AWS Guidance: Integrating Learning Management Systems (LMS) with AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: AWS Guidance: Integrating Learning Management Systems (LMS) with AWS

Table of Contents

AWS Guidance: Integrating Learning Management Systems (LMS) with AWS

Arnaud Lauer, Pete Davis, and Elie Elmalem, Amazon Web Services

June 2026 (document history)

Learning Management Systems (LMS) are central platforms for modern education, connecting students, faculty, and curricula in a shared online environment. As educators and institutions strive to deliver more personalized and engaging learning experiences, integrating these platforms with AWS services enables better student outcomes, deeper learning engagement, and more effective teaching practices. From AI-driven personalization and automated administrative tasks to data-rich insights, integrating LMS platforms with AWS services opens a range of capabilities for more adaptive, inclusive, and future-ready learning ecosystems.

This guide outlines integration patterns, architectures, and implementation guidance for connecting LMS platforms (such as Moodle, Canvas, Blackboard, and others) with AWS services. It also provides a deep dive into a specific use case that leverages generative AI services like Amazon Bedrock, demonstrating how these technologies can transform learning experiences through practical implementation patterns and code samples using Moodle as the base LMS.

Intended audience

This guidance document is primarily designed for technical architects, managers, and leaders who are responsible for planning and implementing LMS integration strategies with AWS services. The content is particularly relevant for technical teams within educational institutions and AWS Partners looking to integrate their solutions with LMS. Although the guide focuses on technical aspects, it is also valuable for IT decision-makers who need to understand integration patterns, architectural options, and implementation considerations.

Objectives

After reading this guide, you will be able to:

- Evaluate and select the appropriate integration pattern (plugin, LTI, standalone, event-driven, or Extract, Transform, Load (ETL) for your use case

- Design an architecture that connects your LMS to AWS services securely
- Implement a generative AI integration usi Amazon Bedrock with Moodle
- Apply responsible AI practices in educational contexts
- Deploy working integration examples using the provided code samples

Prerequisites

- An active AWS account
- Administrative access to an LMS platform (Moodle, Canvas, or Blackboard)
- Programming experience with PHP or Python
- Familiarity with REST APIs and web application architecture

Service limitations

Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Key benefits of LMS-AWS integration

When effectively implemented, LMS-AWS integration can deliver several valuable capabilities:

- **Enhanced Learning Experiences**: Create personalized learning paths, generate AI-enhanced content, and implement intelligent tutoring systems that adapt to individual student needs and learning styles.
- **Administrative Efficiency**: Automate tasks like grading, content generation, and feedback mechanisms, allowing educators to focus on higher-value activities like interactions with students.
- **Analytics and Insights**: Harness advanced data analytics to deeply understand each student's learning journey, enabling tailored interventions, personalized support, and adaptive learning experiences that significantly enhance individual educational outcomes and overall student success.
- **Security**: Implement enterprise-grade security and compliance capabilities that meet educational data protection requirements while enabling innovation.

- **Cost Efficiency**: Adopt a pay-as-you-go model for specialized data and AI processing capabilities, avoiding large upfront investments in educational technology infrastructure.

Successful implementation requires careful consideration of several factors: technical expertise in both AWS services and LMS platforms; effective change management to help educators and students adapt to new capabilities; comprehensive data governance policies to ensure responsible data handling; and appropriate cost monitoring and quota management mechanisms to manage the pay-as-you-go model effectively.

Integration options overview

Multiple integration patterns exist for connecting AWS services with Learning Management Systems (LMS), each offering different advantages depending on your specific use case. Organizations should evaluate which approach (or combination of approaches) best meets their requirements by considering factors such as:

- Technical complexity and available expertise
- User experience requirements
- Data flow needs
- Existing LMS capabilities and constraints
- Implementation timeframes and resources

Integration patterns generally fall into two broad categories:

1. **Front-end integration**: Focuses on an interactive experience for the LMS user
2. **Back-end integration**: Connects data and services between systems without user interaction

Depending on your learning objectives, technical landscape, and organizational needs, you may benefit from implementing one specific pattern or combining multiple approaches to create a comprehensive solution.

Front-end integration approaches

Integration Approach	Description	Pros	Cons	Best For
Native LMS Plugin/Extension	Custom code that extends LMS functionality directly within the platform'	• Deep LMS integration • Native user experience	• LMS-specific development • Maintenance challenges with LMS version updates	• Single LMS environments • Organizations with expertise in the specific LMS

	s extension framework	• Direct access to LMS data models	• Limited by LMS plugin architecture constraints	technology stack
Learning Tools Interoperability (LTI)	Standards-based integration that embeds external tools while maintaining interoperability across LMS platforms	• Cross-platform compatibility • Independent release cycles • Standardized authentication and data exchange	• Operates in iframe / separate context limiting some UX options • Requires implementing LTI standards	• Multi-LMS environments, vendors building tools for multiple institutions • Standardized deployments

Back-end integration approaches

The back-end integration approaches can be used in isolation or are often combined to achieve the best possible outcome.

Integration Approach	Description	AWSServices	Best For
API Integration	Custom code running on AWS communicating with LMS services through APIs	• AWS Lambda • Amazon EC2 • Containers	• Real-time data integration for external applications
ETL/ELT Data Pipeline	Batch data extraction and transformation for analytics	• AWS Glue • Amazon S3 • Amazon AppFlow	• Analytics use cases • Reporting
Event-Driven Architecture	Event producers and consumers that react to changes in LMS or AWS systems	• Amazon EventBridge	• Near real-time requirements • Complex workflows

- Amazon Simple Notification Service
- Amazon Simple Queue Service

Use case mapping by integration option

Use Cases Requiring UI Integration

These are example scenarios that integrate directly with AWS services through the LMS interface:

- **AI Writing Assistant**: LTI or plugin provides real-time writing feedback and suggestions using Amazon Bedrock foundation models.
- **Interactive AI Tutor**: LTI or plugin with conversational interface using Amazon Bedrock agents to answer student questions about course content.
- **Document Summarization Tool**: LTI tool that processes course documents with Amazon Bedrock to generate concise summaries and key points.
- **Language Translation**: Plugin that translates course content into multiple languages using Amazon Bedrock.
- **Quiz Generation**: LTI or plugin that creates assessments from course materials using Amazon Bedrock Knowledge Bases with course specific content.

Use Cases Not Requiring UI Integration

These are example scenarios that typically leverage back-end integration patterns where users interact primarily with the LMS, while AWS services work behind the scene:

- **Student Performance Analytics**: ETL pipeline extracts LMS data to AWS analytics services for comprehensive performance dashboards beyond native LMS capabilities.
- **Automated Content Recommendations**: Event-driven architecture responds to student activities in the LMS to trigger personalised content recommendations using AI services.
- **Predictive Learner Support**: machine learning (ML) models process LMS data via ETL to identify at-risk students before performance issues become critical.

- **Automated Assignment Grading**: Integration processes submissions through Amazon Bedrock for objective assessment without UI changes.
- **Knowledge Base Population:** Upload new course content to an Amazon Bedrock Knowledge Base to power Retrieval Augmented Generation (RAG) workloads such as chat applications and question generation.

Detailed integration architecture options

This section provides architectural guidelines, implementation details, and best practices for each key integration option. Each pattern includes components, workflow, security considerations, and practical implementation guidance to help you design effective AWS-LMS integrations.

LMS plugin integration

LMS Plugin Integration directly extends the LMS platform's functionality through its native extension framework, providing a seamless experience for users while leveraging AWS services for enhanced capabilities.

Architecture and Components

The LMS plugin integration architecture connects the Learning Management System with AWS services through a secure, scalable interface:

1. **LMS Environment**: Hosts the LMS platform and the custom plugin code within the LMS application server. This is where users interact with the plugin interface. It contains four interconnected components:
   - **LMS UI with Plugin UI**: The interface users interact with directly
   - **Custom LMS Plugin**: The extension code that adds AWS-powered capabilities
   - **LMS Core Functionality**: Native LMS features that the plugin interacts with
   - **LMS Content**: Educational materials and data accessed by the plugin
2. Amazon API Gateway**/** AWS AppSync: Serves as the secure bridge between the LMS and AWS services by:
   - Receiving requests from the LMS plugin
   - Routing authenticated requests to appropriate backend services
   - Returning responses to the LMS environment
3. **Authentication Layer**: Secures API access through either:
   - AWS Lambda Authorizer: Provides custom authorization logic for validating requests based on LMS context

- AWS Identity and Access Management**:** A role associated with an EC2 instance or IAM access keys can be used to sign requests using SigV4. IAM access keys are not recommended as they are long-term credentials.

4. **AWS services**: Backend services that provide the actual functionality, such as Amazon Bedrock for generative AI capabilities, Amazon S3 for content storage, and other AWS services as needed for specific use cases

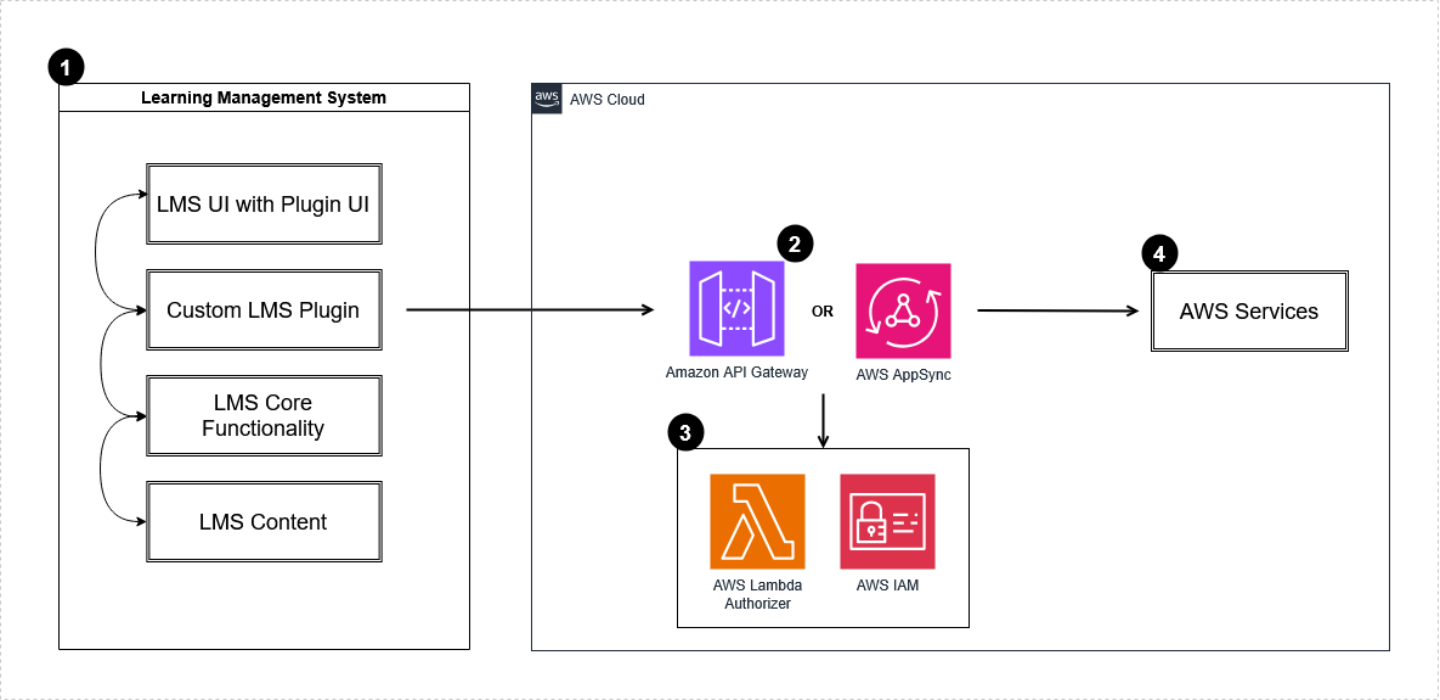

Figure 1: LMS Plugin Integration Pattern

User Flow

When a user interacts with the plugin in the LMS:

1. The plugin captures the user's input and relevant LMS context
2. The plugin sends an authenticated request to API Gateway
3. API Gateway authorizes the request using either a Lambda Authorizer or IAM
4. Upon successful authentication, the request is forwarded to appropriate AWS services
5. AWS services process the request and return results
6. Results flow back through API Gateway to the plugin

7. The plugin renders the results within the LMS interface

This architecture ensures secure communication between the LMS and AWS while maintaining a seamless user experience within the familiar LMS environment.

Implementation Technologies

The plugin implementation depends on the LMS platform's technology stack:

- **Moodle**: PHP with specific plugin structure requirements
- **Blackboard**: Java/Spring and JavaScript adhering to Blackboard's Building Block API
- **Canvas**: Does not support native plugins but recommends the use of LTI or Canvas APIs for integration

When developing plugins for any LMS platform, developers must:

1. Understand the specific LMS's plugin architecture and guidelines
2. Follow security best practices for the platform
3. Implement appropriate authentication and authorization mechanisms
4. Adhere to the platform's UI/UX guidelines
5. Consider performance implications and scalability
6. Maintain compatibility with LMS version updates

Authentication and Security

When integrating AWSservices from an LMS plugin, the recommended approach is to use a proxy API integration: The plugin calls a custom API endpoint (for example, API Gateway) that acts as a proxy to the AWS services. Authentication and authorization are handled at the API layer. AWS service logic is isolated from the LMS environment. For this proxy API approach:

- **API Authorization**: IAM roles, LMS token, IAM Access Key (if other options are not viable)
- **Plugin Implementation**: The plugin makes HTTPS requests to your secured API endpoints
- **Credential Protection**: If using long term credentials such as IAM Access Key these will need to appropriately stored and protected

This architecture creates a clear security boundary between the LMS and AWS environments. The LMS plugin only needs to know the API Gateway endpoint URL and appropriate authentication method for the API (such as IAM Role or OAuth token). Regardless of the specific integration approach, LMS plugin development requires careful consideration of security, performance, and maintainability. Proper authentication, authorization, and data protection mechanisms must be implemented to ensure the integrity of the overall system.

Advantages

- **Tight UI/UX integration**: The plugin becomes part of the native LMS interface, providing a seamless experience for users who don't want to leave the LMS environment.
- **Direct Access to the LMS Context**: Plugins have access to course context, user roles, and other LMS-specific data that external applications would need to request.
- **Leverage Existing Authentication**: Can utilize the LMS's authentication and authorization mechanisms, simplifying security implementation.

Limitations

- Requires plugin development expertise for each LMS
- Must maintain compatibility with LMS version updates
- Subject to LMS plugin architecture constraints
- Requires separate implementations for each LMS platform

Learning Tools Interoperability (LTI)

Learning Tools Interoperability (LTI) is an education technology standard developed and maintained by 1EdTech (formerly IMS Global Learning Consortium). LTI enables third-party tools to integrate seamlessly with LMS platforms by providing a standardized way for external applications to authenticate with an LMS and exchange data. This standard ensures secure, efficient, and consistent interoperability between learning platforms and external educational tools. It allows any LTI compliant tool to plug into any compliant LMS.

Common LTI implementations include digital textbook launches, grade assessment tools, and interactive learning modules. These integrations work well for straightforward use cases where the need for data exchange is minimal and the interaction model is simple. Grade passback scenarios,

such as quiz scores from external platforms or assignment completion status, also function adequately within LTI's constraints.

However, the integration of Generative AI tools presents even more demanding requirements. These systems need training data from student interactions, course content and materials, assessment responses and feedback, learning patterns and preferences, and cross-course correlations. They require real-time data processing, continuous learning and adaptation, deep content understanding, and the ability to handle complex interaction patterns – all of which exceed LTI's capabilities.

For modern educational technology needs, LTI often serves best as one component of a larger integration strategy. Many successful implementations use LTI for its strengths (secure launches, basic data passing, grade return) while supplementing it with additional integration methods for more complex requirements.

Architecture and Components

The LTI integration architecture consists of:

1. **LMS Environment**: Any LTI-compliant LMS (for example Moodle, Canvas, Blackboard)
    - **LMS UI with iFrame Container**: The standard LMS interface where the LTI tool will appear embedded
    - **LTI Launch Interface**: Handles the LTI launch process and OAuth-based security protocol
    - **LMS Core Functionality**: Native features of the LMS that the LTI tool might interact with
    - **LMS Content**: Course materials and data that might be accessed by the LTI tool
2. **LTI Tool Application**: A web application hosted on AWS that implements the LTI standard
    - **LTI Tool Frontend**: Web application hosted on AWS that renders inside the iFrame container
    - **LTI Tool Backend**: Server-side component that processes requests and business logic
3. **AWS Services Layer**: Backend AWS services providing enhanced functionality such as Amazon Bedrock

The LTI standard enables secure communication between the LMS and the AWS-hosted tool while maintaining separation between the systems. This provides flexibility for the tool's development while ensuring secure data exchange with the LMS environment.

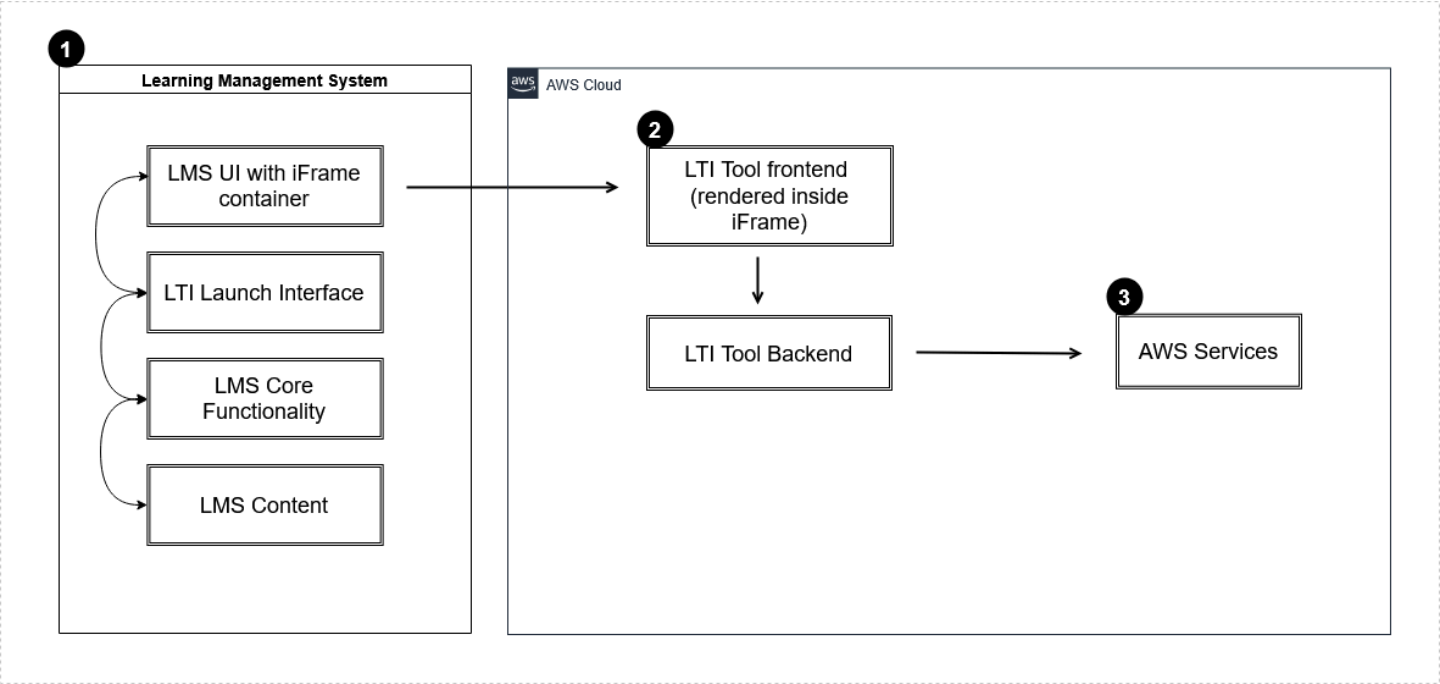

Figure 2: LTI Integration Pattern

User Flow

When a user interacts with the LTI plugin in the LMS:

1. The LMS and the LTI tool perform a handshake process that enables the exchange of information.
2. Upon success, the LTI tool is displayed inside an iFrame on the LMS.
3. The tool makes authenticated calls to an AWS backend.
4. The tool returns the requested information.
5. Alternatively, the tool can also pass information back to the LMS.

Implementation Technologies

- **Frontend**: Any modern web framework can be used (React, Vue, Angular)
- **Backend**: Node.js, Python, or other web technologies with LTI library support
- **LTI Version**: Preferably LTI 1.3 with LTI Advantage features for enhanced capabilities

- **Hosting**: Commonly deployed as a containerized application on Amazon Elastic Container Service (Amazon ECS) / Amazon Elastic Kubernetes Service (Amazon EKS) or as a serverless application using API Gateway and Lambda

Authentication and Security

LTI implements a secure authentication mechanism based on OAuth 2.0:

1. User clicks an LTI tool link within the LMS
2. LMS generates a signed JSON Web Token (JWT) containing user context and launch parameters
3. User is redirected to the LTI tool with the JWT
4. LTI tool validates the JWT signature using the LMS's public key
5. Tool creates a session and displays the appropriate content
6. AWS services are accessed by the LTI application's backend using IAM roles

This approach keeps AWS credentials entirely separate from the LMS environment, improving security posture.

Advantages

- Cross-platform compatibility across LMS environments (Moodle, Canvas, Blackboard, etc.)
- Independent development and release cycles
- Standardised authentication and data exchange
- Portable across institutions and departments

Limitations

- Functions within iframe/separate context, limiting some UX options
- Requires implementing and maintaining LTI standards compliance
- More complex authentication flow than direct plugin integration
- Might have limited access to certain LMS features

Standalone application with API integration

This pattern involves developing a standalone application that interfaces with the LMS through its APIs. While operating independently, the application can both read from and write data to the LMS as needed. This integration approach is particularly valuable when LMS data needs to be accessed by external systems. For instance, when creating analytics solutions for student engagement or developing dashboards for non-LMS users (such as registrar office staff or deans), a separate application might be more appropriate than trying to extend the LMS itself. This allows for specialized tools tailored to these specific user groups' needs.

Architecture and Components

1. **Standalone Web/Mobile Application**:
   - **LMS API**: The interface exposed by the LMS that allows external applications to read and write data
   - **LMS Core Functionality**: Native features and functions of the LMS
   - **LMS Content**: Course Materials, user, data, and other educational content stored in the LMS
2. **AWS Cloud:**
   - **LMS API Client**: Component that handles pull/push API calls to communicate with the LMS API
   - **Application Frontend**: User interface layer that users interact with directly
   - **Application Backend**: Server-side components that processes business logic
   - **AWS Services**: Backend services that provide specialized functionality such as Amazon S3 or Amazon Bedrock.

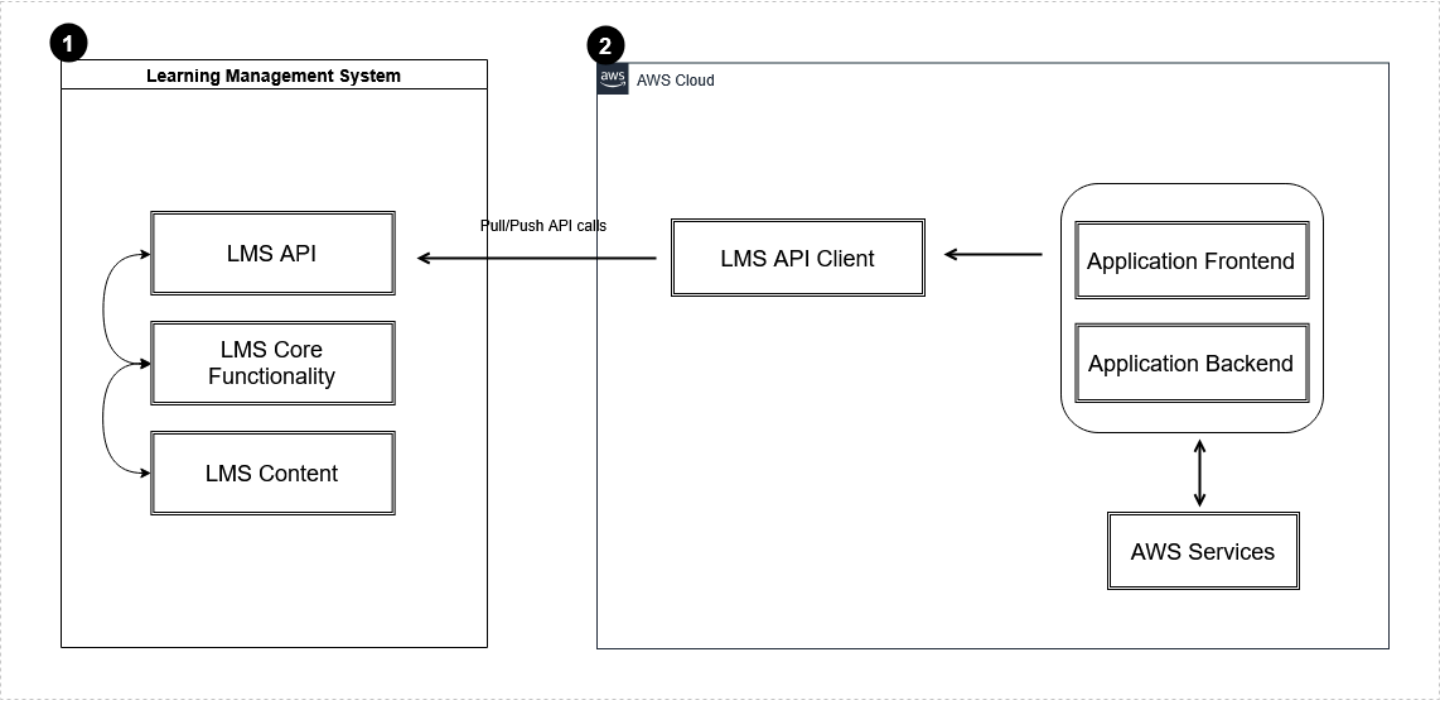

Figure 3: Standalone API Integration Pattern

User Flow

1. User authenticates with the standalone application
2. Application authenticates with LMS using OAuth 2.0 or API keys
3. Application retrieves or updates necessary data from LMS API
4. User interacts with application features that leverage the retrieved data and AWS services
5. AWS-processed results are displayed to user and optionally saved back to LMS

Implementation Technologies

- **Application Hosting**: Amplify, Elastic Beanstalk, or container services
- **Authentication**: OAuth2, LMS Access Key
- **LMS Communication**: Software Development Kits (SDKs) or REST client libraries for the specific LMS

Authentication and Security

When authenticated against the LMS API most platform use either OAuth 2.0 or other tokens to authenticate the request from the application.

Advantages

- Complete flexibility in application design and user experience
- Independent deployment and scaling from the LMS
- Full control over the technology stack

Limitations

- Limited by available LMS API capabilities
- Might require ongoing updates as LMS APIs evolve
- Additional complexity in maintaining data consistency
- Many requests to the LMS API might have a negative impact on LMS system performance so rate limiting should be implemented
- Tight coupling between systems, patterns such as messaging, API Gateways or facades can limit this

Event-driven integration

Event-driven integration leverages the LMS' event system to trigger actions in AWS services based on user activities or system changes within the LMS.

Architecture and Components

1. **LMS Environment**: Hosts the LMS platform and the custom plugin code within the LMS application servers. This is where users interact with the LMS to generate events. It contains three interconnected components:
   - **Custom LMS Plugin:** Extension code that is invoked for each event raised and passes the event to Amazon EventBridge or publishes to the stream
   - **LMS Core Functionality**: Native LMS features that the user interacts with creating LMS Events

- **LMS Content**: Educational materials and data created and accessed by the users

2. **EventBridge / Streaming**: Receives the events from the LMS Plugin and either triggers rules to invoke downstream services when the event matches a defined pattern, or processes records in the stream to invoke appropriate AWS services. Streaming services can be Amazon Kinesis Data Streamsor Amazon Managed Streaming for Apache Kafka.
3. **AWS Services**: Backend services that provide the actual functionality, such as Lambda for executing custom code.

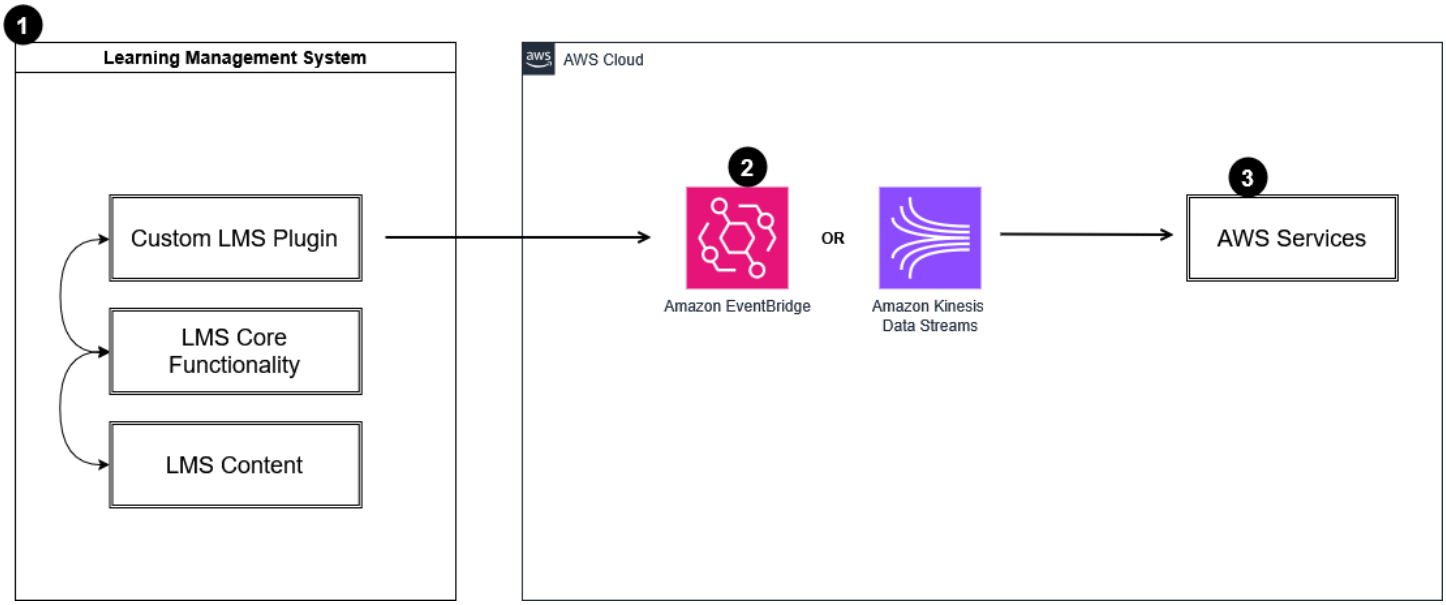

Figure 4: Event-Driven Integration Pattern

System Flow

When an event occurs in the LMS, the following process takes place:

1. LMS captures the event in its Events subsystem
2. Custom LMS Plugin observes each created event
3. The plugin sends an authenticated request to EventBridge or streaming service with the event payload
4. AWS API validates the request using IAM
5. Upon successful authentication, the request is added to an EventBridge event bus or stream
6. A rule receives incoming events and routes them to appropriate AWS services based on matching event patterns or records are processed off the stream and processed by the consumer.

This architecture ensures secure communication between the LMS and AWS services allowing LMS platform capabilities to be extended with AWS services.

Implementation Technologies

The LMS plugin implementation depends on the platform's technology stack, for example:

- **Moodle**: The Moodle Events API can be used with a local plugin to send the events to EventBridge.
- **Canvas**: Canvas Live Events can be published to Amazon SQS so AWS services can pull the messages or can be invoked via a Lambda function. Using a Lambda function to write the events on the Amazon SQS queue to EventBridgemight still be useful to decouple the subscribers.
- **Blackboard Learn**: Learn Activity Streams are an implementation of the Caliper Specification which enables the collection of learning data in a standardized way. The events are sent to a Caliper Event Store which can be used to invoke AWS services.

The plugin implementation must adhere to the LMS's plugin development guidelines and APIs. This often involves:

- Implementing specific plugin lifecycle methods and configuration pages
- Accessing LMS data through provided APIs, data access patterns and security practices.

Authentication and Security

When integrating with AWS, three primary authentication methods exist:

1. IAM **Role (Preferred Method):**
   - Attach a role to EC2 instances or container if the LMS is running on AWS
   - Provides temporary, secure credentials for AWS API calls
   - Eliminates need to store long-term access keys in configuration
2. IAM Roles Anywhere **(Recommended for Non-AWS Hosted Moodle):**
   - Uses X.509 certificates to obtain temporary AWS credentials for workloads running outside AWS
   - Provides temporary credentials without long-lived access keys
   - Requires attaching an appropriate IAM policy to your IAM Roles Anywhere role

- See IAM Roles Anywherefor setup instructions

3. **AWS Access Key (Alternative Method):**
   - Use when an IAM Role cannot be assumed
   - Requires careful security management
   - Long-term credentials that must be manually rotated

AWS SDK support these methods.

Advantages

- Access to broad range of AWS capabilities
- Enhanced scalability
- System decoupling
- Near real-time updates
- Improved reliability as events / stream records can be persisted and replayed
- Independent development and release cycles
- Flexibility in development language

Limitations

- Unable to integrate with LMS user interface
- Callbacks to the LMS from downstream services might cause unintended system load

ETL integration

ETL (Extract, Transform, Load) integration focuses on batch processing of LMS data for analytics, reporting, and other data-intensive use cases.

Architecture and Components

1. **Learning Management System**:
   - **LMS API**: The interface exposed by the LMS that provides access to educational data
   - **LMS Core Functionality**: Native features and functions of the LMS

- **LMS Content**: Course materials, user data, and other educational content stored in the LMS

2. **AWS Cloud**:
   - **Scheduled Extraction**: Time-based trigger that initiates the ETL process
   - **AWS Glue ETL Jobs**: Service that extracts data from the LMS API, transforms it, and loads it into storage
   - **Amazon S3 (data lake)**: Storage repository for processed educational data
   - **AWS Analytics Services**: Tools for analyzing and visualizing the processed data
   - **AWS Lambda**: Function that can process data or write results back to the LMS (indicated by dotted lines)

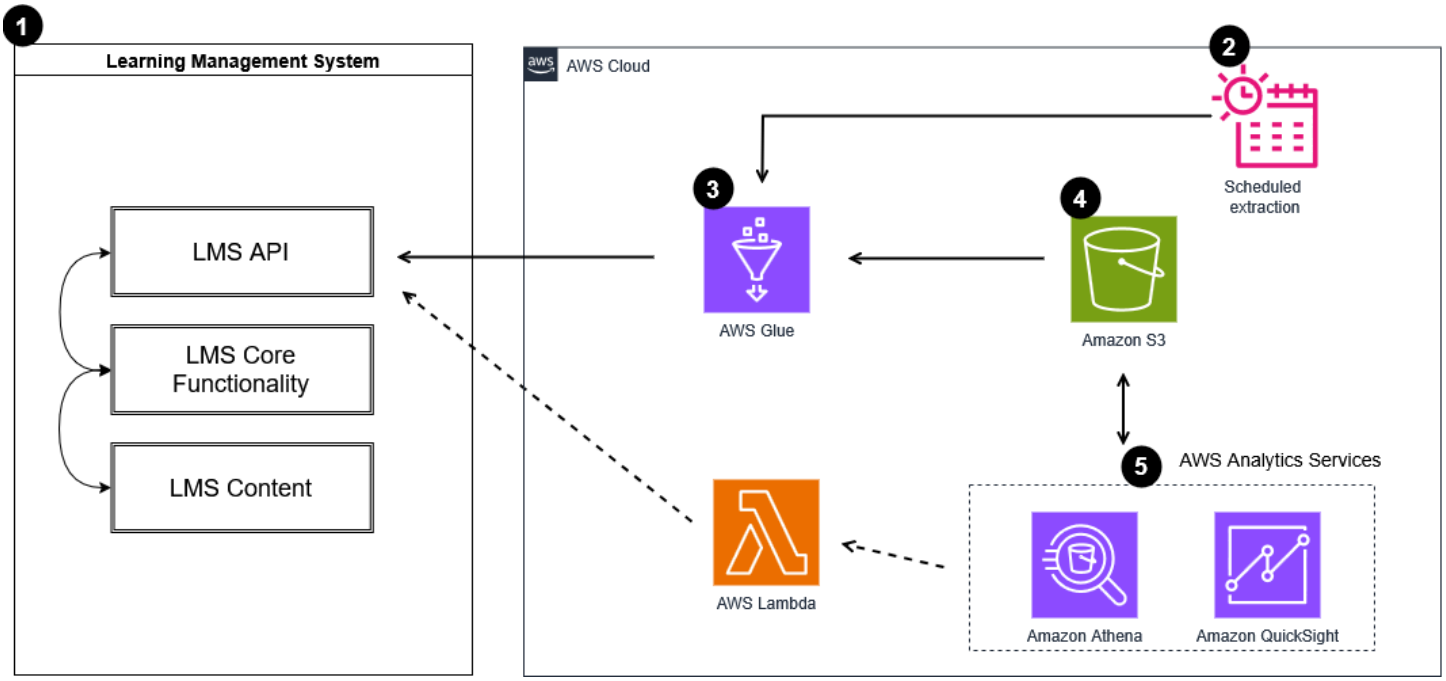

Figure 5: ETL Pipeline Integration Pattern

System Flow Example

1. **Scheduled Extraction** triggers **AWS Glue ETL Jobs** at predetermined intervals
2. **AWS Glue ETL Jobs** connect to the **LMS API** to extract data
3. Extracted data is transformed and loaded into **Amazon S3 (data lake)**
4. **AWS Analytics Services** access the data lake to perform analysis

5. Optionally, insights and processed data can be fed back to the LMS via **AWS Lambda** functions (shown by dotted lines)

Implementation Technologies

- **ETL Processing**: AWS Glue for serverless data integration
- **Data Storage**: Amazon S3 for the data lake architecture
- **Analytics**: Amazon Athena for SQL queries, Amazon Quick Sight for visualization
- **Scheduling**: Amazon EventBridge or AWS Glue triggers for triggering extraction on schedule
- **Optional Processing**: AWS Lambda for additional transformations or writing data back to LMS

Authentication and Security

When authenticated against the LMS API most platform use either OAuth 2.0 or other tokens to authenticate the request from the application.

Advantages

- Allows LMS data to be processed and visualized using modern analytics and AI tools
- Design for scalable extraction of potentially large datasets

Limitations

- Full extractions might lead to excessive system load, investigate change data capture mechanisms or consider a separate LMS instance for extraction
- Extraction during core hours might impact user performance
- Additional data governance required to address privacy and compliance requirements outside of the LMS

The following AWS Workshops are useful for exploring ETL based integration in more detail:

- Higher Education Data Lake Immersion Day
- Amazon SageMaker Unified Studio Workshop - Improve Student Engagement

Concrete generative AI example with Amazon Bedrock

This section demonstrates practical implementation approaches for integrating Amazon Bedrock's generative AI capabilities with Learning Management Systems. The most appropriate pattern should be chosen based on the particular use case.

Amazon Bedrock is particularly valuable for LMS integration as it provides access to multiple foundation models, built-in RAG capabilities, and agent functionality without requiring machine learning expertise. Additionally, Amazon Bedrock offers comprehensive guardrails functionality that enables educational institutions to implement appropriate content filtering and moderation policies. These guardrails help ensure AI-generated content meets educational standards and institutional policies by managing toxicity, harmful content, and inappropriate responses—a critical consideration in educational environments.

These patterns demonstrate how to leverage these capabilities to enhance teaching and learning experiences while maintaining appropriate content safeguards.

Pattern comparison

Pattern	Complexity	Use Cases	Implementation Time	Key Benefits
Direct Model Access	Low	• Text transformation • Summarization • Translation	Days	• Quick implementation • Flexible integration points
Moodle AI Subsystem	Medium	• Native Moodle integration • Standardized AI features	Days-Weeks	• Native UI • Admin controls • Platform integration

Knowledge Base Integration	Medium-High	• Context-aware Q&A • Course-specific assistance	Weeks	• Improved accuracy with course context • Personalized responses
Agents Integration	High	• Interactive tutoring • Complex workflows • Personalized support	Weeks-Months	• Conversational experience • Multi-step task completion

Responsible AI

When implementing generative AI capabilities within educational environments, institutions should consider:

- Content Appropriateness: Implement appropriate guardrails to ensure AI-generated content is academically appropriate and aligns with institutional values.
- Attribution and Academic Integrity: Define clear policies on how AI-generated content should be used in academic work and how it should be attributed.
- Monitoring and Oversight: Establish processes to review AI interactions periodically to ensure quality and appropriateness.
- Transparency: Clearly communicate to users when they are interacting with AI systems versus human instructors.
- Bias Mitigation: Regularly review AI outputs for potential biases in educational content or assessment.
- Usage Quotas and Cost Controls: Implement appropriate usage limits to manage costs while ensuring equitable access.

Implement the recommendations from Transforming application development and maintenance operating models on AWS with generative AI.

Implementing Guardrails: Utilize Amazon Bedrock Guardrails to establish appropriate content boundaries. Configure guardrails to:

- Filter inappropriate or harmful content in AI responses
- Block specific topics that may be outside the educational context
- Ensure responses align with institutional policies and educational objectives
- Implement different guardrail configurations based on user roles (for example,, stricter controls for younger students)

Direct model access

The Direct Model Access pattern provides the simplest integration approach, connecting your LMS to Amazon Bedrock foundation models. This pattern is ideal for straightforward text transformation tasks like summarization, translation, or explanation generation that don't require additional context beyond what's provided in the prompt.

In this section, we'll use Moodle as our example LMS platform to demonstrate the implementation, though the core concepts can be adapted to other LMS platforms with similar plugin architectures.

Use Cases

- **Content Summarization**: Condensing lengthy course materials or student submissions
- **Language Translation**: Translating course content to support multilingual learners
- **Explanation Generation**: Creating simplified explanations of complex concepts
- **Format Conversion**: Transforming content between different formats (for example, bullet points to paragraphs)

Architecture and Components

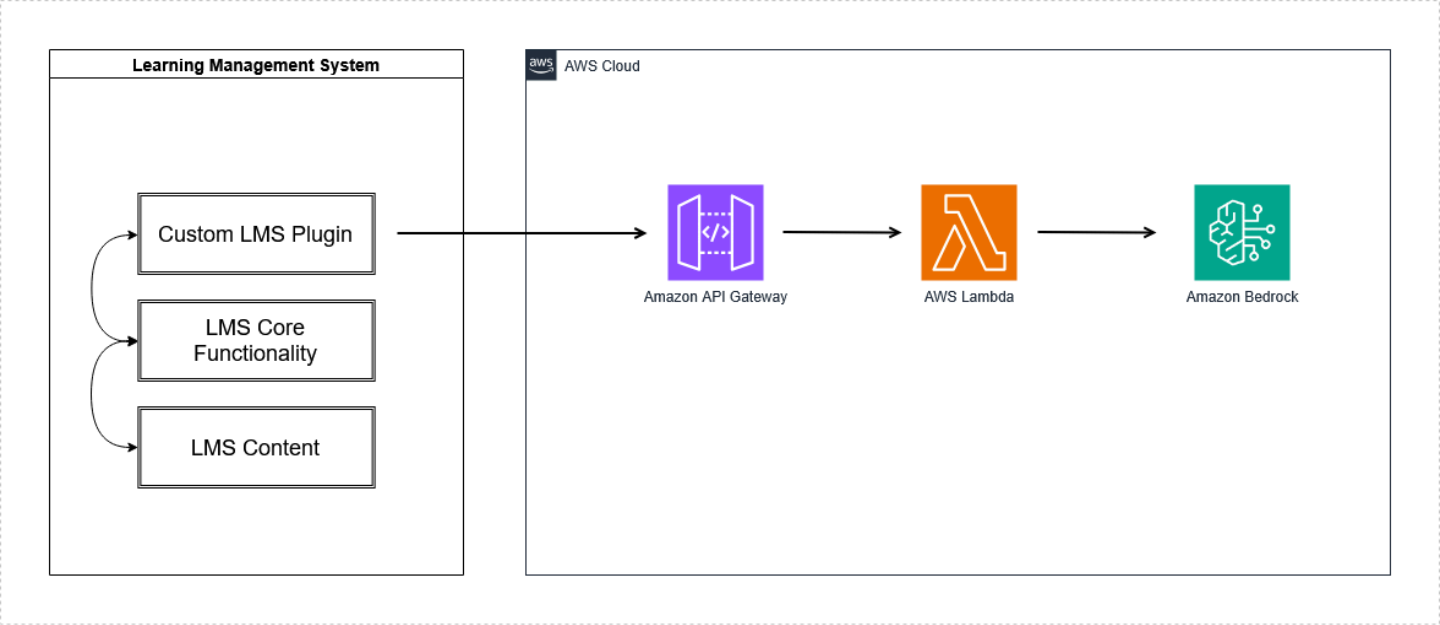

Figure 6: LMS Plugin Integration with Amazon Bedrock

The Moodle plugin calls an API Gateway endpoint backed by a Lambda function. It makes a POST request to interact with the Bedrock service.

1. **Custom LMS Plugin Component**: Native extension to the LMS that provides the UI and handles user interaction.
2. **API Gateway**: Secure endpoint that receives requests from the plugin.
3. **AWS Lambda**: Processes requests, formats prompts for the model, and handles responses.
4. **Amazon Bedrock**: Provides access to foundation models like Claude or Amazon Nova.

User Flow Example

The architecture follows this flow:

1. User selects text in the LMS and activates the plugin function (for example, "Summarize this")
2. Plugin sends the text to API Gateway with the requested operation
3. Lambda function formats an appropriate prompt for the selected operation
4. Amazon Bedrock processes the prompt and returns generated content
5. Lambda processes the response and returns it to the plugin

6. Plugin displays the results to the user within the LMS interface

Moodle Plugin Implementation Approach

Moodle plugins are extensions that allow institutions to extend Moodle's core functionality with customized features. For educational applications, these plugins can leverage Amazon Bedrock's generative AI capabilities to enhance teaching and learning experiences. To integrate Amazon Bedrock with Moodle, you can create a custom plugin based on the specific integration point you need. Moodle supports several plugin types that are well-suited for generative AI implementations:

Plugin type	Description of plugin	Generative AI examples
Block plugin	• Small displays or tools that can be moved around pages	• AI chatbot for Q&A • Daily study recommend ations
Editor plugin	• Alternative text editors	• AI writing assistant • In line content generation (for example, text, code)
Report plugin	• Create reports for admins	• Report generation • Summarization
Search engine	• Enhance search functiona lity (supports systems like Elasticsearch)	• AI enabled semantic search
Assignment feedback plugin	• Feedback to users about an assignment	• AI generated feedback

The choice of plugin type depends on your specific use case, where in the Moodle interface you want the AI functionality to appear, and how deeply you need to integrate with Moodle's core systems. For more details on plugin types, see Moodle Plugin Types.

As Moodle is written in PHP, plugins must also be written using PHP. They allow for full control of functionality and customization. Each plugin requires a specific file structure that can vary depending on the plugin type.

A simple way to get started is using a block plugin as it offers a balance of visibility, flexibility, and implementation simplicity. Block plugins can be easily added to course pages and provide a dedicated interface for AI interactions without disrupting existing course components.

Implementation Considerations

- **Token Limits**: Manage content length to stay within model token limits
- **Cost Management**: Implement usage tracking and quotas to control costs
- **Error Handling**: Gracefully handle errors and provide feedback to users
- **Prompt Engineering**: Create effective prompts that yield consistent, high-quality outputs
- **User training**: Provide clear guidance on appropriate use and limitations

Moodle AI subsystem

LMS providers are rapidly integrating AI technology into their platforms, such as the Blackboard Learn AI Design Assistant. This section focuses on how Moodle integrates with AI technology through its dedicated AI subsystem.

Moodle 4.5 introduced a dedicated AI subsystem. This new functionality provides the foundation for integrating a wide range of AI capabilities directly into the Moodle LMS. The AI subsystem is designed to be provider-agnostic, allowing Moodle sites to leverage any open source or commercial large language model (LLM) through customized provider plugins.

As of version 4.5, Moodle's AI subsystem does not support AWS and Amazon Bedrock out of the box. The underlying architecture is engineered to support future expansion, with additional provider integrations planned for upcoming Moodle versions. However, a custom plugin is available and can be downloaded to integrate Amazon Bedrock as a model provider for Moodle's AI subsystem. For installation instructions, see the aiprovider_bedrock plugin.

Use Cases

- **Text Summarization**: Summarize text content such as course content
- **Image Generation**: Create images through AI
- **Text Generation**: Generate text through AI

Architecture and Components

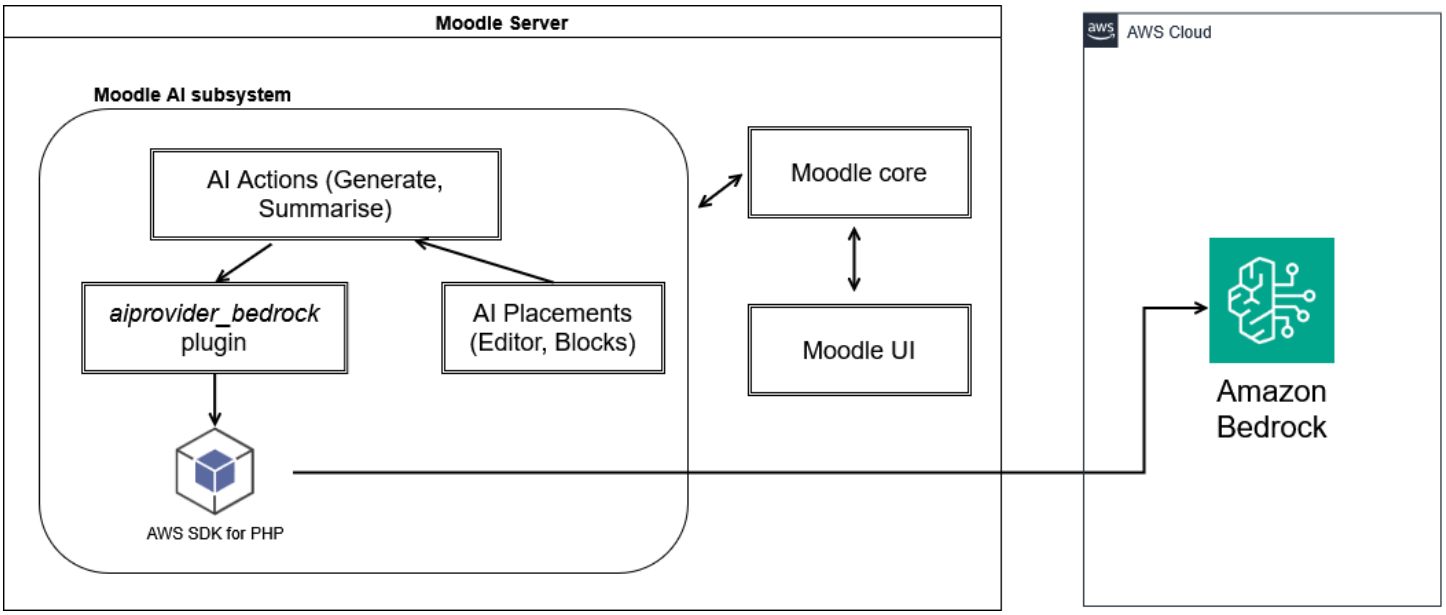

Figure 7: Moodle AI Subsystem Integration with Amazon Bedrock

- **Placements** define the specific areas within Moodle where AI-powered features can be accessed and utilized by users. For now, 2 placements are supported in Moodle core. For more information, see AI placements.
- **Actions** represent the diverse range of AI functionalities made available, such as generating text, creating images, and summarizing course content. For more information, see AI actions.
- **Providers** are the external AI services that power these actions, with each provider plugin handling the necessary integrations and configurations.

Administrators can decide which actions and placements to enable for each provider. Placements and providers are considered as plugins by Moodle and can be modified and customized. For more information, see the AI Plugins documentation.

The AI subsystem supports system and user level quotas to help manage LLM costs.

Implementation with Amazon Bedrock

The aiprovider_bedrock plugin allows Moodle administrators to:

1. Configure Amazon Bedrock as an AI provider in the Moodle AI subsystem
2. Select which models to use for different AI actions

3. Set permission controls for who can access AI features
4. Monitor usage and costs associated with AI functionality

This integration leverages Moodle's native AI framework while connecting to Amazon Bedrock's powerful foundation models, offering a standardized way to access generative AI capabilities across the Moodle platform.

Knowledge base integration

The Knowledge Base Integration pattern enhances AI capabilities by incorporating course-specific content into responses, creating a more contextually aware educational assistant. This approach uses Amazon Bedrock Knowledge Bases to implement Retrieval Augmented Generation (RAG), which enables the system to retrieve relevant information from course materials and incorporate that content into generated responses. The system can also provide citations to original course materials, helping students identify authoritative sources while maintaining academic integrity.

Unlike the direct model access approach, this pattern indexes course materials from the LMS into Amazon Bedrock Knowledge Bases, enabling more accurate and relevant responses tailored to specific courses and educational contexts.

Use Cases

- **Contextual Q&A**: Answer student questions using course-specific knowledge
- **Enhanced Research Support**: Help students explore course literature with deeper context
- **Content Discovery**: Help users find relevant materials across large course libraries
- **Customized Learning Resources**: Generate additional learning materials based on course content

Architecture and Components

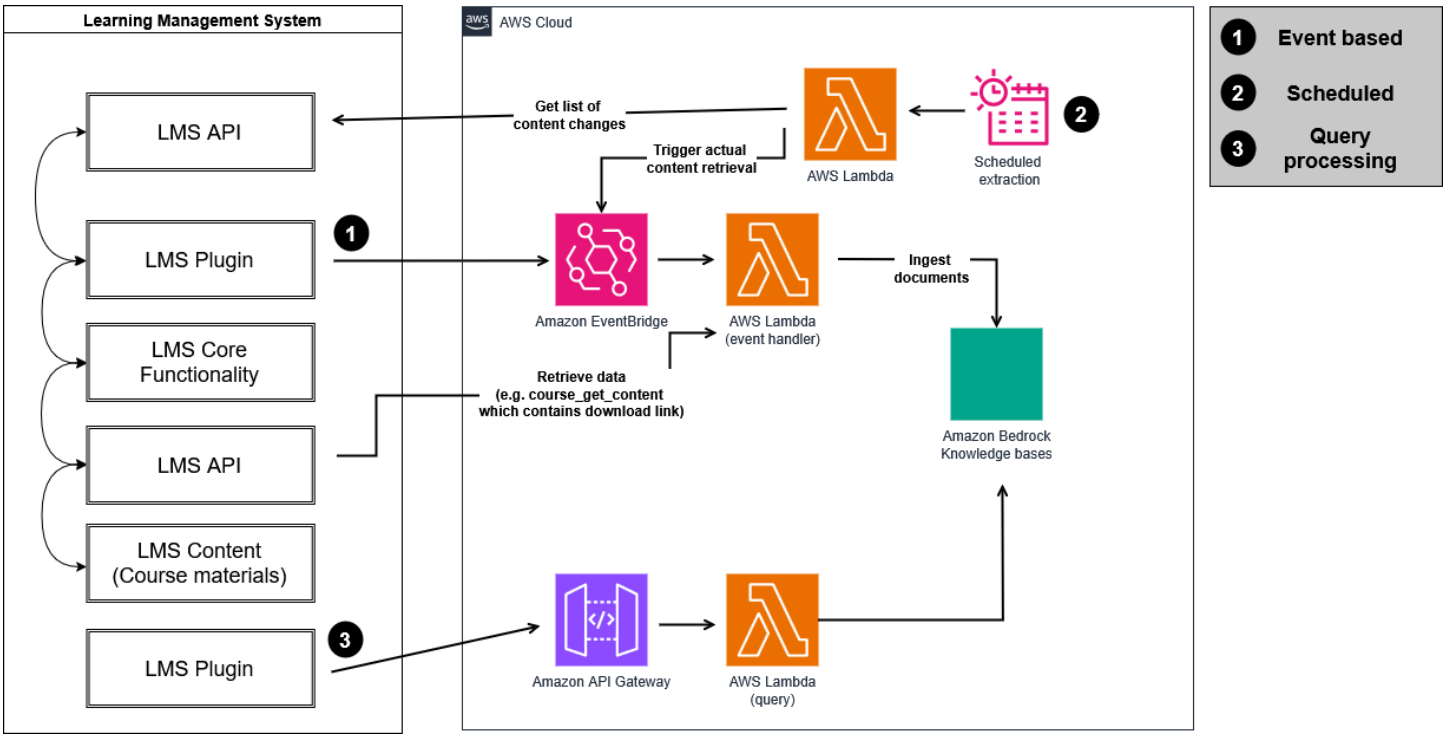

Figure 8: Moodle Integration with Amazon Bedrock Knowledge Bases

Content Synchronization Flow

This flow ensures course materials from the LMS are accurately represented in the Knowledge Base:

- **Event-Driven Path**:
  - LMS events are triggered by content updates when materials change
  - LMS events are forwarded to EventBridge in near real-time for processing
  - Ensures the knowledge repository is updated in near real-time
- **Scheduled Path**:
  - Scheduled jobs periodically scan for new or updated content
  - Useful for LMS platforms with limited event capabilities or initial population of the Knowledge Base
  - Ensures comprehensive coverage even if events are missed
- **Content Processing**:
  - Content is extracted directly from the LMS API
  - Amazon Bedrock Knowledge Bases will:

- Extract the text from various file formats
- Chunk the content for optimal retrieval
- Pass the chunk though an embedding model to generate vectors representing the content
- Allow metadata to be added for improved context retrieval
- Ingest the content and vectors into the knowledge base

Query Processing Flow

This flow occurs when users interact with the system to ask questions:

- User submits a question through the LMS interface
- Query is enriched with course context (course ID, module, user role)
- Knowledge Base retrieves relevant content chunks from course materials based on the context
- Foundation model generates a response using both the question and retrieved context
- Response with citations is presented to the user

Implementation Considerations

- **Content Synchronization**: Establish processes for keeping the knowledge base updated with LMS content including updates and deletes
- **Knowledge Organization**: Apply appropriate metadata attributes such as course ID, department, or topic to ingested content and implement metadata filtering in queries to improve relevance and scoping of retrievals
- **Citation and Attribution**: Ensure generated responses include references to source materials
- **Permission Management**: Apply appropriate access controls to respect content licensing and privacy
- **Vector Store Optimization**: Consider chunking and embedding strategies for optimal retrieval

Agents integration

The Agents integration leverages Amazon Bedrock Agents to create purposeful, interactive AI assistants that can perform complex tasks and workflows within the educational context, going beyond simple Q&A to provide more comprehensive support through structured agent capabilities.

Use Cases

- **Learning Coach**: Agent that guides students through difficult concepts with personalized explanations
- **Research Assistant**: Agent that can search, summarize, and synthesize academic resources
- **Course Designer**: Agent that helps instructors create and refine course content
- **Student Support**: Agent that leverages Knowledge Bases containing institutional FAQs, policies, and course resources to answer common questions and direct students to appropriate resources, combining retrieval-based responses with action capabilities to help students navigate support systems
- **Assessment Helper**: Agent that provides feedback on drafts and helps with revisions

The example below uses an LTI for integration but this could also be an LMS plugin.

Architecture with AWS services

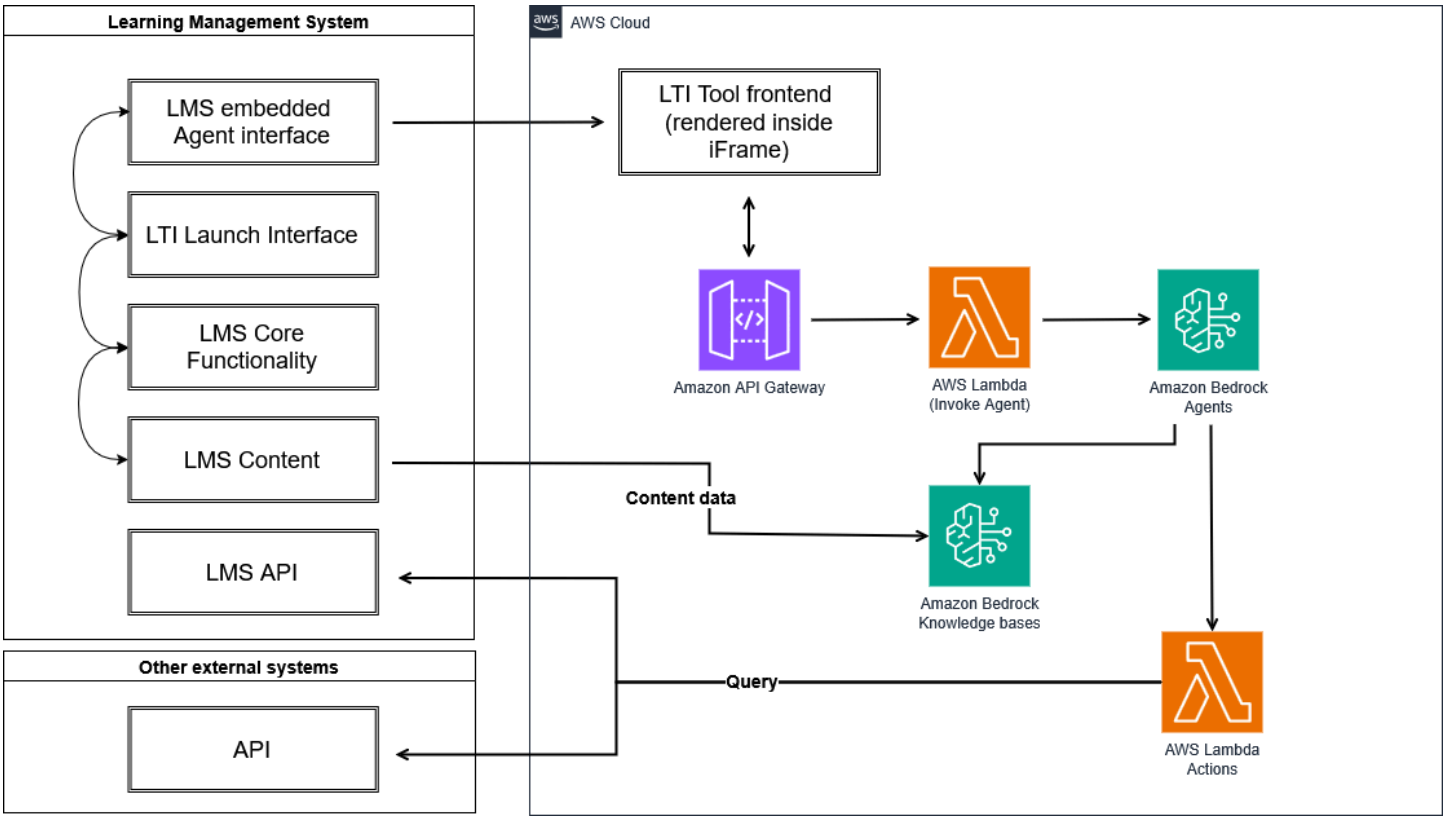

Figure 9: Moodle Integration with Amazon Bedrock Agents

Learning Management System:

- **LMS Embedded Agent Interface**: The component where users interact with the AI agent within the LMS
- **LTI Launch Interface**: Handles the LTI protocol to securely launch the tool from the LMS
- **LMS Core Functionality**: Native features and functions of the LMS
- **LMS Content**: Course materials and educational data stored in the LMS
- **LMS API**: Interface allowing external applications to interact with LMS data

Other External Systems:

- **API:** Interface allowing external applications to interact. Examples include Student Information Systems and Enterprise Resource Planning

AWS Cloud:

- **LTI Tool Frontend**: Web application rendered inside an iframe that provides the agent's interface
- **API Gateway**: Secures and manages API requests between components
- **Lambda**: Invokes the Amazon Bedrock agent
- Amazon Bedrock **Agents**: Orchestrates interactions and determines which actions to perform
- **Amazon Bedrock Knowledge Bases**: Stores and retrieves course-specific content for contextual responses
- **AWS Lambda (Actions)**: Executes specific tasks requested by the agent, including queries back to the LMS API or other APIs

User Flow

1. Student or instructor launches the AI assistant from within the LMS
2. LTI launches the assistant interface with appropriate context
3. User engages in conversation with the agent
4. Agent processes queries through Bedrock and performs actions as needed
5. Results are presented to the user in a conversational format
6. Session state is preserved for continued interaction

Implementation Considerations

- **Agent Design**: Define clear action groups based on educational workflows
- **Conversation Management**: Implement effective dialogue management strategies
- **Integration with LMS Data**: Create secure methods for agents to access relevant LMS data
- **User Permissions**: Apply role-based controls to agent capabilities
- **Usage Monitoring**: Track interaction patterns to improve agent effectiveness
- **Conversation History**: Manage conversation storage with appropriate privacy controls

Implementation examples

This section provides concrete implementation examples for different integration patterns discussed earlier. Each example includes code snippets, configuration steps and practical guidance focused on Moodle as the reference LMS platform.

These examples serve as starting points you can adapt to your specific requirements.

> **Note**
>
> The code snippets below are simplified to make them easier to understand, full working examples can be found at https://github.com/aws-samples/sample-moodle-integrations-on-aws with detailed deployment instructions in the README.md file. When building your own plugins, you should follow Moodle's coding standards and development best practices.

Moodle direct plugin integration

This example demonstrates how to create a simple block plugin for Moodle. It will follow a use case where a teacher would like to give students access to a translation tool for their French language class.

A complete sample for this pattern is available in the sample-moodle-integrations-on-aws repository:

- Moodle Plugin: `/moodle/plugin/blocks/block_aitranslator`
- AWS Infrastructure (CDK): `/cdk/constructs/moodle_aitranslator.py`
- API Gateway (CDK): `/cdk/constructs/apigateway.py`
- Lambda Authorizer: `/lambda/moodle_authorizer`
- Translation Lambda: `/lambda/translate`

Plugin structure

For this example, we've chosen a block plugin for its flexibility. Note that each plugin type can have a different folder structure and requirements, so make sure to check the Moodle documentation.

```
block_<plugin_name>/
### block_<plugin_name>.php // Code for the plugin
### db/
#   ### access.php // Defines access controls on the plugin
### lang/
#   ### en/
#       ### block_<plugin_name>.php // Language translations, en needed by default
### settings.php // Form to capture the API Gateway URI during plugin configuration
### version.php // Required for all plugins, defines plugin version + required moodle
 ver
```

As a developer, you can add more files for additional code logic in the plugin. Block plugins are ideal for adding UI components directly into course pages.

Setting up authentication for the plugin

Authenticating a plugin against your backend services is crucial to ensure only authorized users can make calls to your APIs and access AWS resources. In this example we will be using the Token feature of the Moodle Web Services to generate a token that can be validated by an API GatewayLambda Authorizer.

Plugin code

The block plugin requires 2 functions as part of its structure:

1. An `init()` method which runs on the plugin instantiation. You can use this function to set certain variables, for instance the plugin title.
2. A `get_content()` method which runs when the plugin is rendered on the page.

This translates to the following high level implementation code:

```
//block_translator.php

<?php

require_once($CFG->libdir . '/externallib.php');

class block_aitranslator extends block_base {

    public function get_content() {
```

```
        // Here you would typically have code to generate the HTML content for the
 block


        // Code to handle AJAX request coming from HTML components
        if (!empty($_POST['aitranslator_ajax']) && !
empty($_POST['aitranslator_question'])) {

            $question = required_param('aitranslator_question', PARAM_TEXT);

            $response = $this->get_ai_response($question);
            echo htmlentities($response, ENT_QUOTES, 'UTF-8');
            exit;
        }

        return $this->content;
    }
```

The `get_content()` methods embeds HTML code in the plugin to display its content. When the AJAX request is made the `get_ai_response()` method is called which will generate a token for the current user and make the call to the backend with the token included in the `Authorization` header.

The `get_user_token()` method is called from `get_ai_response()` to generate the token:

```
private function get_user_token() {
    global $DB;

    $service = $DB->get_record('external_services', ['shortname' => 'ai_translator']);

    $tokenobj = external_generate_token_for_current_user($service);

    return $tokenobj->token;
}
```

In this example the `ai_translator` service is a custom service that has been configured with appropriately scoped permissions, see Using web services.

Setting up the backend

To set up the backend, create an API Gateway REST API.

Once the API is created, create an API Gateway Lambda authorizer. The Lambda function called by the authorizer will make a call back to Moodle using the token for authentication, if successful this validates the request came from the intended Moodle system and the request can be authorized. Additional requests can be made to Moodle to get further information about the user making the request if necessary.

```
def validate_moodle_token(token: str, moodle_url: str) -> str:
    try:
        # Simple token validation - just check if token is valid
        data = {
            "wstoken": token,
            "wsfunction": "core_webservice_get_site_info",
            "moodlewsrestformat": "json",
        }

        url = f"{moodle_url}/webservice/rest/server.php"

        response = requests.post(url, data=data, timeout=10)
        response.raise_for_status()
        result = response.json()

        # Check for error responses
        if isinstance(result, dict) and ("exception" in result or "error" in result):
            error_code = result.get("errorcode", "unknown")
            error_msg = result.get("message", "Unknown error")
            logger.warning(
                "Token validation failed",
                extra={"error_code": error_code, "error_message": error_msg},
            )
            return None

        # Successful validation - extract username
        username = result.get("username")
        if username:
            return username
```

The `ai_translator` custom service must have the function `core_webservice_get_site_info` associated with it to allow the authorizer to successfully call the Moodle service method.

Then, create a POST method on the API which will be integrated with a Lambda function. Attach the authorizer to the method. For more information on creating APIs and methods, visit the API Gateway documentation.

Create the Lambda function which will call Amazon Bedrock and ask it to translate. Visit Lambda integrations for REST APIs API Gateway to know more about creating the function.

The following high level Python code shows how you can make a call to Amazon Bedrock using data passed by API Gateway.

```
def call_bedrock(data: str):

    conversation = [
        {
            "role": "user",
            "content": [{"text": """ I would like you to translate the following
 content into French. Just give me the translation and no more: """ + data}],
        }
    ]
    # Send the message to the model, using a default inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract the response text.
    response_text = response["output"]["message"]["content"][0]["text"]

    return {'output': response_text}
```

The code makes use of Bedrock's Converse API to call the model and get a translation.

Packaging

Once the final plugin structure is created it should be a zipped at the `block_<plugin_name>` level (not the block level) to be ready for installation.

Installation and configuration

There are two methods to install plugins on Moodle that haven't been registered in the Moodle plugins directory:

1. Uploading a ZIP archive through the user interface
2. Unzip directly to `/path/to/moodle/blocks/` directly on the server(s)

The most appropriate approach will depend on your Moodle environment, see Installing plugins for further details.

Once the plugin has been installed it can be configured through Site administration > Plugins > Blocks > `plugin_name`> which will display the configuration page defined in `settings.php`

Testing the plugin

Now that all the files have been packaged, the plugin uploaded and installed, you can enable the Moodle *Edit Mode* and add your plugin to your Moodle page.

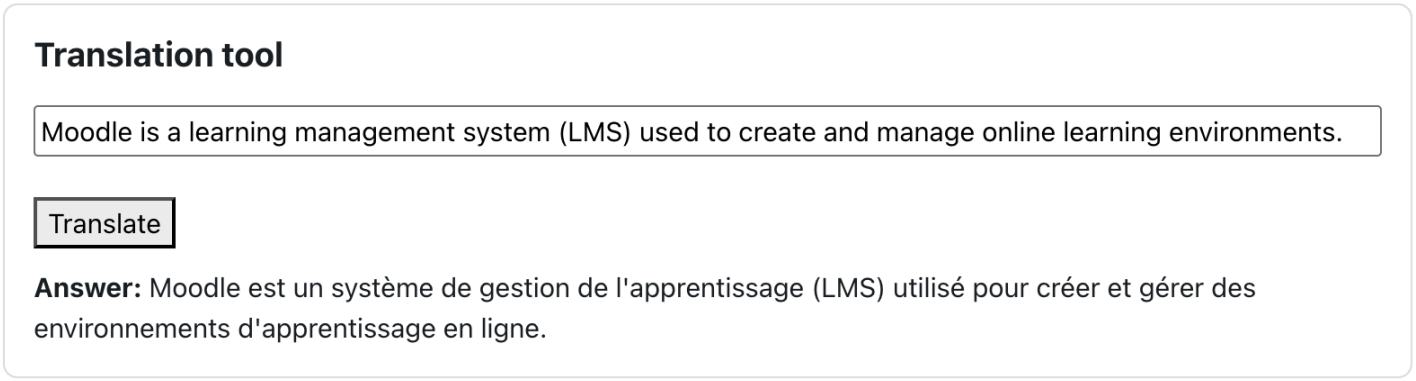

Figure 10: Translation tool powered by Amazon Bedrock

LTI Implementation

This example demonstrates how to create a basic LTI tool that integrates with Amazon Bedrock.

A complete sample for this pattern is available in the sample-moodle-integrations-on-aws repository:

- AWS Infrastructure (CDK): `/cdk/constructs/lti.py`
- API Gateway (CDK): `/cdk/constructs/apigateway.py`
- Lambda Backend: `/lambda/lti`
- React Frontend: `/lti_frontend`

The LTI process works using a handshake system, where both Moodle and the LTI compliant tool talk to each other and exchange information. The following diagram explains the process for LTI Advantage/1.3 (recommended):

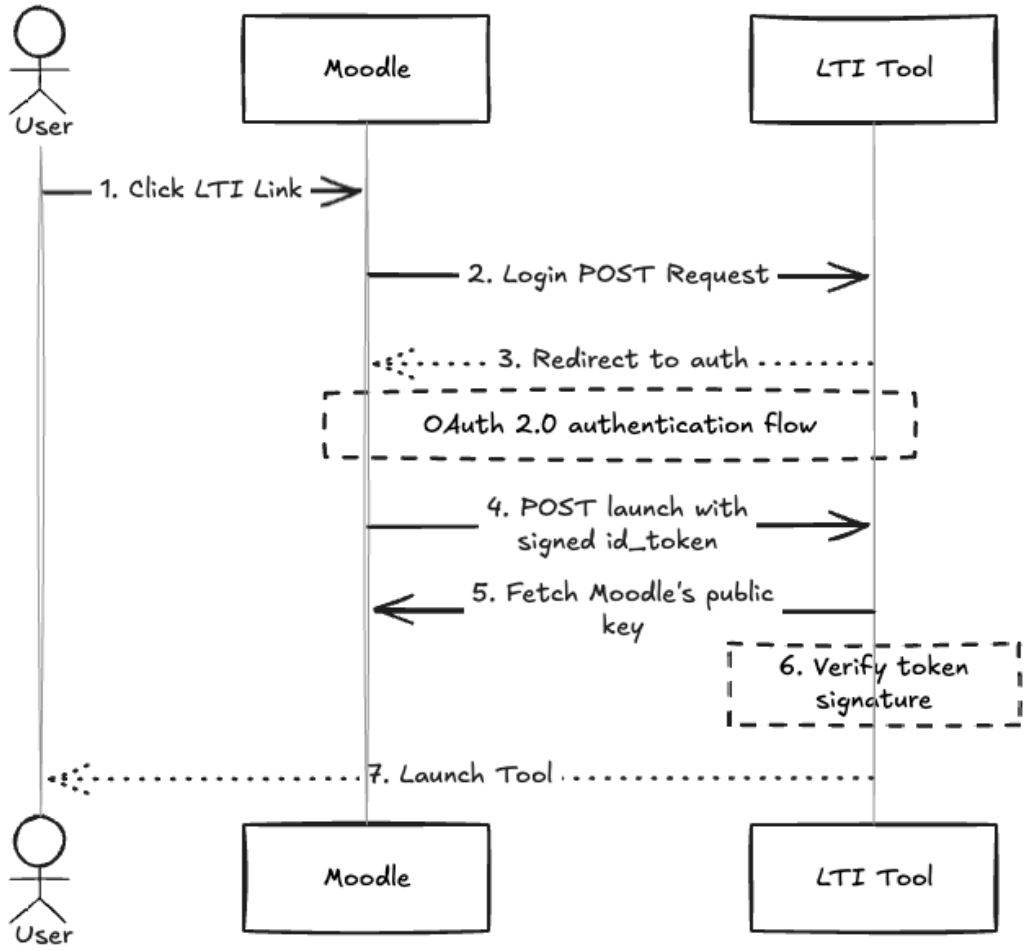

Figure 11: Moodle and LTI Sequence Diagram

The end user, inside Moodle, clicks on a course activity that makes use of a LTI compliant tool. The following sections will explain in more details how the tool can be configured to integrate with Moodle.

Tool registration

When creating a tool, you need to provide Moodle parameters and endpoints which will be used by the LMS to communicate with the tool.

Tool Parameter	Description	Example
Tool URL	The URL of your tool's **launch** endpoint	https://<id>.execute-api.<region>.amazonaws.com/prod/lti/launch
Public keyset	The URL of an endpoint which will return a **public key**	https://<id>.execute-api.<region>.amazonaws.com/prod/lti/key
Initiate login URL	The URL of your tool's **login** endpoint	https://<id>.execute-api.<region>.amazonaws.com/prod/lti/login
Redirect URI (s)	List of URI (s) that Moodle can redirect to. You would usually add here at least your **launch URI** as Moodle redirects there as part of the handshake process	NA

Once the tool is registered, Moodle will then provide you with parameters that will need to be used as part of the handshake process.

Moodle Parameter	Description	Example
Client ID	The moodle client ID	8HG7gkYnkB8nl4r
Public keyset URL	The URL of an endpoint to retrieve the keys used by Moodle when signing JWT tokens	https://moodle-publi-472qkfavulvq-48382734.us-east-2.elb.amazonaws.com/mod/lti/certs.php
Access token URL	The URL of Moodle's endpoint used when your tool needs to make additional calls back to	https://moodle-publi-472qkfavulvq-48382734.us-east-2.elb.amazonaws.com/mod/lti/token.php

	Moodle (for instance, sending grades back to Moodle)	
Authentication request URL	The URL of Moodle's auth service	https://moodle-publi-472qkfavulvq-48382734.us-east-2.elb.amazonaws.com/mod/lti/auth.php

Login request

When Moodle tries to launch your tool in an iframe/window (depending on the tool configuration), it makes a POST request to your `/login` endpoint with information such as `login_hint` (user ID) and `lti_message_hint` (launch context). The purpose of this endpoint is to redirect back to Moodle's authentication page, passing along these hints along with your tool's client ID and redirect URI.

The following Python code shows how this can be handled using a Lambda function behind a API Gateway `/login` endpoint. The sample in the repo shows a more complete implementation include Cross-Site Request Forgery (CSRF) protection:

```
# Extracting login_hint and lti_message_hint from payload
login_hint = event.get("login_hint")[0]
lti_message_hint = event.get("lti_message_hint")[0]

# Generate secure random tokens for CSRF protection
state = str(uuid.uuid4())
nonce = str(uuid.uuid4())

# Generating auth URL with query string parameters
url = (
    f"{OIDC_AUTH_URL}?"
    f"scope=openid&"
    f"response_type=id_token&"
    f"client_id={client_id}&"
    f"redirect_uri={urllib.parse.quote(redirect_url)}&"
    f"login_hint={urllib.parse.quote(login_hint)}&"
    f"state={state}&"
    f"nonce={nonce}&"
    f"response_mode=form_post&"
    f"prompt=none&"
```

```
    f"lti_message_hint={urllib.parse.quote(lti_message_hint)}"
)

# Response to redirect to Moodle's auth
return Response(
    status_code=302,
    headers={"Location": url},
    body=""
)
```

In this code, we use the values Moodle provided for the **CLIENT_ID** and **REDIRECT_URL** and constructed a URL with query string parameters to point to Moodle's auth service. Some parameters include:

- Response type: In this case *id_token* asks Moodle's auth to come back with a JWT ID token
- Redirect URI: This is the `/launch` endpoint that Moodle will redirect to once completing the auth process. This URI must be registered during tool creation (see section Tool registration)

The application running in the iframe is then redirected to the Auth service, which upon completion will do another POST request to the `/launch` endpoint.

POST request to /launch signed with JWT token

In the POST request Moodle makes to the `/launch` endpoint, it adds to the payload a JWT token. Decoding this token will provide the LTI tool the ability to extract information about the user launching the tool and the course that it is launched from. It can also decode any custom parameters that Moodle is configured to send to this tool. The launch endpoint needs to follow these steps:

1. Verify that the token comes from Moodle. This can be done by getting the key Moodle used to sign the token. This key can be retrieved using the **Public keyset URL** given by Moodle. Upon retrieving that key, the code can verifies Moodle's signature on the token.
2. Decode the token and extract relevant information for the tool.
3. Generate a URL combining the Tool's hosted URL (https://cloudfront-disro.net for instance) and a session token containing the information retrieved from the JWT token.
4. Redirect the iframe to the generated URL.

The following python code shows how this can be done using Lambdaand API Gateway.

```
# Extract user information from API Payload
user_info = {
    'user_id': payload.get("user_id"),
    'name': payload.get("name"),
    'email': payload.get("email"),
}

# Generate signed JWT session token
jwt_secret = get_jwt_secret()
token_id = str(uuid.uuid4())
token = jwt.encode(
    {
        **user_info,
        'exp': datetime.now(timezone.utc) + timedelta(seconds=TOKEN_EXPIRY_SECONDS),
        'iat': datetime.now(timezone.utc),
        'jti': token_id
    },
    jwt_secret,
    algorithm='HS256'
)

# Construct the redirect URL
redirect_url = f"{WEBSITE_URL}#token={token}"

return Response(
    status_code=302,
    headers={"Location": redirect_url},
    body=""
)
```

Tool launch

Once the `/launch` endpoints completes its processing, it redirects the app in the iframe to the tool's URL, which will allow the user to see and use the tool. The tool is able to display the course it is being launched from as well as the User ID thanks to the session token passed as part of the redirect.

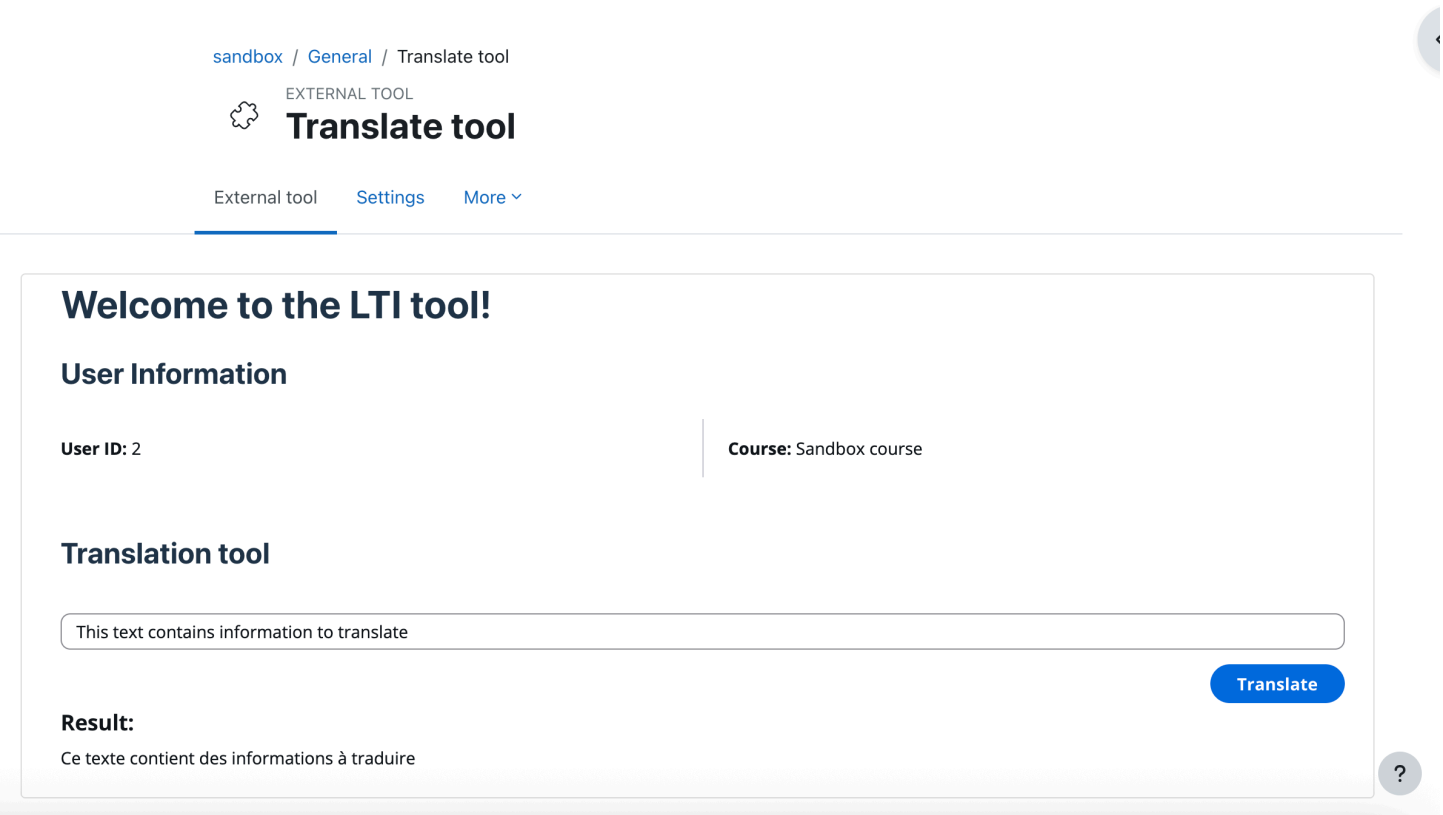

Figure 12: LTI-based Translation tool

The tool can then have its own backend, to talk to services like Amazon Bedrock.

Passing data back to Moodle

The following diagram shows how an LTI compliant tool can pass data back into Moodle. It takes the example of a tool that would like to submit grades back.

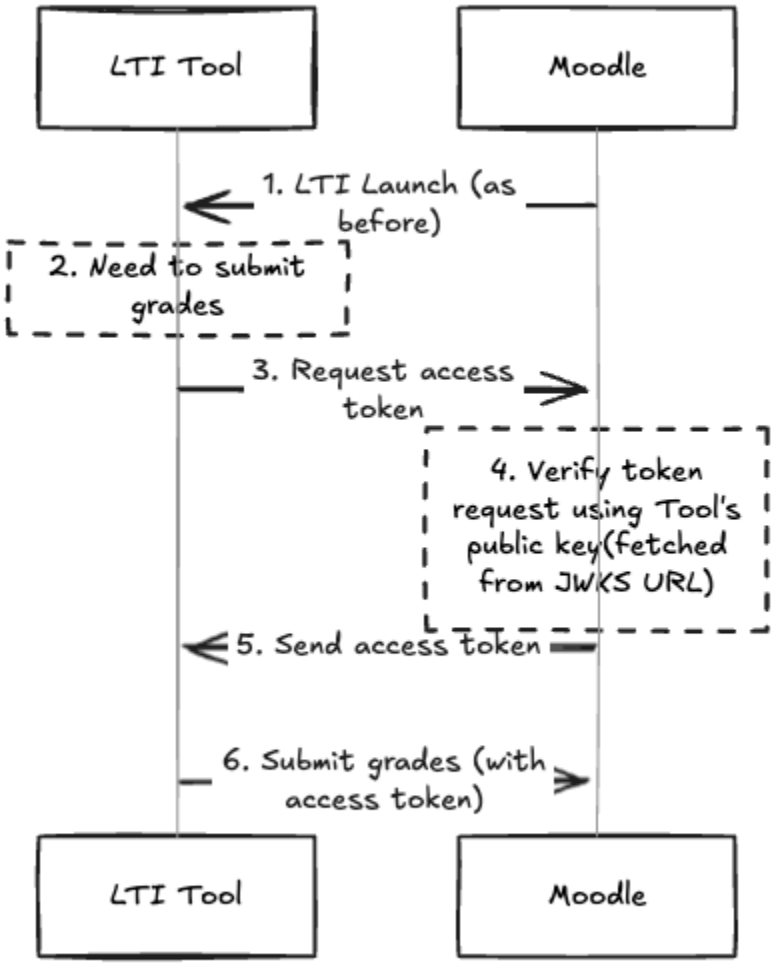

Figure 13: Moodle LTI Sequence Diagram

1. The LTI tool is launched using the process explained in the previous sections
2. The tool requests an access token using Moodle's access token URL (provided upon tool registration)
3. Moodle then checks the request by validating the key against the one returned by the tool's configured **Public Keyset URL**
4. Moodle sends an access token back upon successful validation
5. The tool sends the grades to Moodle using the access token given by Moodle for authentication

Event driven

Educational institutions struggle to deliver timely, accurate information to students, faculty, and staff. Retrieval-Augmented Generation (RAG) services solve this by combining generative AI with institutional knowledge, creating dynamic platforms that retrieve and generate contextual responses.

This section will show how to build a Moodle local plugin that forwards events from the Moodle events system to Amazon EventBridge. The next section (API Integration) will demonstrate how the events can be used to retrieve the content from Moodle and update Amazon Bedrock Knowledge Bases in near real-time. This ensures users access current content while maintaining normal Moodle workflows.

A complete sample for this pattern is available in the sample-moodle-integrations-on-aws repository:

- Moodle Plugin: `/moodle/plugin/local/local_awsevents`
- AWS Infrastructure (CDK): `/cdk/constructs/moodle_events.py`
- Event Handlers (CDK): `/cdk/constructs/moodle_event_handlers.py`

Create the plugin structure

The plugin uses Moodle's local plugin type, which is the recommended approach for event consumers that communicate with external systems.

```
local_<plugin_name>/
### classes // Autoloaded classes
#   ### aws_eventbridge.php // Called by observer to send the event to EventBridge
#   ### observer.php // Invoked as events are observed
### db
#   ### events.php // Define event subscriptions that the plugin listens for
### lang
#   ### en
#       ### local_<plugin_name>.php // Language translations, en needed at a minimum
### settings.php // Define settings for the plugin
### version.php // Required for all plugins, defines plugin version + required Moodle
 ver
```

This structure shows the required files for the plugin to work and be detected by Moodle. As a developer, more files can be added for additional code as required.

Implement the configuration interface

The `settings.php` file allows an administrator to configure the plugin with the required parameters. The strings are retrieved from the language file.

The code first creates the setting page and adds a number of different options for configuring the plugin. The options would include how to authenticate with the AWS API and the name of the target EventBridge event bus.

Define the event subscriptions

The `events.php` file is used to configure which events the plugin should process. The snippet below is subscribing to the `course_module_created` event.

```
$observers = [
    // Observer for course module created events
    [
        'eventname' => '\core\event\course_module_created',
        'callback'  => '\local_awsevents\observer::process_event',
    ],
];
```

Process the event

The `process_event` method of the `observer` class is invoked for each event that has been subscribed to. The main functionality is to instantiate the `aws_eventbridge` class and invoke the `send_event` method.

```
class observer {

    public static function process_event(\core\event\base $event) {
        global $CFG;

        // Initialize AWS EventBridge handler
        $handler = new aws_eventbridge();
```

```
        // Send event to AWS EventBridge
        $result = $handler->send_event($event);

    }
}
```

Send the event to EventBridge

The constructor in the `aws_eventbridge` class retrieves the connection details from the `CFG` object. An `EventBridgeClient` is created using the AWS SDK for PHP and the event bus name is stored in a local property. The client is declared as a static variable to enable connection reuse across multiple invocations of the observer. This is a performance optimization that prevents the overhead of creating a new AWS SDK client for each event processed.

```
<?php

namespace local_awsevents;

use Aws\EventBridge\EventBridgeClient;
use core\event\base;

defined('MOODLE_INTERNAL') || die();
class aws_eventbridge {
    /** @var EventBridgeClient Cached AWS EventBridge client instance */
    private static $client = null;
    private $eventbus;

    public function __construct() {
        global $CFG;

        // Initialize client only once (cached across multiple events)
        if (self::$client === null) {
            // Get authentication method
            $auth_method = !empty($CFG->local_awsevents_auth_method) ? $CFG-
>local_awsevents_auth_method : 'role';
            // Prepare client configuration
            $config = [
                'version' => 'latest',
                'region'  => $CFG->local_awsevents_region
            ];
            // For role authentication, we don't need to specify credentials
            if ($auth_method === 'key') {
```

```
                $config['credentials'] = [
                    'key'    => $CFG->local_awsevents_key,
                    'secret' => $CFG->local_awsevents_secret,
                ];
            }

            // Initialize AWS EventBridge client (cached)
            self::$client = new EventBridgeClient($config);
        }

    $this->eventbus = $CFG->local_awsevents_eventbus;
    }
}
```

The `send_event` method builds the structure of the payload based on the Moodle event and calls the `putEvents` method from the `EventBridgeClient` to write the event to EventBridge. As EventBridge has relatively high service limits for the `PutEvents` API and is being invoked as a background process without the need to authenticate individual identities in this scenario it is recommended to integrate directly with the AWS service rather than proxying through API Gateway.

```
public function send_event(base $event): bool {
        global $CFG;

        // Prepare event data
        $eventData = [
            'Entries' => [
                [
                    'EventBusName' => $this->eventbus,
                    'Source' => 'moodle.events',
                    'DetailType' => $event->eventname,
                    'Detail' => json_encode([
                        'eventname' => $event->eventname,
                        'component' => $event->component,
                        'action' => $event->action,
                        'target' => $event->target,
                        'objecttable' => $event->objecttable,
                        'objectid' => $event->objectid,
                        'crud' => $event->crud,
                        'edulevel' => $event->edulevel,
                        'contextid' => $event->contextid,
                        'contextlevel' => $event->contextlevel,
```

```
                        'contextinstanceid' => $event->contextinstanceid,
                        'userid' => $event->userid,
                        'courseid' => $event->courseid,
                        'relateduserid' => $event->relateduserid,
                        'anonymous' => $event->anonymous,
                        'other' => $event->other,
                        'timecreated' => $event->timecreated
                    ]),
                    'Time' => new \DateTime()
                ]
            ]
        ];

        // Send event to EventBridge
        $result = $this->client->putEvents($eventData);

        return true;
}
```

Packaging

As the example plugin has a dependency on the AWS SDK for PHP this needs to be installed with the plugin. There are three approaches for this:

1. **Include AWS SDK for PHP in the plugin**: Before creating the ZIP archive of the plugin for installation, the AWS SDK for PHP should be installed into the plugin folder structure. This will lead to a larger plugin but will ensure the version of the AWS SDK for PHP deployed has been tested with the plugin and if the plugin is copied directly to your Moodle servers the dependency is included.
2. **Deploy AWS SDK for PHP during deployment**: By adding a `db/install.php` file to the plugin you can script the installation of the AWS SDK for PHP as your plugin is installed. The script is only called during the installation so depending on the Moodle environment this may not work in multi server environments.
3. **Depend on local_aws plugin**: The local_aws plugin installs the the AWS SDK for PHP on the server. Adding a dependency on this in the `version.php` file will ensure it is available before the local plugin can be installed.

Once the final plugin structure is created it should be a zipped at the `<plugin_name>` level (not the local level) to be ready for installation.

Installation and configuration

There are two methods to install plugins on Moodle that haven't been registered in the Moodle plugins directory:

1. Uploading a ZIP archive through the user interface
2. Installation directly on the server(s)

The most appropriate approach will depend on your Moodle environment, see Installing plugins for further details. Once the plugin has been installed it can be configured through Site administration > Plugins > Local plugins > `<plugin_name>` which will display the configuration page defined in `settings.php`.

API integration

The previous section described how Moodle events can be sent to EventBridge. This section explains how those events can be used to populate a Amazon Bedrock Knowledge Base which can be used for RAG services.

A complete sample for this pattern is available in the sample-moodle-integrations-on-aws repository:

- Lambda Function: `/lambda/index_moodle_file`
- AWS Infrastructure: Referenced in `/cdk/constructs/moodle_event_handlers.py`

Configure EventBridge

The example plugin sends Moodle events to a custom EventBridge event bus. Using a custom event bus for application events is a best practice. Create the event bus by following the Creating an event bus inAmazon EventBridge guide and configure the event bus name in the plugin settings.

The event bus receives Moodle events, this example will focus on events are triggered by a course module being created. Create anEventBridge rule to capture these specific events. A rule contains an event pattern and a target. An example event pattern could be:

```
{
```

```
  "source": ["moodle.events"],
  "detail": {
    "action": ["created"],
    "eventname": ["\\core\\event\\course_module_created"]
  },
}
```

The `aws_eventbridge` class in the plugin sets the event source to `moodle.events` for all events passed to EventBridge. Detail attributes are derived from the original Moodle event. The event pattern can be extended to include other events that are relevant such as:

- `\core\event\course_module_updated`
- `\core\event\course_module_deleted`

Create Bedrock Knowledge Base

A Bedrock Knowledge Base can be created following this guide to index and store the generated vectors. A **Custom** data source type should be selected and **AmazonOpenSearch Serverless** chosen as the Vector Store.

Configure Moodle

The Lambda function will call the Moodle REST API. To enable this, configure Moodle web services:

- Navigate to: Site Administration > Server > Web services > Overview
- Complete the following steps:
  - Enable web services
  - Enable the **REST** protocol
  - Create a user with required capabilities
  - Create an external service:
    - Select "**Can download files**" option
    - Assign the **core_course_get_contents** function
    - Assign the previously created user
  - Create a token for the service and user
  - After saving changes, keep the token page open for the next step

Create Secret

Store the Moodle token using AWS Secrets Manager. Following this guide, using the OAuth token example for secret creation with the Moodle token as the value.

Lambda Function

The Lambda function processes EventBridge events through these key steps:

- Retrieve the Moodle access token from Secrets Manager

```
def get_moodle_token() -> str:
    """Retrieve Moodle API token from AWS Secrets Manager."""
    response = sm_client.get_secret_value(
        SecretId=os.environ["MOODLE_TOKEN_SECRET_ARN"]
    )
    return response["SecretString"]

TOKEN = get_moodle_token()
```

- Extract course_id and module_id from the event

```
@event_source(data_class=EventBridgeEvent)
def lambda_handler(
    event: EventBridgeEvent, context: LambdaContext
):
    course_id: int = event.detail["courseid"]
    module_id: int = event.detail["objectid"]
```

- Use Moodle REST API to retrieve the course contents

```
def get_course_info(course_id: int):
    # Call Moodle web service to get course contents
    response = requests.get(
        f"{MOODLE_URL}/webservice/rest/server.php",
        params={
            "wstoken": TOKEN,
            "wsfunction": "core_course_get_contents",
```

```
            "moodlewsrestformat": "json",
            "courseid": course_id,
        },
        timeout=10,
    )
    response.raise_for_status()
    return response.json()
```

- Process the response of `core_course_get_contents` to find the specific module that was passed in the event (note this is implemented at the course level as the `core_course_get_course_module` API does not return details to download the file)
- Retrieve file download URI and mime type for the files in the module
- Download the file from the specified URI and save locally

```
response = requests.get(f"{file_info.file_url}&token={TOKEN}", timeout=30)
response.raise_for_status()

with open(file_info.file_path, "wb") as f:
      f.write(response.content)
```

- Ingest the file directly into the Amazon Bedrock Knowledge Base

```
with open(file_info.file_path, "rb") as f:
    file_content = f.read()

# Ingest document into Bedrock Knowledge Base
response = bedrock_agent_client.ingest_knowledge_base_documents(
    knowledgeBaseId=KNOWLEDGE_BASE_ID,
    dataSourceId=DATA_SOURCE_ID,
    documents=[
        {
            "content": {
                "custom": {
                    "customDocumentIdentifier": {"id": file_info.file_url},
                    "inlineContent": {
                        "byteContent": {
                            "data": file_content,
                            "mimeType": file_info.mime_type,
                        },
```

```
                        "type": "BYTE",
                    },
                    "sourceType": "IN_LINE",
                },
                "dataSourceType": "CUSTOM",
            }
        }
    ],
)
```

The Lambda function should be configured with reserved concurrency to prevent excessive API calls to Moodle during multiple content change events.

Next Steps

- Review the sample code repository for complete working implementations of each integration pattern.
- Identify which integration pattern best fits your use case by referring to the comparison tables in the Integration Options Overview section.
- Set up a development environment with a test LMS instance and an AWS account to prototype your chosen pattern.
- For production deployments, review the Guidance for Deploying Moodle LMS on AWS reference architecture.

Resources

AWS services

- AWS Amplify
- Amazon API Gateway
- Amazon AppFlow
- Amazon Bedrock
- Amazon Bedrock Agents
- Amazon Bedrock Guardrails
- Amazon Bedrock Knowledge Bases
- AWS Elastic Beanstalk
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Amazon EventBridge
- AWS Glue
- AWS Identity and Access Management
- Amazon Kinesis Data Streams
- AWS Lambda
- Amazon Managed Streaming for Apache Kafka
- AWS Secrets Manager
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Simple Storage Service

LMS references

- Blackboard Learn Activity Streams
- Canvas Live Events

- Moodle AI Plugins
- Moodle Amazon Bedrock API provider
- Moodle Plugin Types

Reference architectures

- Guidance for Deploying Moodle Learning Management System on AWS
- Modernize Moodle LMS with AWS serverless services
- Moodle Reference Architecture

Guidance

- Transforming application development and maintenance operating models on AWS with generative AI

Patterns

- Performance Efficiency - Monitoring and Observability with Lambda Powertools for your API Application
- Build Rapid REST API with Lambda Powertools Python and CDK
- Check AWS Cloud Development Kit (AWS CDK) applications or CloudFormation templates for best practices by using cdk-nag rule packs
- Integrating microservices by using AWS serverless services

Workshops

- Higher Education Data Lake Immersion Day
- Amazon SageMaker Unified Studio Project - Improve Student Engagement

Standards

- Caliper Specification

- Learning Tools Interoperability (LTI)

Case studies

- Anthology uses embedded analytics offered by Amazon Quick Sight to democratize decision making for higher education
- Canvas (LMS) integration with Amazon Connect
- Coventry University Migrates Moodle to AWS
- How UCL migrated its Moodle virtual learning environment to the cloud in 10 weeks

Attachments

To access additional content that is associated with this document, unzip the following file:

attachment.zip

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Initial publication	—	June 2026

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

A2A (Agent-to-Agent)

A stateful protocol for agent-to-agent collaboration supporting task delegation and state transfer.

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

Agent

An AI system that can autonomously reason, plan, and take actions using tools to achieve goals.

Agent Ops

Operational practices for building, testing, deploying, and running AI agents in production at scale.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include SUM and MAX.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities.

For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

Citizen Developer

A business user who creates AI applications using no-code/low-code platforms without specialized technical skills.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in

an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.

- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the "2021-05-27 00:15:37" date into "2021", "May", "Thu", and "15", you can help the learning algorithm learn nuanced patterns associated with different data components.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

FM gateway

A centralized intermediary that controls and normalizes access to foundation models. Also known as an *LLM gateway*.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub CSPM, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

guardrails (AI)

Safety mechanisms that filter, validate, and constrain agent inputs and outputs to help ensure responsible and safe AI behavior.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a machine learning model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

human-in-the-loop (HitL)

A workflow pattern where agent execution pauses for human review and approval at critical decision points.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this

period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally

move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large language model (LLM)

A deep learning AI model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see What are LLMs.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

MCP

See Model Context Protocol.

Model Context Protocol (MCP)

A stateless protocol for agent-to-tool communication.

MCP server

A service that exposes one or more tools through the Model Context Protocol.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic PUT and DELETE requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

privacy by design

A system engineering approach that takes privacy into account through the whole development process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

prompt chaining

Using the output of one LLM prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can

publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

RAG

See Retrieval Augmented Generation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see What is RAG.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata.

The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

A system engineering approach that takes security into account through the whole development process.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services

or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

Shadow AI

Unauthorized AI applications built or used outside of governed channels within an organization.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

tool

A function or API that an agent can invoke to perform operations in external systems.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zero-shot prompting

Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.