repo to install CDK and all necessary third-party dependencies.
  - MDAA will locate its own modules within the local source code repo. Note that specifying specific MDAA versions in `local_mode` will result in NPM packages being installed.

Additional MDAA CLI commands:

Use the `-h` parameter to print a list of all MDAA CLI parameters

```
<path_to_cloned_repo>/bin/mdaa -h
```

Use the `-c` parameter to specify a config file. Otherwise MDAA CLI will attempt to use `mdaa.yaml` from the local directory.

```
<path_to_cloned_repo>/bin/mdaa -c <optional-path-to-mdaa-config-file> <cdk action>
```

Specify a `<cdk action>`, which MDAA CLI will run against every configured module/CDK app:

```
<path_to_cloned_repo>/bin/mdaa <cdk action>
```

To list (Terraform validate) all stacks:

```
<path_to_cloned_repo>/bin/mdaa list
```

To synth (Terraform validate) all stacks:

```
<path_to_cloned_repo>/bin/mdaa synth
```

To diff (Terraform plan) all stacks:

```
<path_to_cloned_repo>/bin/mdaa diff
```

To deploy (Terraform deploy) all stacks:

```
<path_to_cloned_repo>/bin/mdaa deploy
```

To deploy only env=dev modules/stacks:

```
<path_to_cloned_repo>/bin/mdaa deploy -e dev
```

To deploy only domain1 and domain2 modules/stacks:

```
<path_to_cloned_repo>/bin/mdaa deploy -d domain1, domain2
```

To deploy only the test\_roles\_module and test\_datalake\_module modules/stacks:

```
<path_to_cloned_repo>/bin/mdaa deploy -m test_roles_module, test_datalake_module
```

Any CLI params not recognized by MDAA CLI will be pushed down to the CDK/Terraform CLI. In this example, --no-rollback will be pushed down to CDK:

```
<path_to_cloned_repo>/bin/mdaa deploy --no-rollback
```

## Deployment from Published NPM Packages

MDAA can be installed from a private NPM package repo, and will also attempt to install MDAA modules from a private NPM repo. This is necessary if specific MDAA versions are specified in mdaa.yaml.

Ensure that your private NPM repo is accessible and contains the appropriate MDAA NPM artifacts. If using a localhost based NPM repo (such as Verdaccio), ensure it is running on localhost and



updated with the latest MDAA packages from S3 (See [PREDEPLOYMENT](#)). When executed from its NPM package, MDAA will also attempt to NPM install each MDAA module from NPM repo.

Install MDAA from your private NPM repository using:

Global Installation:

```
npm install -g @aws-mdaa/cli
```

Then, MDAA can be executed globally:

```
mdaa -h
```

Optionally, both the MDAA CLI can be instead npm installed in a local directory:

```
npm install @aws-mdaa/cli
```

MDAA commands can then be run within the local directory using NPX.

```
npx mdaa -h
```

## Deployment of MDAA Modules/CDK Apps using CDK CLI

MDAA Modules are developed as independant CDK apps which can be directly executed using the CDK CLI. This is generally useful for development and troubleshooting directly against the MDAA codebase, but is not recommended for normal use.

To execute MDAA Modules/CDK apps using the CDK CLI:

1. Clone the MDAA source repo.
2. At the root of the repo, run npm install to install all packages required across all modules.
3. Change directory to the MDAA Modules/CDK apps source code directory (typically under packages/apps/< module category >/< module >)
4. Run the following CDK commands with the required context:

```
cdk synth -c org=<organization> -c env=<dev|test|prod> -c domain=<domain name>  
-c module_configs=<app_config_paths> -c tag_configs=<tag_config_paths> -c  
module_name=<module_name>
```

```
cdk synth -c org="sample-org" -c env="dev" -c domain="mdaa1" -c  
module_configs="warehouse.yaml" -c tag_configs="tags.yaml" -c module_name="testing"
```

## Required Context

The following context values are required for all modules. Note that additional context values may be required if context is referenced from within the module/app config.

- **org** - Name of the organization
- **env** - Name of the target environment (ie. dev/test/prod)
- **domain** - Name of the deployment domain (allows multiple deployments in same org/env/account)
- **module\_name** - Name of the MDAA module (allows multiple deployments of the same CDK app within same org/domain/env)
- **module\_configs** - Comma separated list of paths to one or more app config files. Multiple config files will be merged, with later-listed config files taking precedence over earlier-listed config files.
- **tag\_configs** - Comma separated list of paths to one or more tag config files. Multiple config files will be merged, with later-listed config files taking precedence over earlier-listed config files.

## Deploying starter packages

The Modern Data Architecture Accelerator provides several starter packages that you can deploy to quickly establish common data architecture patterns. Each starter package includes pre-configured components that work together to create a complete solution for specific use cases.

When deploying starter packages, consider the following:

- **Environment requirements:** Review the prerequisites for each package to ensure your AWS environment meets the necessary requirements
- **Configuration options:** Each package includes sample configuration files that can be customized to match your specific needs
- **Deployment methods:** Packages can be deployed using the MDAA CLI, AWS CDK directly, or through the CloudFormation installer
- **Post-deployment steps:** Most packages require additional configuration after deployment to fully enable all features

The following sections provide detailed information about each available starter package, including architecture components, deployment instructions, and usage guidelines.

## AI/ML Starter Package

The AI/ML Starter Package establishes a comprehensive environment for developing, training, and deploying machine learning models at scale.

This implementation demonstrates AWS best practices for creating an enterprise-grade data science platform. It combines data lake capabilities with SageMaker Studio to provide data scientists with the tools they need while maintaining appropriate governance and security controls.

This architecture is particularly effective when:

1. You need to enable data science teams to rapidly develop and deploy ML models.
2. Your organization requires governed access to data and model resources.

Deploy this package when you need a scalable, secure foundation that supports your organization's machine learning and AI initiatives.

The AI/ML Starter Package provides a comprehensive environment for developing, training, and deploying machine learning models. This package is organized into three domains that work together to create a complete data science platform:

### Shared Domain Components

- **IAM Roles** - Secure access controls for data science teams
- **Data Lake** - S3 buckets for storing training data and model artifacts
- **Glue Data Catalog** - KMS-encrypted metadata management
- **Lake Formation** - Fine-grained access control for data assets
- **Athena Workgroups** - SQL-based data exploration capabilities
- **Audit Components** - CloudTrail integration for comprehensive governance

### DataOps Domain Components

- **DataOps Projects** - Shared resources for data engineering workflows

- **Glue Crawlers** - Automated metadata discovery and schema management

## DataScience Domain Components

- **SageMaker Studio** - Fully managed development environment for ML
- **Team Workspaces** - Isolated environments for data science teams
- **Jupyter Notebooks** - Pre-configured templates for common ML tasks
- **Model Registry** - Version control for ML models
- **Training Pipelines** - Automated workflows for model development
- **Deployment Infrastructure** - Endpoints for model serving

This package accelerates your AI/ML initiatives by providing a ready-to-use environment with AWS best practices built in. It's ideal for organizations looking to establish or enhance their machine learning capabilities with a secure, scalable foundation.

## Deployment Instructions

Step-by-step guide for deploying the AI/ML Starter Package

You can deploy the AI/ML Starter Package using one of two methods: 1. CloudFormation Installer Method (recommended for most users) 2. Manual CLI Deploy Method (for advanced customization)

### Method 1: CloudFormation Installer Method

#### Prerequisites

Before deploying using the CloudFormation installer, ensure you have:

1. An AWS account with permissions to create the required resources
2. A GitHub CodeConnect connection (if using GitHub as source) or an S3 bucket with the solution code (if using S3 as source)
3. A VPC with at least one subnet (required for SageMaker Studio)

## Deployment Steps

### Step 1: Launch the CloudFormation stack

## Step 2: Configure the stack

1. Assign a unique name to your stack (e.g., `mdaa-aiml-<accountname>-<region>`).
2. Under Parameters:
  - Enter your organization name in the `OrgName` field
  - Select `github` as the Source (default)
  - Verify the Repository Owner is `aws` and Repository Name is `modern-data-architecture-accelerator`
  - Enter your GitHub CodeConnect Connection ARN
  - Select `basic data science` as the Sample Name
  - Provide Subnet ID and VPC ID (required for SageMaker Studio)
3. Choose Next, review the settings, and acknowledge that the template might create IAM resources
4. Choose Submit to deploy the stack

## Step 3: Monitor the deployment

1. Navigate to the AWS CodePipeline console to monitor the deployment progress
2. The pipeline will show a status of either `In Progress` or `Complete`
3. The deployment typically takes about 30-45 minutes for the AI/ML configuration

## Step 4: Verify deployment

Check that all CloudFormation stacks have completed successfully and the installer pipeline shows a `COMPLETE` status

## Method 2: Manual CLI Deploy Method

### Prerequisites

Before deploying the AI/ML Starter Package using the CLI method, ensure you have:

1. AWS CLI configured with appropriate credentials
2. Node.js 16.x or later installed
3. AWS CDK installed (`npm install -g aws-cdk`)
4. CDK bootstrapped in your target account and region

## 5. A VPC with at least one subnet (required for SageMaker Studio)

### Deployment Steps

#### Step 1: Clone the MDAA repository

```
git clone https://github.com/aws/modern-data-architecture-accelerator.git &&
cd modern-data-architecture-accelerator
```

#### Step 2: Configure your deployment

- Copy the sample configuration files:

```
cp -r sample_configs/basic_datascience_platform my_aiml_config
cd my_aiml_config
```

- Edit the `mdaa.yaml` file to set your organization name, datascience team name and VPC/subnet information:

```
organization: <your-org-name>
context:
  vpc_id: <your vpc id>
  subnet_id: <your subnet id>
  datascience_team_name: <your datascience team name>
```

#### Step 3: Deploy the solution \* Ensure you are authenticated to your target AWS account.

- Optionally, run the following command to understand what stacks will be deployed:

```
../bin/mdaa ls
```

- Optionally, run the following command to review the produced templates:

```
../bin/mdaa synth
```

- Run the following command to deploy all modules:

```
../bin/mdaa deploy
```

**Step 4: Verify deployment** \* Check the AWS CloudFormation console to ensure all stacks have been created successfully \* Verify the SageMaker Studio Domain, IAM roles, and other resources have been created

## Usage Instructions

How to effectively use the AI/ML Starter Package after deployment

Once the MDAA deployment is complete, follow these steps to interact with the AI/ML platform:

### Initial Setup and Data Access

#### 1. Create sample data for testing

- Check the DATASETS.md file in the sample\_configs directory for instructions on creating a sample\_data folder
- Alternatively, prepare your own data files for upload

#### 2. Assume the shared-roles-data-admin role

- This role is configured with AssumeRole trust to the local account by default
- It has write access to the data lake

#### 3. Upload sample data to the transformed bucket

- Upload the sample\_data folder and contents to the transformed bucket

#### 4. Run the Glue Crawler

- In the AWS Glue Console, trigger/run the Glue Crawler
- Once successful, view the Crawler's CloudWatch logs to observe that tables were created

## Using SageMaker Studio

#### 1. Assume the data-scientist role

- This role is configured with AssumeRole trust to the local account by default
- Important: The role session name must match the userid specified in the datascience-team.yaml configuration

#### 2. Access SageMaker Studio

- Navigate to the Amazon SageMaker console

- Go to the Domain section and find the deployed SageMaker Studio Domain
- Launch the user profile matching your role session name/userid
- SageMaker Studio should launch

### 3. Work with data in Athena

- In the Athena Query Editor, select the MDAA-deployed Workgroup from the dropdown list
- The tables created by the crawler should be available for query under the MDAA-created Database
- Run queries to explore and analyze your data

### 4. Develop ML models

- Use the pre-configured Jupyter notebooks in SageMaker Studio
- Access your data through the Athena integration
- Train models using SageMaker's built-in algorithms or custom code
- Deploy models to SageMaker endpoints for inference

For more detailed information about the configuration files and their purposes, refer to the README.md file in the sample\_configs/basic\_datascience\_platform directory of the MDAA repository.

## Datalake Starter Package

The Datalake Starter Package establishes a robust foundation for storing and managing large volumes of data in its native format.

This S3 Data Lake implementation demonstrates best practices for creating an enterprise data lake on AWS. Access to the data lake can be granted to IAM and federated principals, with comprehensive access controls.

This architecture is particularly effective when:

1. You need to store both structured and unstructured data in a centralized repository.
2. Your organization requires governed access to data lake resources.

Deploy this package when you need a scalable, secure foundation that supports your organization's data storage and analytics requirements.



The Datalake Starter Package provides a complete foundation for enterprise data management with a secure, scalable architecture organized into multiple domains:

## Shared Domain Components

- **IAM Roles** - Predefined roles for data administrators, users, and ETL processes with least-privilege permissions
- **S3 Data Lake** - Multi-zone storage architecture with raw and transformed data buckets
- **Access Policies** - Granular bucket policies with prefix-based access controls
- **Glue Data Catalog** - KMS-encrypted metadata repository for data assets
- **Lake Formation** - Configured for IAM-based access control to data resources
- **Athena Workgroups** - Query environment with predefined settings for data analysis
- **Audit Components** - CloudTrail integration with secure S3 buckets for comprehensive audit trails

## DataOps Domain Components

- **DataOps Projects** - Shared resources for data engineering workflows
- **Glue Crawlers** - Automated metadata discovery for data assets

This architecture provides a robust foundation for organizations that need to:

1. Centralize data storage with appropriate security controls
2. Implement governance and compliance requirements
3. Enable self-service data access for various user roles
4. Support both batch and interactive data processing

The Datalake Starter Package follows AWS best practices for security, scalability, and operational excellence, making it an ideal starting point for organizations building their modern data architecture.

## Deployment Instructions

Step-by-step guide for deploying the Datalake Starter Package

You can deploy the Datalake Starter Package using one of two methods: 1. CloudFormation Installer Method (recommended for most users) 2. Manual CLI Deploy Method (for advanced customization)

## Method 1: CloudFormation Installer Method

### Prerequisites

Before deploying using the CloudFormation installer, ensure you have:

1. An AWS account with permissions to create the required resources
2. A GitHub CodeConnect connection (if using GitHub as source) or an S3 bucket with the solution code (if using S3 as source)

### Deployment Steps

#### Step 1: Launch the CloudFormation stack

#### Step 2: Configure the stack

1. Assign a unique name to your stack (e.g., `mdaa-basicdatalake-<accountname>-<region>`)
2. Under Parameters:
  - Enter your organization name in the `OrgName` field
  - Select `github` as the Source (default)
  - Verify the Repository Owner is `aws` and Repository Name is `modern-data-architecture-accelerator`
  - Enter your GitHub CodeConnect Connection ARN
  - Select `basic_datalake` as the Sample Name
3. Choose Next, review the settings, and acknowledge that the template might create IAM resources
4. Choose Submit to deploy the stack

#### Step 3: Monitor the deployment

1. Navigate to the AWS CodePipeline console to monitor the deployment progress
2. The pipeline will show a status of either `In Progress` or `Complete`

3. The deployment typically takes about 15-20 minutes for the basic datalake configuration

## Step 4: Verify deployment

Check that all CloudFormation stacks have completed successfully and the installer pipeline shows a COMPLETE status

## Method 2: Manual CLI Deploy Method

### Prerequisites

Before deploying the Datalake Starter Package using the CLI method, ensure you have:

1. AWS CLI configured with appropriate credentials
2. Node.js 16.x or later installed
3. AWS CDK installed (`npm install -g aws-cdk`)
4. CDK bootstrapped in your target account and region

### Deployment Steps

#### Step 1: Clone the MDAA repository

```
git clone https://github.com/aws/modern-data-architecture-accelerator.git &&
cd modern-data-architecture-accelerator
```

#### Step 2: Configure your deployment

- Copy the sample configuration files:

```
cp -r sample_configs/basic_datalake my_datalake_config
cd my_datalake_config
```

- Edit the `mdaa.yaml` file to set your organization name:

```
organization: <your-org-name>
```

**Step 3: Deploy the solution** \* Ensure you are authenticated to your target AWS account.

- Optionally, run the following command to understand what stacks will be deployed:

```
../bin/mdaa ls
```

- Optionally, run the following command to review the produced templates:

```
../bin/mdaa synth
```

- Run the following command to deploy all modules:

```
../bin/mdaa deploy
```

**Step 4: Verify deployment** \* Check the AWS CloudFormation console to ensure all stacks have been created successfully \* Verify the S3 buckets, Glue Crawler, IAM roles, and other resources have been created

## Usage Instructions

How to effectively use the Datalake Starter Package after deployment

Once the MDAA deployment is complete, follow these steps to interact with the data lake:

### Initial Setup and Data Upload

#### 1. Create sample data for testing

- Check the DATASETS.md file in the sample\_configs/basic\_datalake directory for instructions on creating a sample\_data folder
- Alternatively, prepare your own data files for upload

#### 2. Assume the data-admin role

- This role is configured with AssumeRole trust to the local account by default
- Note that this role is the only role configured with write access to the data lake
- All other roles (including existing administrator roles in the account) will be denied write access

#### 3. Upload sample data to the transformed bucket

- Upload the sample\_data folder and contents to the transformed bucket.

## Data Discovery and Querying

### 1. Run the Glue Crawler

- In the AWS Glue Console, locate the crawler created by the deployment
- Trigger/run the Glue Crawler
- Once successful, view the Crawler's CloudWatch logs to observe that tables were created

### 2. Assume the data-user role

- This role is configured with AssumeRole trust to the local account by default
- It has read-only access to the data lake

### 3. Query data using Athena

- In the Athena Query Editor, select the MDAA-deployed Workgroup from the dropdown list
- The tables created by the crawler should be available for query under the MDAA-created Database

For more detailed information about the configuration files and their purposes, refer to the README.md file in the sample\_configs/basic\_datalake directory of the MDAA repository.

## GenAI Accelerator Starter Package

The GenAI Accelerator Starter Package delivers a production-ready foundation for building sophisticated AI agents using Amazon Bedrock. Out of the box, it demonstrates a customer support assistant that can understand queries, search knowledge bases, and take actions - but this is just the beginning.

Built on AWS best practices, this package provides the building blocks to create any type of agentic AI application. Whether you need sales assistants, technical troubleshooters, or domain-specific experts, the modular architecture adapts to your unique requirements.

This architecture is particularly effective when:

1. You need to deploy intelligent agents with autonomous decision-making capabilities that can perform complex tasks.
2. Your organization requires governed access to AI resources with appropriate content safety measures.

Deploy this package when you need a scalable, secure foundation that supports your organization's agentic AI initiatives with enterprise-grade controls.

The GenAI Accelerator Starter Package provides a complete environment for developing and deploying agentic AI applications. This package is organized into multiple domains that work together to create a comprehensive AI platform:

## Shared Domain Components

- **IAM Roles** - Comprehensive roles including data-admin with Bedrock permissions, agent execution roles, and Lambda execution roles
- **Data Lake** - KMS-encrypted S3 buckets for knowledge base data sources and document storage
- **Access Policies** - Granular permissions for AI resource management and data access

## GenAI Domain Components

- **Bedrock Agents** - Intelligent agents with custom instructions and action groups
- **Knowledge Bases** - RAG-enabled knowledge bases with vector stores for efficient retrieval
- **Lambda Functions** - Custom functions for agent action groups and document processing
- **Guardrails** - Content safety measures to ensure appropriate AI responses
- **Vector Stores** - Efficient knowledge retrieval using Amazon OpenSearch Serverless

This package accelerates your agentic AI initiatives by providing a ready-to-use environment with AWS best practices built in. It's ideal for organizations looking to establish or enhance their AI capabilities with autonomous agents, intelligent task execution, and enterprise-grade AI governance.

## Deployment Instructions

Step-by-step guide for deploying the GenAI Accelerator Starter Package

You can deploy the GenAI Accelerator Starter Package using Manual CLI Deploy Method

### Manual CLI Deploy Method

#### Prerequisites

Before deploying the GenAI Accelerator Starter Package using the CLI method, ensure you have:

1. AWS CLI configured with appropriate credentials
2. Node.js 16.x or later installed
3. AWS CDK installed (`npm install -g aws-cdk`)
4. CDK bootstrapped in your target account and region
5. A VPC with at least two private subnets (required for OpenSearch Serverless)
6. Access to Bedrock foundation models in your region

## Deployment Steps

### Step 1: Clone the MDAA repository

```
git clone https://github.com/aws/modern-data-architecture-accelerator.git &&
cd modern-data-architecture-accelerator
```

### Step 2: Configure your deployment

- Copy the sample configuration files:

```
cp -r sample_configs/genai_accelerator my_genai_config
cd my_genai_config
```

- Edit the `mdaa.yaml` file to set your organization name and VPC/subnet information:

```
organization: <your-unique-org-name>
context:
  vpc_id: <your vpc id>
  subnet_id_1: <your subnet id 1>
  subnet_id_2: <your subnet id 2>
  subnet_id_3: <your subnet id 3>
  llm_model: <your_model> # e.g. anthropic.claude-3-5-sonnet-20240620-v1:0 or an ARN
  kb_embedding_model: <kb_embedding_model> # e.g. amazon.titan-embed-text-v2:0
  kb_parsing_model: <kb_parsing_model> # e.g. anthropic.claude-3-5-sonnet-20240620-v1:0
```

## Important Notes:

- **Cross-region inference:** Use inference profile ARN in `llm_model` (e.g., `arn:aws:bedrock:us-east-1:<account_id>:inference-profile/anthropic.claude-3-7-sonnet-20250219-v1:0`)
- **Single region:** Use model ID directly

**Step 3: Deploy the solution** \* Ensure you are authenticated to your target AWS account.

- Optionally, run the following command to understand what stacks will be deployed:

```
../bin/mdaa ls
```

- Optionally, run the following command to review the produced templates:

```
../bin/mdaa synth
```

- Run the following command to deploy all modules:

```
../bin/mdaa deploy
```

**Step 4: Verify deployment** \* Check the AWS CloudFormation console to ensure all stacks have been created successfully \* Verify the Bedrock agents, knowledge bases, S3 buckets, and other resources have been created

## Usage Instructions

How to effectively use the GenAI Accelerator Starter Package after deployment

Once the MDAA deployment is complete, follow these steps to interact with the GenAI platform:

### Initial Setup and Document Upload

#### 1. Request Bedrock model access

- Navigate to the Amazon Bedrock console
- Go to Model access and request access to the foundation models you plan to use
- Wait for approval (typically immediate for most models)



## 2. Assume the data-user role

- This role is configured with AssumeRole trust to the local account by default
- It has permissions to upload documents to the knowledge base S3 buckets

## 3. Upload documents to knowledge base buckets

- Upload documents to the support-docs prefix for customer support materials
- Upload documents to the product-docs prefix for product documentation
- Ensure documents are uploaded with KMS encryption

## Using Bedrock Agents

### 1. Assume the data-admin role

- This role is configured with AssumeRole trust to the local account by default
- It has comprehensive permissions for managing Bedrock resources

### 2. Sync knowledge bases

- Navigate to the Amazon Bedrock console
- Go to Knowledge Bases and select your deployed knowledge base
- Check if your documents are automatically synced to the knowledge base

### 3. Test the Bedrock Agent

- In the Amazon Bedrock console, go to Agents
- Select the customer-support-agent
- Use the Test Agent interface to interact with the agent
- Ask questions related to the documents you uploaded

### 4. Monitor agent performance

- Use CloudWatch logs to monitor agent interactions
- Review Bedrock traces for detailed execution information
- Monitor Lambda function logs for action group executions

## Advanced Usage

### 1. Customize agent behavior

- Modify the agent instructions in the bedrock-builder.yaml configuration

### 2. Update Lambda functions for custom action groups

- Adjust guardrails for content safety requirements

## 2. Expand knowledge bases

- Add additional data sources to existing knowledge bases
- Create new knowledge bases for different domains
- Configure custom transformation Lambda functions

For more detailed information about the configuration files and their purposes, refer to the README.md file in the sample\_configs/genai\_accelerator directory of the MDAA repository.

## Governed Lakehouse Starter Package

The Governed Lakehouse Starter Package delivers an enterprise-ready data lakehouse with comprehensive governance capabilities using Amazon DataZone and AWS Lake Formation. This package provides fine-grained access control, data product management, and multi-team collaboration features essential for modern data governance.

Built on AWS best practices, this package combines the flexibility of a data lake with the governance and structure of a data warehouse. It enables organizations to implement data mesh architectures while maintaining centralized governance and compliance controls.

This architecture is particularly effective when:

1. You need fine-grained access control for structured data across multiple teams and projects.
2. Your organization requires comprehensive data governance with data product management capabilities.

Deploy this package when you need a scalable, governed foundation that supports enterprise data governance requirements with DataZone integration and Lake Formation security controls.

The Governed Lakehouse Starter Package provides a complete environment for enterprise data governance with comprehensive access controls. This package is organized into multiple domains that work together to create a governed data platform:

### Shared Governance Components

- **IAM Roles** - Comprehensive roles including data-admin with Lake Formation permissions and data-user roles with fine-grained access controls

- **Lake Formation Settings** - Configured to disable automatic IAM grants and enable fine-grained access control
- **Glue Data Catalog** - KMS-encrypted metadata repository with governance controls

## Data Domain Components

- **S3 Data Lake** - Multi-tier storage architecture with raw, transformed, and curated data buckets
- **KMS Encryption** - Customer-managed keys for data-at-rest encryption
- **Bucket Policies** - Fine-grained access controls integrated with Lake Formation

## Governance Domain Components

- **DataZone Domain** - Data product management and discovery platform
- **Environment Blueprints** - Standardized environments for data producers and consumers
- **Project Templates** - Reusable configurations for different team types

## DataOps Domain Components

- **DataOps Projects** - Multiple project configurations for data producers and consumers
- **Glue Crawlers** - Automated metadata discovery with governance integration
- **Lake Formation Permissions** - Database and table-level access controls

This package accelerates your data governance initiatives by providing a ready-to-use environment with AWS best practices built in. It's ideal for organizations looking to establish or enhance their data governance capabilities with enterprise-grade controls, data product management, and multi-team collaboration features.

## Deployment Instructions

Step-by-step guide for deploying the Governed Lakehouse Starter Package

You can deploy the Governed Lakehouse Starter Package using Manual CLI Deploy Method

## Manual CLI Deploy Method

### Prerequisites

Before deploying the Governed Lakehouse Starter Package using the CLI method, ensure you have:

1. AWS CLI configured with appropriate credentials
2. Node.js 16.x or later installed
3. AWS CDK installed (`npm install -g aws-cdk`)
4. CDK bootstrapped in your target account and region

### Deployment Steps

#### Step 1: Clone the MDAA repository

```
git clone https://github.com/aws/modern-data-architecture-accelerator.git &&  
cd modern-data-architecture-accelerator
```

#### Step 2: Configure your deployment

- Copy the sample configuration files:

```
cp -r sample_configs/governed_lakehouse my_governed_lakehouse  
cd my_governed_lakehouse
```

- Edit the `mdaa.yaml` file to set your organization name:

```
organization: <your-unique-org-name>
```

#### Step 3: Deploy the solution \* Ensure you are authenticated to your target AWS account.

- Optionally, run the following command to understand what stacks will be deployed:

```
../bin/mdaa ls
```

- Optionally, run the following command to review the produced templates:

```
../bin/mdaa synth
```

- Run the following command to deploy all modules:

```
../bin/mdaa deploy
```

**Step 4: Verify deployment** \* Check the AWS CloudFormation console to ensure all stacks have been created successfully \* Verify the DataZone domain, Lake Formation settings, S3 buckets, and other resources have been created

## Usage Instructions

How to effectively use the Governed Lakehouse Starter Package after deployment

Once the MDAA deployment is complete, follow these steps to interact with the governed lakehouse:

### Initial Setup and Data Upload

#### 1. Assume the data-admin role

- This role is configured with AssumeRole trust to the local account by default
- It has comprehensive permissions for Lake Formation and DataZone administration
- Note: This role is the only role configured with write access to the data lake

#### 2. Prepare sample data for testing

- Check the DATASETS.md file in the sample\_configs/governed\_lakehouse directory for instructions on creating sample data
- Alternatively, prepare your own structured data files for upload

#### 3. Upload data to the data lake buckets

- Upload data to the appropriate tier based on processing stage:
  - Raw data: \${org}-\${env}-data1-datalake-raw
  - Transformed data: \${org}-\${env}-data1-datalake-transformed
  - Curated data: \${org}-\${env}-data1-datalake-curated
- Ensure data is uploaded with KMS encryption

## Data Discovery and Governance

### 1. Run Glue Crawlers

- Navigate to the AWS Glue Console
- Trigger the Glue Crawlers created by the deployment
- Monitor CloudWatch logs to verify table creation and metadata discovery

### 2. Access DataZone Portal

- Navigate to the Amazon DataZone console
- Launch the DataZone Domain Portal
- Explore the data governance capabilities including data product creation and discovery

### 3. Configure Lake Formation Permissions

- Use Lake Formation to grant fine-grained permissions to different user roles
- Configure database and table-level access controls as needed
- Set up data filters for row and column-level security

## Multi-Team Data Access

### 1. Assume appropriate user roles

- Use data-user roles for read-only access to governed data
- Each role has specific Lake Formation permissions configured

### 2. Query data using Athena

- Access Athena through the configured workgroups
- Query tables that have been cataloged and for which you have Lake Formation permissions
- Use DataZone to discover and subscribe to data products

### 3. Manage data products

- Create data products in DataZone for different business domains
- Publish data assets for discovery by other teams
- Subscribe to data products created by other teams

For more detailed information about the configuration files and their purposes, refer to the README.md file in the sample\_configs/governed\_lakehouse directory of the MDAA repository.

# Update the solution

If you deployed a previous version of this solution, you must first uninstall your previous deployment and then deploy the new version.

# Troubleshooting

Known issue resolution provides instructions to mitigate known errors. If these instructions don't address your issue, see the [Contact AWS Support](#) section for instructions on opening an AWS Support case for this solution.

## Known issue resolution

### Failed to upload data in S3 bucket

**Issue:** Unable to Upload New Data

**Reason:** For security purposes, data upload permissions to the bucket are restricted to users with the data-admin role. Standard admin users do not have upload privileges.

**Resolution:**

1. Go to IAM console and find the role that ends with data-admin
2. Switch to the data-admin role in that account
3. Add the required data in S3 transformed bucket
4. Switch back to the main role
5. Run the crawler to index the new data

### Data Science Configuration Deployment Failure

**Issue:** The deployment failed while deploying basic\_datascience configuration

**Reason:** To set up a data science environment in SageMaker Studio, a unique user profile is needed with a unique name. This profile will grant the user permission to access and launch SageMaker Studio resources.

**Resolution:** User Profile Name Issues:

- Modify the user profile name in datascience-team.yaml
- Please change the <my-own-data-science-profile-name> to something custom that you can identify



```
userProfiles:
  # The key/name of the user profile should be specified as follows:
  # If the Domain is in SSO auth mode, this should map to an SSO User ID.
  # If in IAM mode, this should map to Session Name portion of the
aws:userid variable.
  "<my-own-data-science-profile-name>":
    # Required if the domain is in IAM AuthMode. This is the role
    # from which the user will launch the user profile in Studio.
    # The role's id will be combined with the userid
    # to grant the user access to launch the user profile.
    userRole:
      id: generated-role-id:data-user
```

## Lake Formation Data Lake Deployment Issues

**Issue:** The following error messages

```
Reading config from /Users/xxx/Documents/MDAA/config/lakeformation_datalake/
datascience/datascience-team.yaml.
Error: ENOENT: no such file or directory
```

**Reason:** LakeFormation expects a datascience.yaml to create datascience related configurations

### Resolution:

1. Create a folder named datascience inside lakeformation\_datalake folder
2. Create a file named datascience-team.yaml inside the folder
3. Please add the sample configuration values as below:

```
# List of roles which will be provided admin access to the team resources
dataAdminRoles:
- id: generated-role-id:data-admin

# List of roles which will be provided usage access to the team resources
# Can be either directly referenced Role Arns, Role Arns via SSM Params,
# or generated roles created using the MDAA roles module.
teamUserRoles:
- id: generated-role-id:data-user
```

```
# The role which will be used to execute Team SageMaker resources (Studio Domain Apps,
  SageMaker Jobs/Pipelines, etc)
teamExecutionRole:
id: generated-role-id:team-execution

# The team Studio Domain config
studioDomainConfig:
authMode: IAM
vpcId: "{{context:vpc_id}}"
subnetIds:
- "{{context:subnet_id}}"
notebookSharingPrefix: sagemaker/notebooks/

# List of Studio user profiles which will be created.
userProfiles:
# The key/name of the user profile should be specified as follows:
# If the Domain is in SSO auth mode, this should map to an SSO User ID.
# If in IAM mode, this should map to Session Name portion of the aws:userid variable.
"<my-own-data-science-profile-name>":
# Required if the domain is in IAM AuthMode. This is the role
# from which the user will launch the user profile in Studio.
# The role's id will be combined with the userid
# to grant the user access to launch the user profile.
userRole:
id: generated-role-id:data-user
```

## Failed to resolve context: vpc\_id

**Issue:** Encounters the following error message

```
Error: Failed to resolve context: vpc_id
at MdaaConfigRefValueTransformer.parseContext (/Users/xxx/Documents/MDAA/packages/
utilities/mdaa-config/lib/config.ts:193:19)
at /Users/xxxx/Documents/MDAA/packages/utilities/mdaa-config/lib/config.ts:165:38
```

**Reason:** vpc\_id and subnet\_id are needed to create a secure data environment within the vpc

### Resolution:

1. Go to mdaa.yaml file
2. Please check if you have vpc\_id configured in the file
3. The below values should go after organization in the config file.

#### 4. Please run the deployment again after the values are changed

```
# TODO: Set an appropriate, unique organization name
# Failure to do so may result in global naming conflicts.
organization: trial-datalake-lk
context:
  vpc_id: vpc-000000090000
  subnet_id: subnet-09090909090
```

#### **Issue:** Failure in deploying Generative AI Accelerator

##### Error Message:

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker
daemon running?
[100%] fail: docker build --tag
cdkasset-7a1e3989751f91a191cd33edf97f22ef63c06ad34f01895a7af11a3e32e3a97a . exited
with error code 1
```

##### **Resolution:**

- Ensure Docker is installed and running
- Verify Docker daemon is active before deployment
- Check Docker configuration settings

## Contact AWS Support

If you have [AWS Developer Support](#), [AWS Business Support](#), or [AWS Enterprise Support](#), you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

### Create case

1. Sign in to [Support Center](#).
2. Choose **Create case**.

## How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

## Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that AWS Support needs to process the request.

## Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

## Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

# Uninstall the solution

The following manual steps would need to be followed in order to properly uninstall MDAA:

1. Delete all the stacks that start with the organization name
2. Delete also the stack that the installer template created
3. Delete all the S3 buckets with the organization name as prefix in the bucket name

# Use the solution

This section explains how to use MDAA's configuration capabilities to manage your data platform architecture.

## Using configuration files

MDAA is configured using YAML configuration files. The main CLI configuration file (typically 'mdaa.yaml') specifies:

- Global settings
- Domains
- Environments
- Modules to be deployed

Configuration layouts are flexible and can be:

- Concentrated in a single MDAA config file
- Spread across multiple config files by domain
- Organized by line of business
- Separated by environment

## Configuration File Structures

You can organize your configurations in several ways.

### Single Domain, Shared CDK Apps Configs Across Envs

```
root_folder
#   mdaa.yaml
#   tags.yaml
#
####  domain1
      roles.yaml
      datalake.yaml
```

## Single Domain, Separate CDK Apps Configs Across Envs

```
root_folder
#   mdaa.yaml
#   tags.yaml
#
####  domain1
      ####  dev
      #   #   dev_roles.yaml
      #   #   dev_datalake.yaml
      #
      ####  test
      #   #   test_roles.yaml
      #   #   test_datalake.yaml
```

## Working with dynamic references

MDAA allows use of dynamic references in configuration files, building on CloudFormation Dynamic References:

```
# Example Config File w/Dynamic References
vpcId: "{{resolve:ssm:/path/to/ssm/param}}"
sensitive_value: "{{resolve:ssm-secure:parameter-name:version}}"
db_username: "{{resolve:secretsmanager:MyRDSecret:SecretString:username}}"
```

## Configuration sharing and composition

MDAA supports:

- Sharing configurations across multiple domains and environments
- Composing configurations from multiple files
- Overriding configurations at different levels (global, domain, environment, module)

## Configuration Merging Rules

- Lists on same config key will be merged across config files
- Objects on same config key will be concatenated

- Scalar values will be overridden, with later configs taking precedence

## Customizing the solution

MDAA can be customized using:

- Custom naming modules
- Custom CDK aspects
- Configuration file extensions
- Custom resource naming implementations



# Developer guide

## Setting up MDAA Dev Environment

1. Clone this repo.
2. Install NPM/Node
3. NPM Install CDK, Lerna
4. Authenticate to the MDAA NPM Repo
5. From the root of the repo, run npm install:

```
npm install
```

6. After making code changes, run a build/test using lerna:

```
lerna run build && lerna run test
```

Alternatively, you can run `npm run build && npm run test` in each individual package you have modified.

## Testing

### Testing Overview

The testing approach for MDAA changes varies depending on the type of package being tested (App, Stack, or Construct). Before testing, ensure that the entire MDAA repo is cloned, bootstrapped, and built.

### Testing Constructs and Stacks

Constructs and Stacks should be tested via unit testing using the CDK Assertions framework. This framework can be used to ensure that the CFN resources produced by a MDAA construct or stack are defined as expected in the resulting CFN template. Specific attention should be paid in these unit tests to any resource property which has compliance implications.

## Example Construct/Stack Unit Tests

```
import { MdaaTestApp } from "@aws-mdaa/testing";
import { Stack } from "aws-cdk-lib";
import { MdaaKmsKey } from '@aws-mdaa/kms-constructs';
import { Match, Template } from "aws-cdk-lib/assertions";
import { NagSuppressions } from "cdk-nag";
import { MdaaBucket, MdaaBucketProps } from "../lib";

describe( 'MDAA Construct Compliance Tests', () => {
  const constructTestApp = new MdaaTestApp()
  const constructTestStack = new Stack( constructTestApp, "test-stack" )

  const testKey = MdaaKmsKey.fromMdaaKeyArn( constructTestStack, "test-key",
    "arn:test-partition:kms:test-region:test-account:key/test-key" )

  const testContstructProps: MdaaBucketProps = {
    naming: constructTestApp.naming,
    bucketName: "test-bucket",
    encryptionKey: testKey
  }

  const testConstruct = new MdaaBucket( constructTestStack, "test-construct",
    testContstructProps )
  NagSuppressions.addResourceSuppressions(
    testConstruct,
    [
      { id: 'NIST.800.53.R5-S3BucketReplicationEnabled', reason: 'MDAA Data Lake
does not use bucket replication.' },
      { id: 'HIPAA.Security-S3BucketReplicationEnabled', reason: 'MDAA Data Lake
does not use bucket replication.' }
    ],
    true
  );
  constructTestApp.checkCdkNagCompliance( constructTestStack )
  const template = Template.fromStack( constructTestStack );

  test( 'BucketName', () => {
    template.hasResourceProperties( "AWS::S3::Bucket", {
      "BucketName": constructTestApp.naming.resourceName( "test-bucket" )
    } )
  } )
}
```

```
test( 'DefaultEncryption', () => {
  template.hasResourceProperties( "AWS::S3::Bucket", {
    "BucketEncryption": {
      "ServerSideEncryptionConfiguration": [
        {
          "BucketKeyEnabled": true,
          "ServerSideEncryptionByDefault": {
            "SSEAlgorithm": "aws:kms",
            "KMSEMasterKeyID": testKey.keyArn
          }
        }
      ]
    }
  } )
} )
}
```

## Testing Apps

MDAA Apps can be developed and tested like any other CDK app. This typically involves a `cdk list/synth/diff/deploy` from within the App source directory, while also providing the necessary context values which would otherwise be provided by the MDAA framework. Executing the `cdk` command will result in the application source code being built. However, any changes made in underlying dependencies (such as stacks and constructs) would require either a `lerna run build` at the root of the MDAA repo, or `npm run build` in the package folder for each of the modified dependencies.

### Example CDK Command Invoking a MDAA App

```
cdk synth --require-approval never -c org="" -c env="" -c domain="" -c
module_configs="" -c tag_configs="" -c module_name="" --all
```

# Reference

This section includes information about an optional feature for collecting unique metrics for this solution and a [list of builders](#) who contributed to this solution.

## Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When activated, the following information is collected and sent to AWS:

- Solution ID - The AWS solution identifier
- Unique ID (UUID) - Randomly generated, unique identifier for each deployment
- Timestamp - Data collection timestamp
- Cost Feature Enabled - Information on whether the user is using the cost feature
- Number of Accounts - Number of accounts user has onboarded in their deployment
- Number of Diagrams - Number of diagrams created in each deployment
- Number of Resources - Number of resources discovered in all onboarded accounts

AWS owns the data gathered through this survey. Data collection is subject to the [Privacy Notice](#). To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the [AWS CloudFormation template](#) to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

```
Mappings:
  Solution:
    Metrics:
      CollectAnonymizedUsageMetrics: 'true'
```

to:

```
Mappings:
```

```
Solution:
Metrics:
  CollectAnonymizedUsageMetrics: 'false'
```

1. Sign in to the [AWS CloudFormation console](#).
2. Select **Create stack**.
3. On the **Create stack** page, **Specify template section**, select **Upload a template file**.
4. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.
5. Choose **Next** and follow the steps in [Launch the stack](#).

## Contributors

- Andrew Price
- Amr Ahmed
- Sudeshna Dash
- Guoneng Zhong
- John Reynolds

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The solution is licensed under the terms of the [Apache License, Version 2.0](#).