RBAC for conversational systems

Introduction:

In modern conversational systems (GenAI chatbots), service requests expressed in natural language must be interpreted to identify the resource a user is trying to access and the action they intend to perform. The solution takes a user query and an access token (from Amazon Cognito), extracts the intended action and resource using an LLM, and evaluates the request using Amazon Verified Permissions (AVP). AVP, with Cognito as the identity source and Cedar-based policy definitions, ensures that authorization decisions are enforced consistently and centrally. Developers can integrate this into their GenAI applications to streamline compliance and deliver secure conversational experiences.

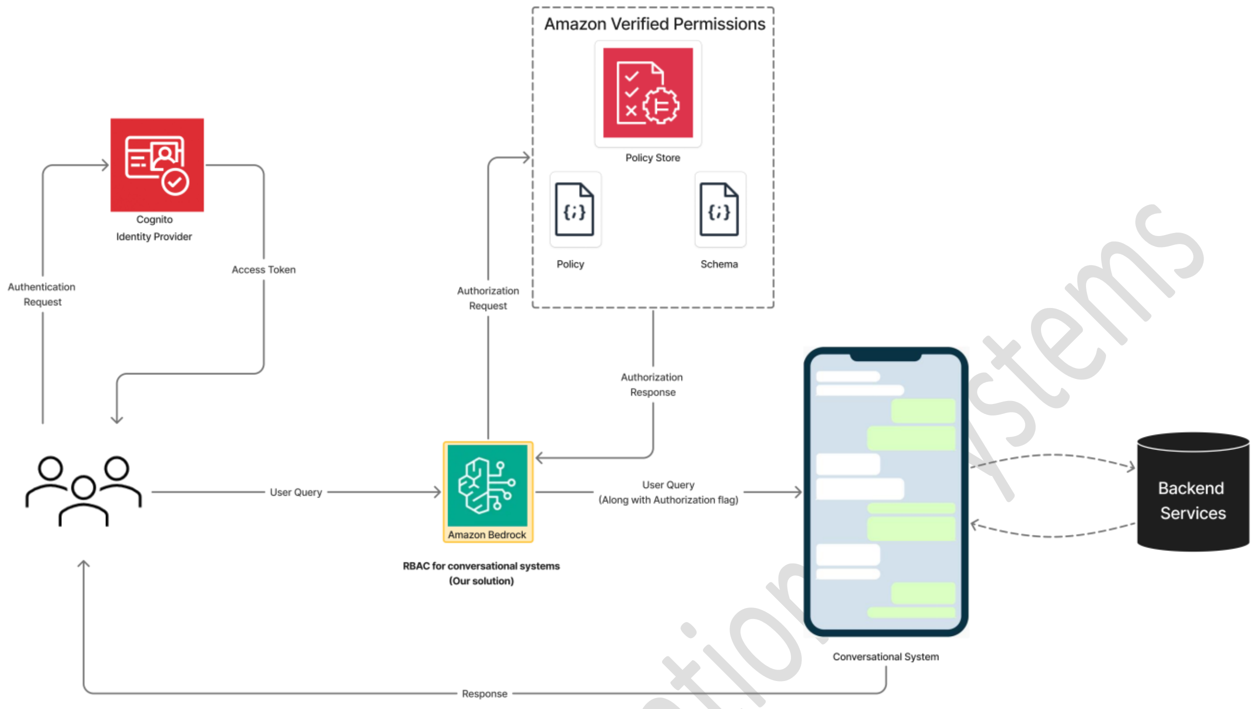
In UI-based systems, resources and actions are well-defined through buttons, forms, and APIs, making it straightforward to enforce RBAC. In LLM-based chatbots, the same information is buried in unstructured natural language, creating the challenge of accurately extracting intent before authorization can even happen. Our solution is designed to address this challenge.

This solution combines LLM-powered intent extraction with enterprise-grade access control to securing natural language interfaces. It converts user queries into structured action-resource pairs, which are evaluated against fine-grained Cedar policies using Amazon Verified Permissions (AVP). With built-in integration to Amazon Cognito, user identities are seamlessly authenticated via access tokens and mapped directly to authorization logic, ensuring secure and compliant access to backend services.

Key highlights:

1. LLM-powered Action-Resource Extraction: Translates free-form user queries into structured action-resource pairs for accurate enforcement.
2. Fine-Grained Access Control with Cedar + AVP: Secure every API or data interaction with policy-based authorization tied to roles, hierarchies, and relationships.
3. Seamless Auth Integration with Cognito: Leverages the user's access token to map identity directly to permission logic, reducing friction and improving compliance.

Usage Flow:



Pre-requisites:

1. Please ensure the following components are already set up in your AWS environment:
    a. AWS Verified Permissions:
        i. You must have **AWS Verified Permissions** enabled in your account.
        ii. A **Policy Store** should be created and active.
        iii. The **Policy Store** must use your Amazon Cognito user pool (see below) as the **identity source**.
        iv. Refer to Appendix for steps for creating a policy store with identity source as a Cognito user pool.
    b. Amazon Cognito:
        i. A **Cognito User Pool** must be created and configured.
        ii. You should have **User Groups** defined within this user pool, as they will be used for role-based access control.
        iii. Refer to Appendix for steps for creating a Cognito user pool.
    c. Amazon Bedrock:
        i. Ensure **Amazon Bedrock** is enabled for your account.

ii. You must have access to the foundation model for inference.

d. IAM Role:

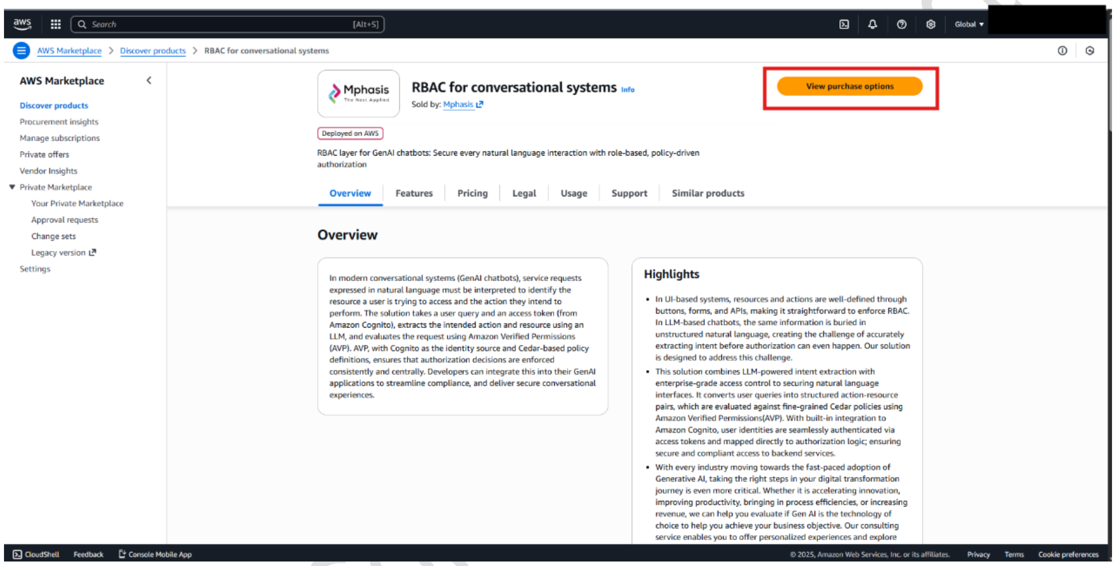
i. An IAM Role must be created which has access to these services and the appropriate role must be attached to the subscriber/user.

How to subscribe:

a. Navigate to AWS marketplace AWS Marketplace: Homepage

b. Search for RBAC for conversational systems

c. Click on View Purchase Options



RBAC for conversational systems Info

Sold by: Mphasis

RBAC layer for GenAI chatbots: Secure every natural language interaction with role-based, policy-driven authorization

Overview

In modern conversational systems (GenAI chatbots), service requests expressed in natural language must be interpreted to identify the resource a user is trying to access and the action they intend to perform. The solution takes a user query and an access token (from Amazon Cognito), extracts the intended action and resource using an LLM, and evaluates the request using Amazon Verified Permissions (AVP). AVP, with Cognito as the identity source and Cedar-based policy definitions, ensures that authorization decisions are enforced consistently and centrally. Developers can integrate this into their GenAI applications to streamline compliance, and deliver secure conversational experiences.

Highlights

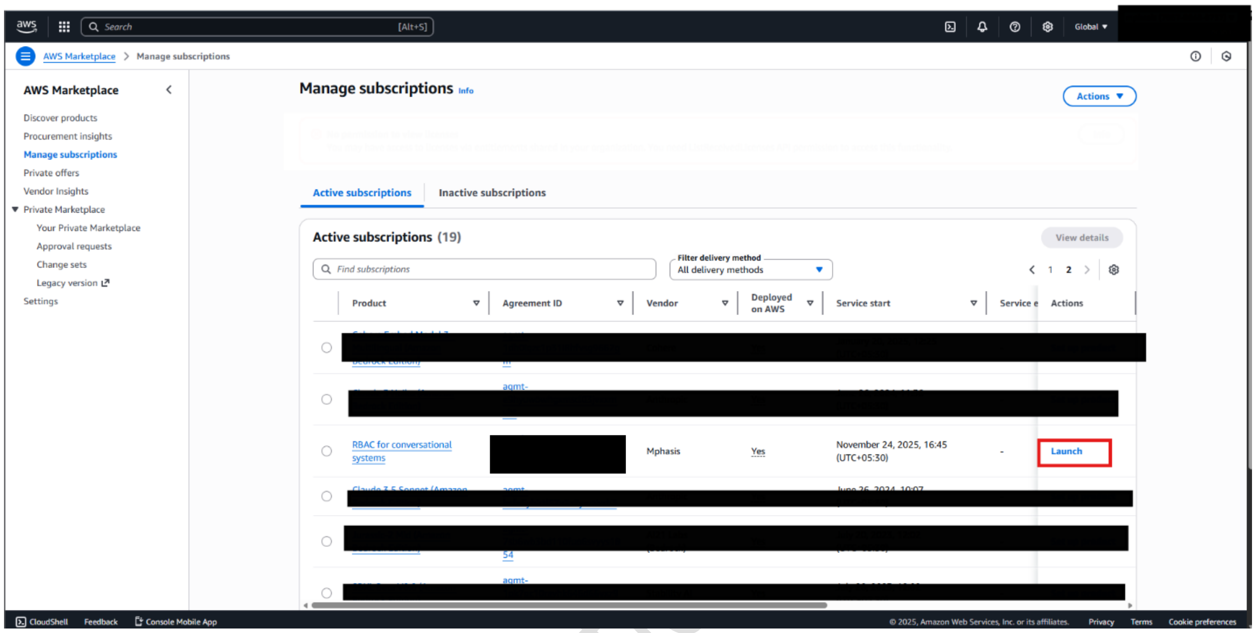
- In UI-based systems, resources and actions are well-defined through buttons, forms, and APIs, making it straightforward to enforce RBAC. In LLM-based chatbots, the same information is buried in unstructured natural language, creating the challenge of accurately extracting intent before authorization can even happen. Our solution is designed to address this challenge.
- This solution combines LLM-powered intent extraction with enterprise-grade access control to securing natural language interfaces. It converts user queries into structured action-resource pairs, which are evaluated against fine-grained Cedar policies using Amazon Verified Permissions(AVP). With built-in integration to Amazon Cognito, user identities are seamlessly authenticated via access tokens and mapped directly to authorization logic; ensuring secure and compliant access to backend services.
- With every industry moving towards the fast-paced adoption of Generative AI, taking the right steps in your digital transformation journey is even more critical. Whether it is accelerating innovation, improving productivity, bringing in process efficiencies, or increasing revenue, we can help you evaluate if Gen AI is the technology of choice to help you achieve your business objective. Our consulting service enables you to offer personalized experiences and explore

© 2025, Amazon Web Services, Inc. or its affiliates.

d. Click on Subscribe.

Steps to Launch the Agent:

1. The purchase will be updated on your Manage Subscriptions page.



AWS Marketplace > Manage subscriptions

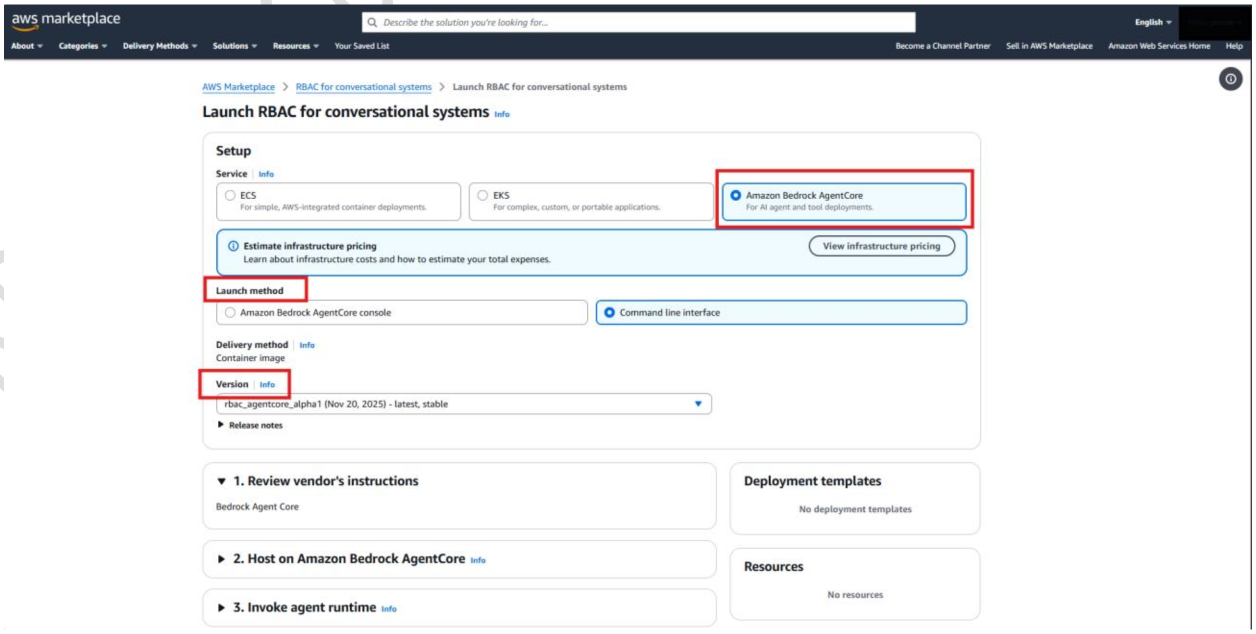
Manage subscriptions Info

Active subscriptions (19)

Filter delivery method: All delivery methods

	Product	Agreement ID	Vendor	Deployed on AWS	Service start	Service e	Actions
○	[redacted]						
○	[redacted]						
○	RBAC for conversational systems	[redacted]	Mphasis	Yes	November 24, 2025, 16:45 (UTC+05:30)	-	Launch
○	[redacted]						
○	[redacted]						
○	[redacted]						

2. Select the product and click Launch to launch the service.
3. Choose Amazon Bedrock AgentCore as the Service
   a. Select Launch method (CLI preferred)
   b. Select the latest version from Version tab
   c. Follow the instructions and host the service on AgentCore (Step2 in the image below)



AWS Marketplace > RBAC for conversational systems > Launch RBAC for conversational systems

Launch RBAC for conversational systems Info

Setup

Service | Info

- ○ ECS — For simple, AWS-integrated container deployments.
- ○ EKS — For complex, custom, or portable applications.
- ● Amazon Bedrock AgentCore — For AI agent and tool deployments.

Estimate infrastructure pricing — Learn about infrastructure costs and how to estimate your total expenses. [View infrastructure pricing]

Launch method

- ○ Amazon Bedrock AgentCore console
- ● Command line interface

Delivery method | Info
Container image

Version | Info
rbac_agentcore_alpha1 (Nov 20, 2025) - latest, stable

▸ Release notes

▾ 1. Review vendor's instructions
Bedrock Agent Core

▸ 2. Host on Amazon Bedrock AgentCore Info

▸ 3. Invoke agent runtime Info

Deployment templates
No deployment templates

Resources
No resources

4. Please set up these environment variables:

Key	**Description**
POLICYSTOREID	The ID of the Bedrock Agent/AgentCore Policy Store to use for authorization and policy evaluation.
MODEL_ID	The specific Bedrock model identifier your application will invoke (e.g., Claude 3.5 Sonnet).
ANTHROPIC_VERSION	The API version for Anthropic models used through AWS Bedrock.
REGION_NAME	The AWS region where Bedrock resources (model, agent, policy store) are located.

Steps to Setup and Run the Agent:

1. After hosting the Agent, continue setting up the agent with the following commands in your terminal:
   a. Set the following environment variables:
      i. AGENT_RUNTIME_ID="<agent runtime id>"
      ii. REGION="us-east-1"
      iii. DISCOVERY_URL= <server metadata url of client application>
      iv. ALLOWED_CLIENTS='[<your client application **Client ID**>]'
      v. REQUEST_HEADER_ALLOWLIST='["Authorization"]'

   b. Get the current configuration:
      i.
      ```
      CURRENT_CONFIG=$(aws bedrock-agentcore-control get-agent-runtime --agent-runtime-id $AGENT_RUNTIME_ID --region $REGION)
      ```

   c. Extract the values (requires jq for JSON parsing):
      i.
      ```
      ROLE_ARN=$(echo $CURRENT_CONFIG | grep -oP '"roleArn":\s*"\K[^"]+')
      ```
      ii.
      ```
      AGENT_RUNTIME_ARTIFACT=$(echo $CURRENT_CONFIG | grep -oP '"agentRuntimeArtifact":\s*\K\{.*?\}.*?\}')
      ```
      iii.
      ```
      NETWORK_CONFIG=$(echo $CURRENT_CONFIG | grep -oP '"networkConfiguration":\s*\K\{.*?\}')
      ```

d. Now update with the existing values plus your new header configuration:

aws bedrock-agentcore-control update-agent-runtime --agent-runtime-id $AGENT_RUNTIME_ID --region $REGION --role-arn $ROLE_ARN \--agent-runtime-artifact "$AGENT_RUNTIME_ARTIFACT" --network-configuration "$NETWORK_CONFIG" \--authorizer-configuration "{\"customJWTAuthorizer\": {\"discoveryUrl\": \"$DISCOVERY_URL\",\"allowedClients\": $ALLOWED_CLIENTS}}" \--request-header-configuration "{\"requestHeaderAllowlist\": $REQUEST_HEADER_ALLOWLIST}"

2. Invoke the Agent from your terminal:
   a. Users can invoke the agent after the setup using the following command:

```
curl --location --request POST '<bedrock agent url >'\
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <Access Token*>' \
--data-raw '{"user_query": "<user query>"}'
```

**The Access Token must be extracted from the Client Application.*

Appendix:

Amazon Cognito

User Pool Creation and Management

1. **Creating a New User Pool**
   a. Sign in to the **AWS Management Console**.
   b. In the service search bar, enter **"Cognito"** and select **Amazon Cognito**.
   c. Navigate to **User Pools**.
   d. Click **Create user pool**.
   e. In the **Sign-in options** section, choose the identifier(s) you want to allow (e.g., *Email*, *Username*).
   f. Configure **Password policy**, **MFA**, and **Account recovery** as required.
   g. In the **Attributes** section, select the standard or custom attributes needed for user.
   h. In the **App integration** section, configure the initial app client:
      i. Provide a client name.

        ii. Disable “Generate client secret” if the client is a browser-based application.

    i. Review the configuration and click **Create user pool**.

2. **Creating User Groups**
    a. From the Cognito dashboard, select **User Pools** and open the user pool created in Step 1.
    b. In the left navigation panel, select **User groups**.
    c. Click **Create group**.
    d. Provide the following details:
        a. **Group name** (e.g., admins, managers, customers)
        b. **Description** (optional)
        c. **Precedence** (optional; lower values take priority)
    e. Click **Create group**.

---

3. **Adding Users to the User Pool**
    a. Navigate to the **Users** section of the desired user pool.
    b. Click **Create user**.
    c. Specify required user information:
        a. Username
        b. Email address
        c. Temporary password (if not using email invitation)
    d. Choose whether to:
        a. Send an invitation email, or
        b. Require the user to reset their password on first sign-in
    e. Click **Create user**.
    f. Add the user to a Group (Optional)

a. Open the user detail page and choose **Groups**.

b. Click **Add to group**.

c. Select the desired group(s) created in Step 2.

d. Click **Add** to confirm.

Using the Client Application to Access the Service

1. **Login to the Client Application**
   a. Users begin by opening the client application (web or desktop) that has been configured to use Amazon Cognito for authentication.

   b. The application redirects the user to the Cognito-hosted login page.

   c. The user enters their credentials (email/username and password).

   d. If Multi-Factor Authentication (MFA) is enabled, the user provides the required verification code.

   e. Amazon Cognito validates the credentials and performs any required security checks, such as password reset or MFA verification.
   f. If authentication is successful, Cognito authorizes the user.

---

2. **Application Receives Tokens**

   a. Cognito generates an **Authorization Code** and sends it back to the application using the application's **redirect URI**.

   b. The application exchanges this code with Cognito to obtain the following tokens:

      i. **Access Token**

      ii. **ID Token**

      iii. **Refresh Token**

   c. The user is automatically redirected to:
      i. https://<your-redirect-uri>?code=<authorization_code>
   d. The application then retrieves the tokens from Cognito in the background.
   e. The client application displays or exposes the **Access Token** for the user.

f. In some configurations, the user may receive the token directly at the redirect page, or the application may provide a dedicated screen to view or copy the token.

g. The user must copy the **Access Token** exactly as provided.

h. To access our service endpoint, the user must include the Access Token in the **Authorization** header of every API request in the following format

    i. Authorization: Bearer <AccessToken>

Amazon Verified Permissions:

Creating an Amazon Verified Permissions Policy Store

1. **Access the Verified Permissions Console**

    a. Sign in to the **AWS Management Console**.

    b. In the search bar, enter **"Verified Permissions"**.

    c. Select **Verified Permissions** from the search results to open the service dashboard.

---

2. **Create a Policy Store**

    a. On the Verified Permissions dashboard, choose **Create policy store**.

    b. Enter a **name** for the policy store (e.g., banking-policy-store).

    c. Under **Schema configuration**, select one of the following:

        a. **Start with AWS example schema**, or

        b. **Provide your own Cedar schema**

    d. Review the configuration and click **Create**.

    e. A **Policy Store ARN** will be generated. Retain this value for application configuration.

---

3. **Add Schema and Policies**

    a. In the left navigation panel, select **Schema**.

    b. Click **Edit schema** (or **Add schema** if no schema exists).

c. Enter or upload the Cedar schema defining:

    a. Entities

    b. Actions

    c. Relationships and attributes

d. Confirm the schema and click **Save**.

e. Navigate to **Policies** in the left panel.

f. Select **Create policy**.

g. Choose the policy type:

    a. **Static policy** (most common), or

    b. **Template-linked policy** (if using templates).

h. Enter policy details using Cedar syntax, specifying:

    a. Principal

    b. Action

    c. Resource

i. Click **Create policy** to save.

---

4. Configure Cognito as an Identity Source for Verified Permissions

a. In the Verified Permissions console, navigate to **Identity sources** (left panel).

b. Select **Add identity source**.

c. Under identity source type, choose **Amazon Cognito**.

d. Provide the following:

    a. The **Cognito User Pool ID**

    b. The **Client ID** (app client) if required

    c. The **issuer URL** (pre-filled for Cognito pools)

e. Select how Cognito identities will map to your policy store principals:

    a. Typically using the **sub** claim (unique user ID)

    b. Optionally map additional attributes (e.g., email, groups, custom claims)

f. Confirm mappings.

g. Click **Add identity source**.