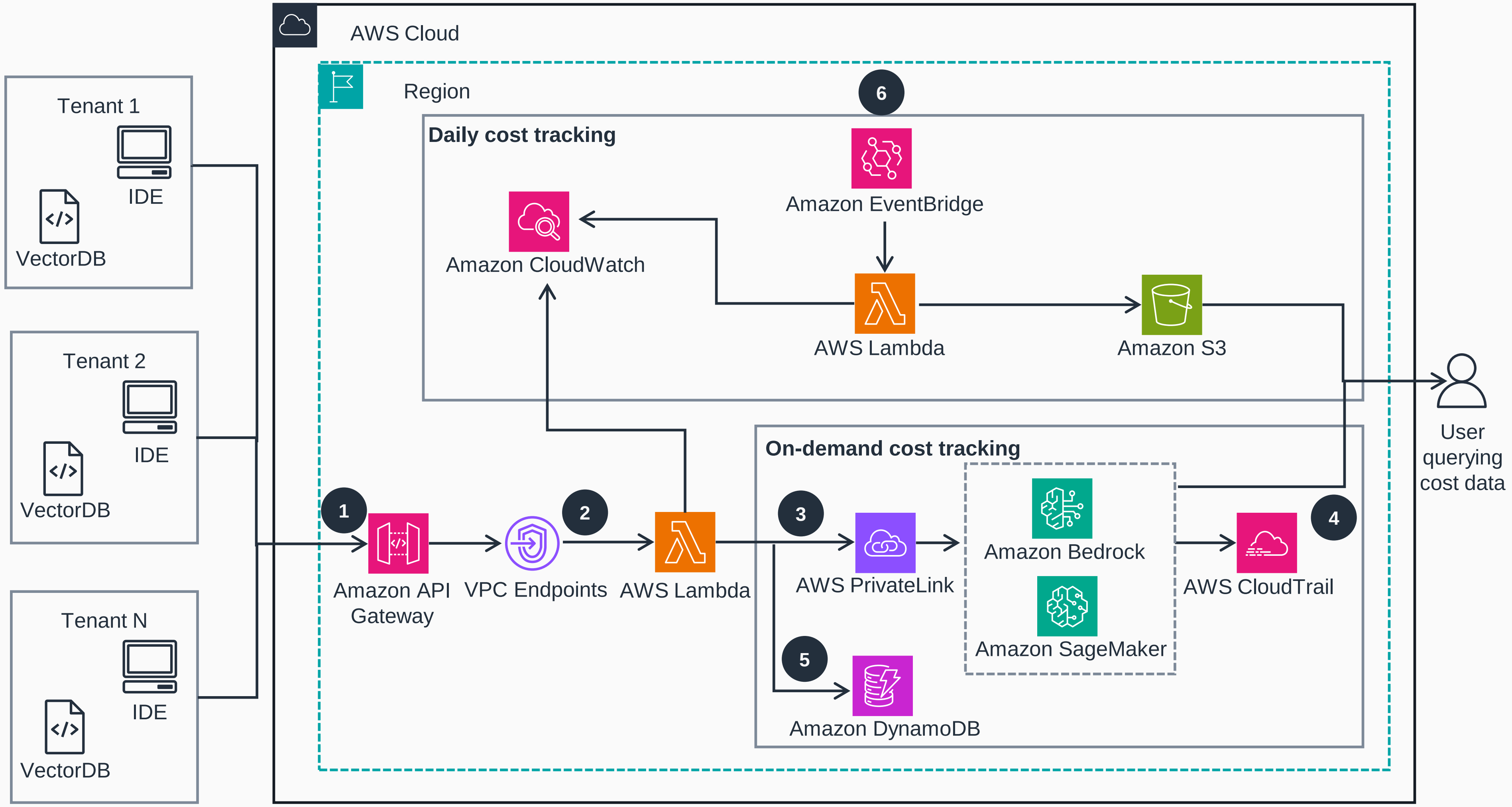
Guidance for a Multi-Tenant, Generative AI Gateway with Cost and Usage Tracking on AWS

Configure a multi-tenant SaaS model for generative AI

This architecture diagram shows how AWS customers can build an internal Software-as-a-Service (SaaS) model for access to artificial intelligence (AI) models, referred to as foundation models (FMs). The model usage and costs can be managed and tracked for each tenant.



1. The Software-as-a-Service (SaaS) **AWS Cloud Development Kit** (CDK) template deploys all required resources to the AWS account. After deployment, a tenant's application sends a POST request to Amazon API Gateway, invoking a specific model by passing the *model_id* as a request parameter and the appropriate model payload in the request body. The tenant can be an individual user, a specific project, a team, or even an entire department within a company.

2. **API Gateway** routes the request to AWS Lambda. The **Lambda** function logs usage information for each tenant in Amazon CloudWatch and invokes the foundation model.

3. **Amazon Bedrock** provides a virtual private cloud (VPC) endpoint powered by **AWS PrivateLink**. In the **CDK** configuration file, you can specify **Amazon SageMaker** endpoints. You can use the **SageMaker** endpoint name as the *model_id* to route the request to the appropriate **SageMaker** endpoint. The **Lambda** function sends the request to **Amazon Bedrock** or **SageMaker**.

4. After the invocation, AWS CloudTrail generates an event. If the call is successful, the **Lambda** function logs usage information and returns the generated response to the application. The type of information logged varies depending on the invoked model and service. It includes details such as team ID, request ID, model ID, input tokens, output tokens, image dimensions, and other relevant information.

5. (Optional) If the model request includes response streaming, the **Lambda** function stores the generated response in **Amazon DynamoDB**.

6. An **Amazon EventBridge** rule runs a **Lambda** function that retrieves the previous day's usage data from **CloudWatch** on a regular basis. The function calculates the associated costs and stores the aggregated data by *team_id* and *model_id* in a CSV file format to Amazon Simple Storage Service (Amazon S3). To query and visualize the data stored in **Amazon S3**, AWS customers have different options, including Amazon S3 Select, **Amazon Athena**, and **Amazon QuickSight**.

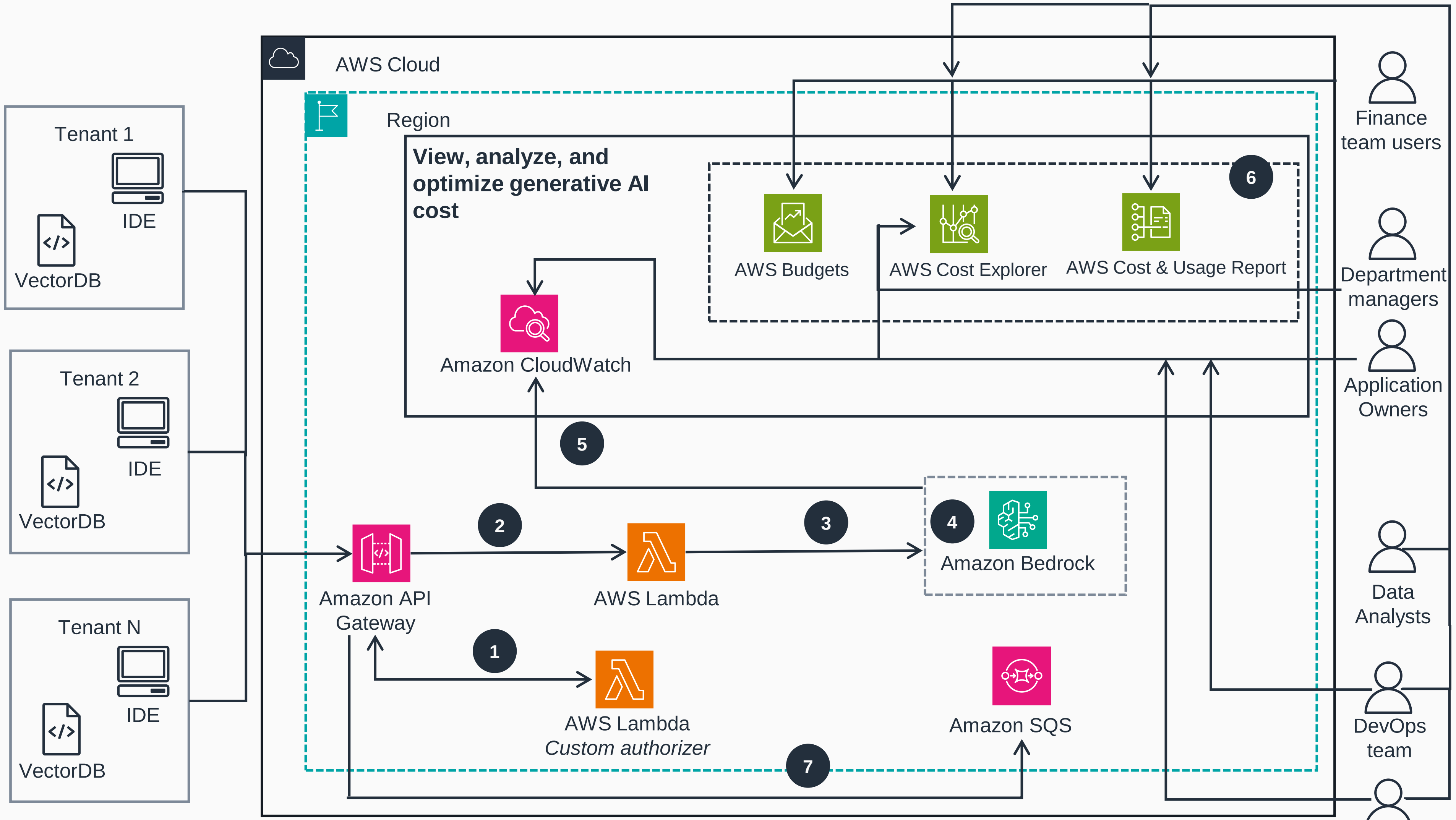
Reviewed for technical accuracy November 20, 2024

AWS Reference Architecture

Guidance for a Multi-Tenant, Generative AI Gateway with Cost and Usage Tracking on AWS

Track cost and usage for generative AI models

This architecture diagram shows how AWS customers can view, analyze, and optimize generative AI costs. It consists of 7 steps with steps 6-7 on the next slide.



1 **AWS Organizations** is used to manage multiple AWS accounts, enabling centralized governance, resource management, and cost allocation across different environments or departments for **Amazon Bedrock** and application inference profiles. Within this multi-account architecture, a tenant's application sends a POST request to **API Gateway**. **API Gateway** first validates the tenant's API key, then invokes a **Lambda** authorizer function which can perform custom authentication and authorization logic based on the key and other request details before allowing access to the API endpoint.

2 A specific model is invoked by passing the application inference profile ARN as a request parameter and the model payload in the request body. **API Gateway** routes the request to **Lambda**.

3 The **Lambda** function retrieves the appropriate application inference profile ARN based on tags, including tenant ID, project ID, and department ID, using either a static configuration or dynamic retrieval through the AWS Resource Groups API. The **Lambda** function invokes the foundation model using the **Amazon Bedrock** API (for example, Converse, InvokeModel) with the application inference profile (retrieved profile ARN) which supports cross-Region inference.

4 When **Amazon Bedrock** processes these requests, it generates runtime metrics associated with the specific inference profile used. When using **Amazon Bedrock** with application inference profiles, the information that can be logged are: Application inference profile ARN, request ID, input tokens count, output tokens count, model ID or ARN, latency or response time, timestamp, custom tags (tenant ID, project ID, department ID), API name used, Region, error codes or messages, billing or cost information, and throttling or rate limiting events.

5 **Amazon Bedrock** integrates with **CloudWatch** to monitor runtime metrics, allowing organizations to visualize data and set alarms based on application inference profile ARNs, enabling real-time cost management and resource optimization for generative AI workloads.

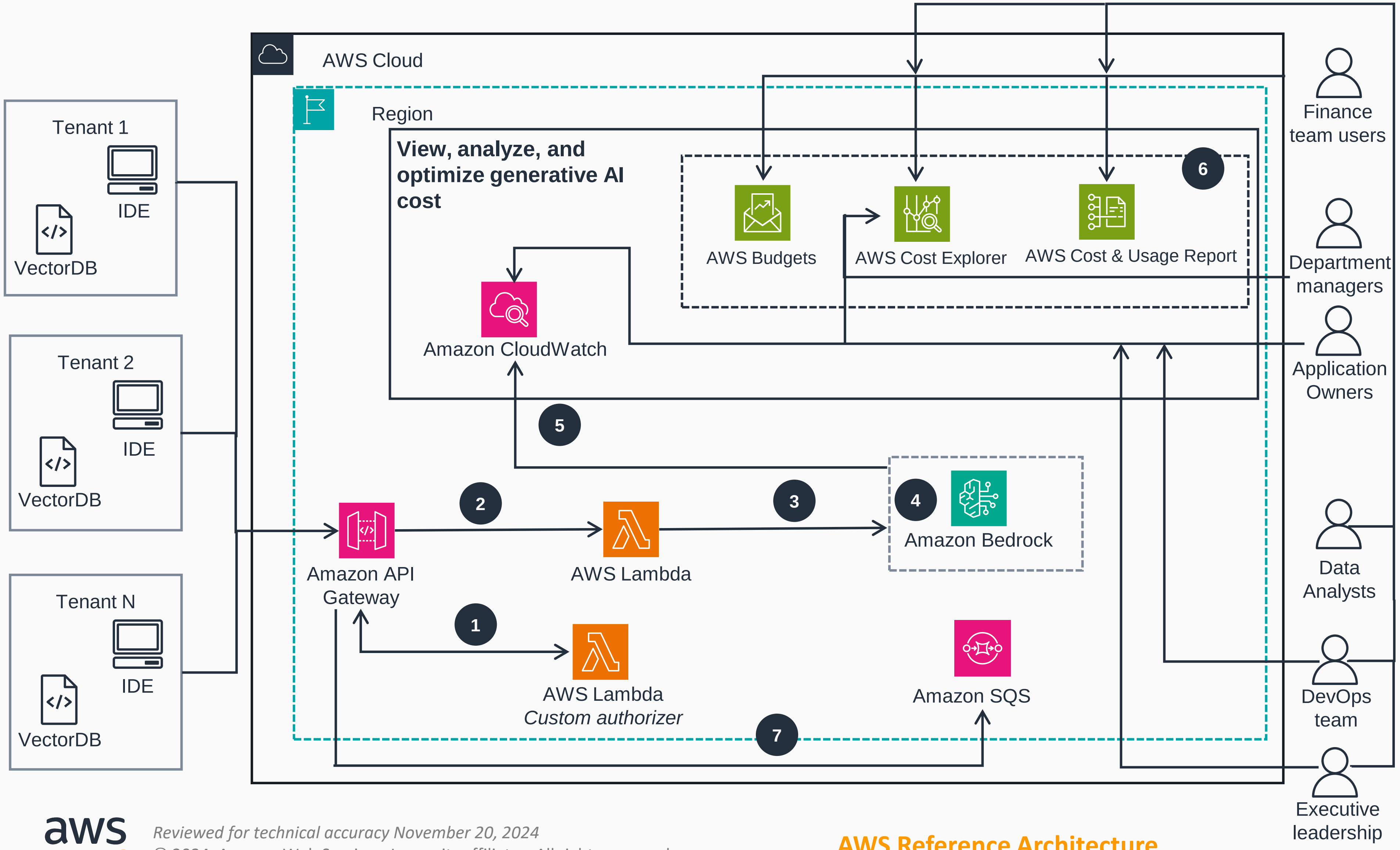
Reviewed for technical accuracy November 20, 2024

AWS Reference Architecture

Guidance for a Multi-Tenant, Generative AI Gateway with Cost and Usage Tracking on AWS

Track cost and usage for generative AI models

Steps 6-7



View, analyze, and optimize generative AI cost

6 Finance teams can use AWS Budgets to set tag-based thresholds and receive alerts as spending approaches limits. They can use AWS Cost Explorer for daily, weekly, and monthly analysis of tagged resources, and use AWS Cost and Usage Reports (AWS CUR) for detailed spending breakdowns. Finance teams can also use AWS Cost Anomaly Detection to automatically identify unusual spending patterns.

Department managers can monitor spending trends specific to their department using AWS Cost Explorer, filtering by tags like *dept:sales*. They can set department-specific budgets and receive notifications through AWS Budgets.

Application owners can use AWS Cost Explorer to analyze costs filtered by application-specific tags (such as app:chat_app). They can set up alarms in **CloudWatch** based on application inference profile ARNs to monitor usage.

Data analysts can work with AWS CUR data using **Athena** and **QuickSight** to create custom reports and visualizations. This allows them to dive deep into the data, uncovering insights and trends.

DevOps teams can monitor runtime metrics in **CloudWatch**, filtered by the application's inference profile ARN. They can set up **CloudWatch** alarms for invocation limits and other operational metrics.

Executive leadership can use AWS Cost Explorer for high-level views of AI spending across the organization. They can also use AWS Cost Anomaly Detection to identify unusual spending patterns.

7 (Optional) **API Gateway** doesn't support streaming directly, so the AWS Solution Constructs uses WebSockets and **Amazon Simple Queue Service** (Amazon SQS) to create a real-time, bidirectional communication and the API.

Reviewed for technical accuracy November 20, 2024

AWS Reference Architecture