Agents for Amazon Bedrock

User Guide - Confidential

Agents for Amazon Bedrock: User Guide - Confidential

Table of Contents

What is Agents for Amazon Bedrock?

Agents for Amazon Bedrock is a fully managed capability that enables developers to configure an agent to complete actions based on organization data and user input. Agents orchestrate interactions between foundation models, data sources, software applications, and user conversations, and automatically call APIs to take actions. Developers can easily integrate the agents and accelerate delivery of generative AI applications saving weeks of development effort.

With Agents for Amazon Bedrock, you can automate tasks for your customers. For example, you can create an agent that helps customers process insurance claims or one that helps customers make travel reservations and answer questions related to these tasks. You don't have to worry about provisioning, managing infrastructure, or writing custom code. Agents for Amazon Bedrock manages monitoring, encryption, user permissions, and API invocation.

Agents for Amazon Bedrock can carry out the following tasks:

- Extend foundation models to understand user requests and break down the tasks it needs to perform into smaller steps.
- Collect additional information from a user through natural conversation.
- Take actions to fulfill a customer's request.
- Make API calls to your company systems to carry out actions.
- Augment performance and accuracy by using data sources that you provide to facilitate Retrieval-Augmented Generation (RAG).
- Carry out source attribution.

To take advantage of Agents for Amazon Bedrock, you carry out the following steps:

1. (Optional) Set up a vector database and then create a knowledge base to store your private data in that database. For more information, see Building a knowledge base (p. 6).
2. Create an agent for your use-case, add actions that it can carry out, and attach the knowledge base you created to augment its performance. For more information, see Building an agent (p. 19).
3. Test your agent in the console or through API calls and modify the configurations as necessary. For more information, see Test your agent (p. 27).
4. When you have sufficiently modified your agent and it is ready to be deployed to your application, create an alias to point to a version of your agent. For more information, see Deploying an agent: versioning and aliases (p. 28).
5. Set up your application to make API calls to your agent alias.

Topics

Accessing Agents for Amazon Bedrock

Agents for Amazon Bedrock is available as a limited preview release. See Request access (p. 3) for information about how to request access to Agents for Amazon Bedrock.

Supported regions in Amazon Bedrock

Agents for Amazon Bedrock is available in the following AWS Regions:

- US East (N. Virginia)
- US West (Oregon)

Setting up Agents for Amazon Bedrock

Before you use Agents for Amazon Bedrock for the first time, complete the following tasks.

Setting up tasks

Request access

Agents for Amazon Bedrock is available as a limited preview release. To request access, contact your AWS account manager.

After your account has access, the admin users in your account can use Agents for Amazon Bedrock. You can also configure IAM permissions to grant access to Agents for Amazon Bedrock for non-admin users in your account.

Console access

To access the Agents for Amazon Bedrock console and playground:

1. Log in to your AWS account.
2. Navigate to: https://us-east-1.console.aws.amazon.com/bedrock/home?region=us-east-1#/

   You can also access the Agents for Amazon Bedrock console in the US West (Oregon) region (us-west-2).

Agents for Amazon Bedrock Documentation

You can access the Agents for Amazon Bedrock preview release documentation at Bedrock Agents User Guide.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.
2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

- For your daily administrative tasks, grant administrative access to an administrative user in AWS IAM Identity Center (successor to AWS Single Sign-On).

  For instructions, see Getting started in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Set up the AWS Command Line Interface (AWS CLI)

You don't need the AWS CLI to use Agents for Amazon Bedrock. If you prefer, you can skip this step and set up the AWS CLI later.

To install and configure the AWS CLI

1. Install the AWS CLI. For instructions, see the following topic in the *AWS Command Line Interface User Guide*: Installing or updating the latest version of the AWS Command Line Interface
2. Configure the AWS CLI. For instructions, see the following topic in the *AWS Command Line Interface User Guide*: Configuring the AWS Command Line Interface

Grant programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center (successor to AWS Single Sign-On) in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

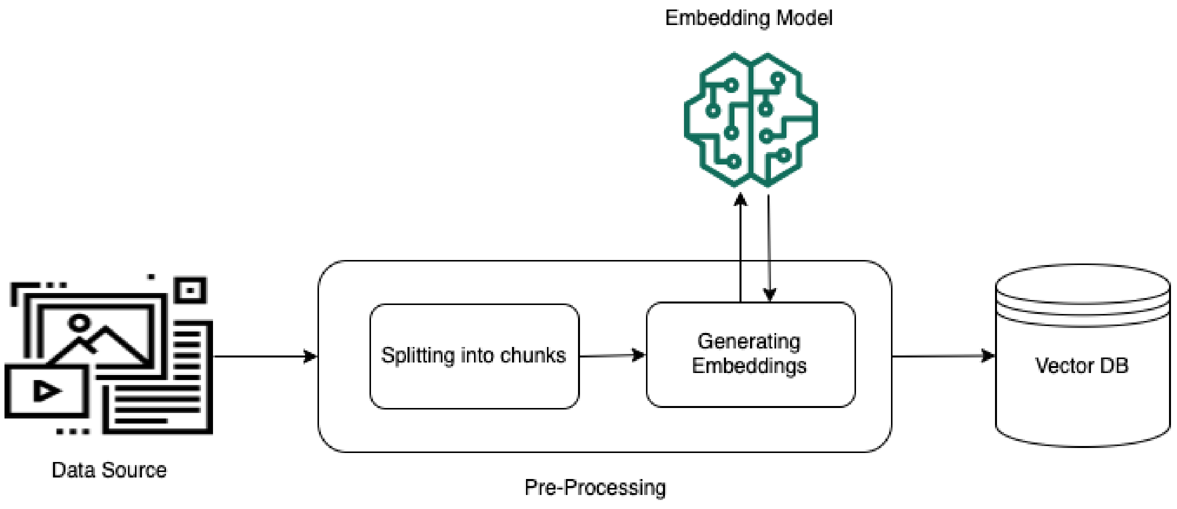
Building a knowledge base

With Agents for Amazon Bedrock, you can enable a Retrieval-Augmented Generation (RAG) workflow by using knowledge bases to build contextual applications by using the reasoning capabilities of LLMs. RAG is a popular technique that combines the use of private data with Large Language Models (LLMs). The combination of Agents for Amazon Bedrock with knowledge bases enables a faster time to market by automating the RAG solution and reducing the build time for your agent. Adding a knowledge base also increases cost-effectiveness by removing the need to continually train your model to be able to leverage your private data.

RAG starts with an initial step to retrieve relevant documents from a data store (most commonly a vector index) based on the user's query. It then employs a language model to generate a response by considering both the retrieved documents and the original query. The following steps in the setup and implementation of RAG are automated for you by the knowledge base service.

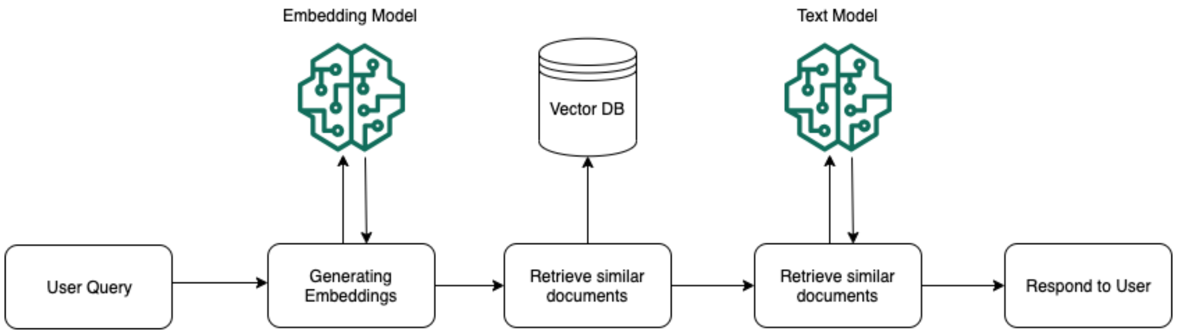
Pre-processing data

To enable effective retrieval from private data, a common practice is to first split the documents into manageable chunks. The following step is to convert the chunks to vectors and then to write them to a vector index while maintaining a mapping to the original document, in order to generate the embeddings. The following image illustrates pre-processing of data for the vector database.



Runtime execution

At runtime, an embedding model is used to convert the user's query to a vector. The vector index is then queried to find documents similar to the user's query by comparing document vectors to the user query vector. In the final step, semantically similar documents retrieved from the vector index are added as context for the original user query. When generating a response for the user, the semantically similar documents are prompted in the text model. The following image illustrates how RAG operates at runtime to augment responses to user queries.



You can build a RAG-based application in Agents for Amazon Bedrock by creating a knowledge base and associating it to an agent to augment its generative capabilities with your own data. You use a knowledge base to load your private data into a vector index. A knowledge base reads data from your Amazon S3 bucket, splits it into smaller chunks, generates vector embeddings, and stores the embeddings in a vector index that you provide. You can associate a knowledge base with multiple agents. After you configure an agent with a knowledge base, it can use the information stored in the corresponding vector index to augment its responses to user queries.

Topics

Create a service role and configure IAM permissions

Knowledge bases use service roles to access AWS resources (for more information, see Creating a role to delegate permissions to an AWS service).

Before you can create a knowledge base, you need to create a service role and attach a trust policy for the role you create.

To create the service role and attach a trust policy

1. Create an IAM role with the prefix `AmazonBedrockExecutionRoleForKnowledgeBase_`. For more information about creating a role, see Creating a role to delegate permissions to an AWS service.
2. Create a trust policy for the role you create. The following shows an example policy you can use. You can restrict the scope of the permission by using one or more global condition context keys. For more information, see AWS global condition context keys. Set the `aws:SourceAccount` value to your account ID. You can use the `ArnEquals` or `ArnLike` condition to restrict the scope to specific knowledge bases.

   Note
   As a best practice for security purposes, replace the * with specific knowledge base IDs after you have created them.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "bedrock.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "account-id"
            },
            "ArnLike": {
                "AWS:SourceArn": "arn:aws:bedrock:region:account-id:knowledge-base/*"
            }
        }
    }]
}
```

3. Attach the trust policy to the role.

For a knowledge base, you need to grant Agents for Amazon Bedrock the following permissions:

- Access to Amazon Bedrock embedding models
- Access to the Amazon S3 object containing your data sources
- (If you encrypted your Amazon S3 data) Permissions to decrypt and encrypt your customer-managed AWS KMS key for your data sources
- (If you create a vector database in Amazon OpenSearch Service) Access to your OpenSearch Service collection
- (If you create a vector database in Pinecone or Redis Enterprise Cloud) Permissions for AWS Secrets Manager to authenticate your Pinecone or Redis Enterprise Cloud account

Topics

Permissions to access Amazon Bedrock embedding models

To allow Agents for Amazon Bedrock to access Amazon Bedrock embedding models to embed your source data, attach the following policy to your Agents for Amazon Bedrock service role. Replace *`region`* with your region your foundation model is in and *`foundation-model-id`* with the ID of the foundation model. Currently, only *`anthropic.claude-v1`* is supported.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels",
                "bedrock:ListCustomModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:region::foundation-model/foundation-model-id"
            ]
        }
    ]
}
```

Permissions to access your data sources in Amazon S3

To allow Agents for Amazon Bedrock to access your Amazon S3 data, attach the following policy attached to your Agents for Amazon Bedrock service role. Replace *`bucket/path/to/folder`* with the path to the object containing all the data source files for your knowledge base and *`account-id`* with the account that the Amazon S3 object belongs to.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::bucket/path/to/folder",
            "arn:aws:s3:::bucket/path/to/folder/*"
        ],
        "Condition": {
            "StringEquals": {
                "aws:PrincipalAccount": "account-id"
            }
        }
    }]
}
```

(Optional) Permissions to decrypt your AWS KMS key for your data sources in Amazon S3

If you encrypted your data sources in Amazon S3 with a AWS KMS key, attach the following policy to your Agents for Amazon Bedrock service role to allow Agents for Amazon Bedrock to decrypt your key. Replace *`region`* and *`account-id`* with the region and account ID to which the key belongs. Replace *`key-id`* with the ID of your AWS KMS key.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "KMS:Decrypt",
        ],
        "Resource": [
            "arn:aws:kms:region:account-id:key/key-id"
        ],
        "Condition": {
            "StringEquals": {
                "kms:ViaService": [
                    "s3.region.amazonaws.com"
                ]
            }
        }
    }]
}
```

(Optional) Permissions to access your vector database in Amazon OpenSearch Service

If you created a vector database in Amazon OpenSearch Service for your knowledge base, attach the following policy to your Agents for Amazon Bedrock service role to allow access to the collection. Replace *region* and *account-id* with the region and account ID to which the database belongs. Input the ID of your Amazon OpenSearch Service collection in *collection-id*. You can allow access to multiple collections by adding them to the `Resources` list.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "aoss:APIAccessAll"
        ],
        "Resource": [
            "arn:aws:aoss:region:account-id:collection/collection-id"
        ]
    }]
}
```

(Optional) Permissions to access your vector database in Pinecone or Redis Enterprise Cloud

If you created a vector database in Pinecone or Redis Enterprise Cloud for your knowledge base, attach the following policy to your Agents for Amazon Bedrock service role to allow AWS Secrets Manager to authenticate your account to access the database. Replace *region* and *account-id* with the region and account ID to which the database belongs. Replace *secret-id* with the ID of your secret.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue"
        ],
        "Resource": [
            "arn:aws:secretsmanager:region:account-id:secret:secret-id"
        ]
    }]
}
```

If your secret is encrypted with a AWS KMS key, attach the following policy to your Agents for Amazon Bedrock service role to allow it to decrypt your key. Replace *region* and *account-id* with the region and account ID to which the key belongs. Replace *key-id* with the ID of your AWS KMS key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/keyId"
            ]
        }
    ]
}
```

(Optional) Permissions for AWS to manage a AWS KMS key for transient data storage during data ingestion

To allow the creation of a AWS KMS key for transient data storage in the process of ingesting your data source, attach the following policy to your Agents for Amazon Bedrock service role. Replace the *region*, *account-id*, and *key-id* with the appropriate values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "kms:GenerateDataKey",
              "kms:Decrypt"
          ],
          "Resource": [
              "arn:aws:kms:region:account-id:key/key-id"
          ]
        }
    ]
}
```

Set up your data for ingestion

Before you can create a knowledge base, you need to carry out the following actions:

1. Configure your source data and upload them to an Amazon S3 bucket.
2. Create a vector database and configure fields for Amazon Bedrock to write data to and to access the index.

Configure your source data

By default, knowledge base automatically splits your source data into chunks, such that each chunk contains, at most, NUMBER tokens. If a document contains less than NUMBER tokens, then it is not split any further. To split your documents into chunks containing less than NUMBER tokens, pre-process the documents by splitting them into smaller chunks.

After you pre-process your source data, upload it to an Amazon S3 bucket. To set up an Amazon S3 bucket, see Getting started with Amazon S3. Knowledge base supports the following file formats:

- Plain text (.txt)
- Markdown (.md)
- HyperText Markup Language (.html)
- Microsoft Word document (.doc/.docx)
- Comma-separated values (.csv)
- Microsoft Excel spreadsheet (.xls/.xlsx)
- Portable Document Format (.pdf)

Create a vector database

Create a vector index in one of the following supported options.

- Amazon OpenSearch Service
- Pinecone
- Redis Enterprise Cloud

After processing your data, Amazon Bedrock writes the following information to the index you created.

- Text extracted from your documents.
- The vectors, corresponding to your text, that were generated by the embeddings model.
- The Amazon S3 path of the source file where the text was extracted from.

Topics

Create a vector index in Amazon OpenSearch Service

1. Log into Amazon OpenSearch Service and create a collection. Take note of the **Collection ARN**, which you will fill out when you create a knowledge base.
2. Once the collection is created, select it and create a vector index.

For detailed documentation on setting up a vector index in Amazon OpenSearch Service, see Working with vector search collections.

While you set up the vector index, take note of the **Collection ARN**, which you will fill out when you create a knowledge base.

There are additional configurations that you must provide when creating a vector index:

- **Vector index name** – The name of the vector index. Choose any valid name of your choice. Later, when you create your knowledge base, enter the name you choose in the **Vector index name** field.

- **Vector field** – The field where the vector embeddings will be stored. Choose any valid name of your choice. Later, when you create your knowledge base, enter the name you choose in the **Vector field** field.
- **Dimensions** – The number of dimensions in the vector. Choose 1536 if you use the Titan Embeddings Model. Later, when you create your knowledge base, enter this number for the **Dimensions** field.
- **Distance metric** – The metric used to measure the similarity between vectors.. We recommend that you experiment with different metrics for your use-case. If you use the Titan Embeddings Model, you can start with **cosine similarity**.
- **Metadata management** – Expand this field and configure the vector index to store additional metadata that a knowledge base can retrieve with vectors. The fields you need to configure are as follows:
  - **Text field** – Amazon Bedrock chunks the raw text in your data and stores the chunks in this field.
    - **Mapping field** – The field where the text will be stored. Choose any valid name of your choice. Later, when you create your knowledge base, enter the name you choose for the **Text field name** field.
    - **Data type** – Select `String`.
    - **Filterable** – Select `False`.
  - **Bedrock-managed metadata field** – Amazon Bedrock stores metadata related to the data in this field. The metadata includes the following:
    - **Mapping field** – Choose any valid name of your choice. Later, when you create your knowledge base, enter the name you choose for the **Bedrock-managed metadata field name** field.
    - **Data type** – Select `String`.
    - **Filterable** – Select `False`.

Security configurations

After you create the knowledge base, you must return to Amazon OpenSearch Service and set up security configurations in your vector database by adjusting the **Network access** and **Data access** settings of your collection. For more information, see Create a knowledge base (p. 15).

Create a vector index in Pinecone

Note
If you use Pinecone, you agree to authorize AWS to access the designated third-party source on your behalf in order to provide the Agents for Amazon Bedrock service to you. You're responsible for complying with any third-party terms applicable to use and and transfer of data from the third-party service.

For detailed documentation on setting up a vector index in Pinecone, see Manage indexes.

While you set up the vector index, take note of the following information, which you will fill out when you create a knowledge base:

- The endpoint URL for your index management page.
- (Optional) The namespace to be used to write new data to your database. For more information, see Using namespaces.

There are additional configurations that you must provide when creating a Pinecone index:

- **Name** – The name of the vector index. Choose any valid name of your choice. Later, when you create your knowledge base, enter the name you choose in the **Vector index name** field.
- **Dimensions** – The number of dimensions in the vector. Choose 1536 if you use the Titan Embeddings Model. Later, when you create your knowledge base, enter this number in the **Dimensions** field.

- **Distance metric** – The metric used to measure the similarity between vectors. We recommend that you experiment with different metrics for your use-case. If you use the Titan Embeddings Model, you can start with **cosine similarity**.

Configure the Secrets Manager

To access your Pinecone index, you must provide your Pinecone API key to Amazon Bedrock through the AWS Secrets Manager.

To set up a secret for your Pinecone configuration

1. Follow the steps at Create an AWS Secrets Manager secret, setting the key as `apiKey` and the value as the API key to access your Pinecone index.
2. To find your API key, open your Pinecone console and select **API Keys**.
3. After you create the secret, take note of its ARN. Later, when you create your knowledge base, enter the ARN in the **Credentials secret ARN** field.

Create a vector index in Redis Enterprise Cloud

Note
If you use Redis Enterprise Cloud, you agree to authorize AWS to access the designated third-party source on your behalf in order to provide the Agents for Amazon Bedrock service to you. You're responsible for complying with any third-party terms applicable to use and and transfer of data from the third-party service.

For detailed documentation on setting up a vector index in Redis Enterprise Cloud, see Integrating Redis Enterprise Cloud with Amazon Bedrock.

While you set up the vector index, take note of the following information, which you will fill out when you create a knowledge base:

- The public endpoint URL for your database.
- The name of the vector index for your database.

Configure the Secrets Manager

To access your Redis Enterprise Cloud cluster, you must provide your Redis Enterprise Cloud security configuration to Amazon Bedrock through the AWS Secrets Manager.

To set up a secret for your Redis Enterprise Cloud configuration

1. Enable TLS to use your database with Amazon Bedrock by following the steps at Transport Layer Security (TLS).
2. Follow the steps at Create an AWS Secrets Manager secret. Set up the following keys with the appropriate values from your Redis Enterprise Cloud configuration in the secret:
   - `username` – The username to access your Redis Enterprise Cloud database. To find your username, look under the **Security** section of your database in the Redis Console.
   - `password` – The password to access your Redis Enterprise Cloud database. To find your password, look under the **Security** section of your database in the Redis Console.
   - `serverCertificate` – The content of the certificate from the Redis Cloud Certificate authority. Download the server certificate from the Redis Admin Console by following the steps at Download certificates.

- `clientPrivateKey` – The private key of the certificate from the Redis Cloud Certificate authority. Download the server certificate from the Redis Admin Console by following the steps at Download certificates.
- `clientCertificate` – The public key of the certificate from the Redis Cloud Certificate authority. Download the server certificate from the Redis Admin Console by following the steps at Download certificates.

3. After you create the secret, take note of its ARN. Later, when you create your knowledge base, enter the ARN in the **Credentials secret ARN** field.

Create a knowledge base

To create a knowledge base

1. Open the Agents for Amazon Bedrock console at https://console.aws.amazon.com/bedrock/.
2. From the left navigation pane, select **Knowledge base**.
3. In the **Knowledge base** section, select **Create knowledge base**.
4. On the **Provide knowledge base details** page, carry out the following actions:
   a. In the **Knowledge base details** section, enter a name for the knowledge base and provide an optional description for it.
   b. In the **IAM permissions** section, choose to let Amazon Bedrock create a new service role or to use an existing service role that allows Amazon Bedrock to access other services on your behalf. For more information, see Create a service role and configure IAM permissions (p. 7).
   c. If you want to attach any tags to the knowledge base, select **Add new tag** In the **tags** section and add the tags as key-value pairs.
   d. Select **Next**.
5. On the **Set up data source** page, you provide the information for the data source to add to the knowledge base by carrying out the following actions:
   a. In the **Data source** section, carry out the following actions:
      i. Provide a name for the data source and the URI of the Amazon S3 object.
      ii. If you encrypted your Amazon S3 data, provide the AWS KMS key in the **Customer-managed AWS KMS key for Amazon S3 data** to allow Amazon Bedrock to decrypt it.
      iii. While converting your data into embeddings, Amazon Bedrock encrypts your transient data with a key that AWS owns and manages, by default. You can select the checkbox labeled **Customize encryption settings (advanced)** under **AWS KMS key for transient data storage**.
   b. In the **Embeddings model** section, choose an embeddings model to convert the knowledge base from your data into an embedding. Currently, only the Amazon Bedrock Titan embeddings model is available.
   c. In the **Vector database** section, select the service that contains a vector database that you have already created. Check that your database is already configured with the required fields (for more information, see Set up your data for ingestion (p. 11)). Fill in the fields to allow Amazon Bedrock to map information from the knowledge base to your database, so that it can store, update, and manage embeddings.

      Note
      If you use a database in Amazon OpenSearch Service, you need to have configured the fields under **Metadata field mapping** beforehand. If you use a database in Pinecone or Redis Enterprise Cloud, you can provide names for these fields here and Amazon Bedrock will dynamically create them in the vector index for you.

   d. Select **Next**.

6. On the **Review and create** page, check the configuration and details of your knowledge base. Select **Edit** in any section that you need to modify. When you are satisfied, select **Create knowledge base**.
7. The knowledge base creation process begins and the **Status** of the source becomes **In progress**. The time it takes to create the knowledge base depends on the amount of data you provided. When the knowledge base is finished being created, a green success banner appears and the **Status** of the knowledge base changes to **Ready**.

   Note

   If you chose to store your embeddings in an Amazon OpenSearch Service vector database, remember to set up your security configurations in OpenSearch Service for the knowledge base after it has been created. For more information, see Security configurations (p. 13).

Set up security configurations for your newly created knowledge base. Follow the steps in the tab corresponding to the database that you set up.

OpenSearch Service

To create a data access policy

1. In the OpenSearch Service console, navigate to your collection.
2. Select **Manage data access**.
3. Select **Create access policy** and give the policy a name and an optional description.
4. Choose **JSON** as the policy definition method and paste the following JSON object into the editor, replacing *collection-name* with the name of your collection and *service-role-arn* with the role ARN that you passed when creating your knowledge base.

```
{[
    {
        "Rules": [
          {
            "Resource": [
              "index/collection_name/*"
            ],
            "Permission": [
              "aoss:DescribeIndex",
              "aoss:ReadDocument",
              "aoss:WriteDocument"
            ],
            "ResourceType": "index"
          }
        ],
        "Principal": [
          "service-role-arn"
        ],
        "Description": "Data access policy"
    }
]
```

5. Select **Create** to create the policy.

Pinecone or Redis Enterprise Cloud

To integrate the third-party knowledge base, attach the following policy to your Agents for Amazon Bedrock service role, replacing *knowledge-base-arn*.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "bedrock:AssociateThirdPartyKnowledgeBase"
        ],
        "Resource": [
            "knowledge-base-arn"
        ],
        "Condition": {
            "StringEquals": {
                "bedrock:ThirdPartyKnowledgeBaseCredentialsSecretArn": "secret-arn"
            }
        }
    }]
}
```

Manage a knowledge base

To manage a knowledge base

1. Open the Agents for Amazon Bedrock console at https://console.aws.amazon.com/bedrock/.
2. From the left navigation pane, select **Knowledge base**.
3. To view details for a knowledge base, either select the **Name** of the source or choose the radio button next to the source and select **Edit**.
4. On the details page, you can carry out the following actions:

   - To change the details of the knowledge base, select **Edit** in the **Knowledge base overview** section.
   - To update the tags attached to the knowledge base, select **Manage tags** in the **Tags** section.
   - If you update the data source from which the knowledge base was created and need to sync the changes, select **Sync** in the **Data source** section.
   - To view the details of a data source, select a **Data source name**. Within the details, you can choose the radio button next to a sync event in the **Sync history** section and select **View warnings** to see why files in the data ingestion job failed to sync.
   - To manage the embeddings model used for the knowledge base, select **Edit provisioned throughput**.
   - Select **Save changes** when you are finished editing.

To delete a knowledge base

1. Open the Agents for Amazon Bedrock console at https://console.aws.amazon.com/bedrock/.
2. From the left navigation pane, select **Knowledge base**.
3. To delete a source, either choose the radio button next to the source and select **Delete** or choose the **Name** of the source and then select **Delete** in the top right corner of the details page.
4. Review the warnings for deleting a knowledge base. If you accept these conditions, enter **`delete`** in the input box and select **Delete** to confirm.
5. Make sure to delete the knowledge base from any agents that it was added to. To do this, carry out the following steps:

   a. From the left navigation pane, select **Agents**.
   b. Choose the **Name** of the agent that you want to delete the knowledge base from.
   c. A red banner appears to warn you to delete the reference to the knowledge base, which no longer exists, from the agent.

d. Select the radio button next to the knowledge base that you want to remove. Select **More** and then choose **Delete**.

Add a knowledge base to an agent

To add a knowledge base to an agent

1. Open the Agents for Amazon Bedrock console at https://console.aws.amazon.com/bedrock/.
2. From the left navigation pane, select **Agents**.
3. Choose the **Name** of the agent that you want to add knowledge bases to.
4. In the **Knowledge base** section, select **Add**.
5. Choose the knowledge base from the dropdown list under **Select knowledge base** and specify the instructions for the agent regarding the knowledge base.

Building an agent

To build an agent, you set up the following components:

- The configuration of the agent itself, which defines the purpose of the agent.
- Action groups that define what actions the agent is designed to carry out.
- (Optional) A knowledge base of data sources to augment the generative capabilities of the agent.

  Note
  If you plan to attach a knowledge base to your agent, first set up your knowledge base by following the steps at Building a knowledge base (p. 6).

You carry out the following steps to set up an agent to interact with your customers:

1. Choose a foundational model that the agent can use for orchestration.
2. Give instructions to the agent, describe the actions it can perform, and provide an API schema for the actions.
3. Add private data to the agent by setting up your data sources in a knowledge base and associating the knowledge base with your agent.
4. Test the agent and iterate on the working draft.
5. After the agent is working as expected, create an alias so that you can integrate the agent with your application.
6. Configure your application to make API calls to the agent alias.
7. Update the working draft of the agent and create new aliases as necessary.

Topics

Create a service role and configure IAM permissions

Before you can create an agent, you need to create a service role and attach a trust policy for the role you create.

To create the service role and attach a trust policy

1. Create an IAM role with the prefix `AmazonBedrockExecutionRoleForAgents_`. For more information about creating a role, see Creating a role to delegate permissions to an AWS service.
2. Create a trust policy for the role you create. The following shows an example policy you can use. You can restrict the scope of the permission by using one or more global condition context keys. For more information, see AWS global condition context keys. Set the `aws:SourceAccount` value to your account ID. You can use the `ArnEquals` or `ArnLike` condition to restrict the scope to specific agents.

   Note
   As a best practice for security purposes, replace the * with specific agent IDs after you have created them.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "bedrock.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "account-id"
            },
            "ArnLike": {
                "AWS:SourceArn": "arn:aws:bedrock:region:account-id:agent/*"
            }
        }
    }]
}
```

3. Attach the trust policy to the role.

Grant Agents for Amazon Bedrock permissions to access the following resources on your behalf through configuring role-based permissions:

- The Amazon Bedrock foundation models
- The Amazon S3 objects containing the OpenAPI schemas for the action groups in your agents

You also need to provide permissions for Agents for Amazon Bedrock to access the AWS Lambda functions for the action groups in your agents through configuring resource-based policies for your Lambda functions.

If you are going to attach a knowledge base to your agent, you also need to provide permissions for Amazon Bedrock to query the knowledge base.

Topics

Permissions to access Amazon Bedrock foundation models

To provide permissions for Agents for Amazon Bedrock to use the Anthropic Claude v1 foundation model for orchestration, attach the following policy to an IAM role.

```
{
    "Version": "2012-10-17",
    "Statement": [{
       "Effect": "Allow",
       "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream"
       ],
    "Resource": [
            "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-v1"
        ]
    }]
}
```

Permissions to access your action group API schemas in Amazon S3

Provide permissions for Agents for Amazon Bedrock to access the Amazon S3 URIs of the API schemas for your agent's action groups. Attach the following policy to an IAM role. In the `Resource` field, you can provide an Amazon S3 object containing the schemas or you can add the URI of each schema to the list.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::path/to/schema"
        ],
        "Condition": {
            "StringEquals": {
                "aws:PrincipalAccount": "account-id"
            }
        }
    }]
}
```

Permissions to access your action group Lambda functions

Provide permissions for Agents for Amazon Bedrock to access the Lambda functions for your agent's action groups. Attach the following resource-based policy to a Lambda function. The `Condition` specifies the ARN of the user account and the ARN of the agent. The `Condition` field is optional, but is recommended for security purposes. For more information, see Using resource-based policies for Lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "statement-name",
        "Effect": "Allow",
        "Principal": {
            "Service": "bedrock.amazonaws.com"
        },
        "Action": [
            "lambda:InvokeFunction",
        ],
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "account-id"
            },
            "ArnLike": {
                "AWS:SourceArn": "arn:aws:bedrock:region:account-id:agent/agent-id"
            }
        }
    }]
}
```

(Optional) Permissions to access your knowledge bases

To provide permissions for Agents for Amazon Bedrock to access a knowledge base you have set up, attach the following policy to an IAM role. Replace *account-id* and *knowledge-base-id* with the appropriate values and *statement-name* with the statement name of your choice.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement-name",
            "Effect": "Allow",
            "Action": [
                "bedrock:QueryKnowledgeBase"
            ],
            "Resource": [
                "arn:aws:bedrock:*:account-id:knowledge-base/knowledge-base-id"
            ]
        }
    ]
}
```

Create an agent

To create an agent, Sign in to the AWS Management Console and open the Amazon Bedrock console at https://console.aws.amazon.com/bedrock/home. Choose **Agents** in the left navigation pane. Then select **Create** at the top right corner of the **Agents** section.

Provide agent details

1. In the **Agent name** section, give the agent a name and an optional description.
2. In the **User input** section, select whether you want the agent to request the user for additional information when trying to complete a task. If you select **No**, the agent doesn't request the user for additional details and informs the user that it doesn't have enough information to complete the task.
3. In the **IAM permissions** section, choose an AWS Identity and Access Management (IAM) role that provides Amazon Bedrock permission to access other AWS services. See Create a service role and configure IAM permissions (p. 19) for more information.
4. In the **Idle session timeout** section, choose the duration that Amazon Bedrock keeps a session with a user open. Amazon Bedrock maintains prompt variables for the duration of the session so that your agent can resume an interaction with the same variables.
5. Select **Next** when you are done setting up the agent configuration.

Select model

1. For the agent to perform orchestration, you need to choose a foundation model. Choose a model provider and then choose a foundation model from the provider in the dropdown menu to train your agent.
2. In **Instructions for the Agent**, provide details to tell the agent what it should do and how it should interact with users. Refer to the following sample instructions:

```
You are an office assistant in an insurance agency. You are friendly and polite. You
 help with managing insurance claims and coordinating pending paperwork.
```

Add Action groups

1. Provide a name for your action group and describe what the action does in the **Description for action groups**. For example, you can use the following text for an action group called `SetupDNSResources`:

```
Use this Action whenever you want to setup DNS resources for setting up a website.
```

2. In **Select Lambda function**, choose a Lambda function that you created in AWS Lambda. The Lambda function provides the business logic that is carried out upon invoking the action. Choose the version of the function to use.

   Note
   Remember to configure a policy to allow the Amazon Bedrock service principal to access the Lambda function. For more information, see Create a service role and configure IAM permissions (p. 19) for more information.. You can optionally provide the ARN of the agent as the `SourceArn` when configuring the policy.

   The following is the general format of the Lambda input event. Use the input event fields to create your Lambda function. For more information, see Event-driven invocation in the AWS Lambda documentation.

```
{
    "messageVersion": "1.0",
    "agent": {
        "name": "string",
        "id": "string",
        "alias": "string",
        "version": "string"
    },
    "inputText": "string",
    "sessionId": "string",
    "actionGroups": [
        {
            "actionGroup": "string",
            "apiPath": "string",
            "httpMethod": "string",
            "parameters": [
                {
                    "name": "string",
                    "type": "string",
                    "value": "string"
                },
                ...
            ],
            "requestBody": {
                "content": {
                    "<content_type>": {
                        "properties": [
                            {
                                "name": "string",
                                "type": "string",
                                "value": "string"
                            },
                            ...
                        ]
                    }
                }
            }
        },
        ...
    ],
    "sessionAttributes": {
        "string": "string",
    }
}
```

Amazon Bedrock expects a response from your Lambda function in the following format.

```
{
    "messageVersion": "1.0",
    "response": [
        {
            "actionGroup": "string",
            "apiPath": "string",
            "httpMethod": "string",
            "httpStatusCode": number,
            "responseBody": {
                "<contentType>": {
                    "body": "string"
                }
            }
        }
    ]
}
```

3. In **Select API schema**, provide a link to the Amazon S3 URI of the schema with the API description, structure, and parameters for the action group. For examples of the schema, see https://github.com/OAI/OpenAPI-Specification/tree/main/examples/v3.0.

   Note
   Remember to assign the IAM role permissions to access the Amazon S3 URI of the schema. For more information, see See Create a service role and configure IAM permissions (p. 19) for more information.

4. Select **Add another Action group** to set up another action group for your agent. When you are done adding action groups, select **Next**.

Add knowledge bases

1. If you have not yet created any knowledge bases, select **Knowledge base** in the left navigation pane and follow the instructions at Create a knowledge base (p. 15) to create one. Otherwise, select a knowledge base from the dropdown menu.
2. Write a prompt in **Knowledge base instructions for the agent** to describe how the agent should use the knowledge base. For example, you can use the following text for a knowledge base called *Domain name system details*:

```
Use this knowledge base whenever you are creating a DNS record
```

3. Select **Add another knowledge base** to set up another knowledge base for your agent. When you are done adding knowledge bases, select **Next**.

Review the configuration of your agent and choose **Edit** for any sections you want to change. Select **Create** when you are ready to create the agent. When the process finishes, a green banner appears at the top to inform you that the agent was successfully created.

Edit your agent

After you create an agent, you can update its configuration as required. The configuration applies to the working draft.

To edit the agent configuration

1. Choose an agent in the **Agents** section.
2. Select **Edit** in the **Agent overview** section.
3. Edit the existing fields as necessary.
4. Select **Save changes**. A green success banner appears at the top.

You might want to try different foundation models for your agent or change the instructions for the agent. These changes apply only to the working draft.

To change the foundation model that your agent uses or the instructions to the agent.

1. Choose an agent in the **Agents** section.
2. Select **View** in the **Working draft** section or choose the working draft.
3. In the **Model details** section, select **Edit**
4. Edit the fields as necessary.
5. Select **Save** to remain in the same window. A green success banner appears at the top.
6. Test the updated agent in the right panel and make changes as necessary.
7. Select **Save and exit** to return to the working draft page.

Manage the action groups of an agent

After creating an agent, you can add more action groups or edit them. Adding and editing take place within the working draft. To carry out these operations, choose an agent from the **Agents** section and then choose the **Working draft** in the **Working Draft** section.

To add an action group

1. Select **Add** in the **Action groups** section.
2. Fill out the action group details
3. Select **Add**. A green success banner appears at the top.

To edit an action group

1. Do one of the following:
   - Choose the radio button next to the action group to edit and select **Edit**.
   - Select an action group to see its details. Choose **Edit** at the top.
2. Edit the existing fields as necessary.
3. Select **Save** to remain in the same window. A green success banner appears if there are no issues and a red banner appears if there are errors in the edit.
4. Test the updated agent in the right panel and make changes as necessary.
5. Select **Save and exit** to return to the working draft page.

Manage the knowledge bases of an agent

After creating an agent, you can add more knowledge bases or edit them. Adding and editing take place within the working draft. To carry out these operations, choose an agent from the **Agents** section and then choose the **Working draft** in the **Working Draft** section.

To add a knowledge base

1. Select **Add** in the **Knowledge bases** section.
2. Choose a knowledge base that you have created and provide instructions for how the agent should interact with it.
3. Select **Add**. A green success banner appears at the top.

To edit a knowledge base

1. Select a knowledge base.
2. In the knowledge base details page, select **Edit**
3. Edit the existing fields as necessary.
4. Select **Save** to remain in the same window. A green success banner appears if there are no issues and a red banner appears if there are errors in the edit.
5. Test the updated agent in the right panel and make changes as necessary.
6. Select **Save and exit** to return to the working draft page.

To delete a knowledge base

1. Select the radio button next to the knowledge base to delete.
2. Select **Delete**.

Test your agent

Once you have created an agent, you can find it in the **Agents** section. When you first create an agent, you will have a *working draft*. The working draft is a version of the agent that you can use to iteratively build the agent. You can interact with the working draft only with the `AgentTestAlias`.

You have access to a test window to interact with your agent. You can use this function to debug your agent and make the necessary changes for it to work as expected.

To test the agent

1. Choose an agent in the **Agents** section.
2. In any of the pages inside an agent, you can select the left arrow icon at the top right to expand the test window.
3. Use the dropdown menu at the top of the test window to choose an alias and associated version to test.

   Note
   If you are inside the working draft, you can only test the working draft.
4. Enter a message to request the agent to perform a task. Use the test window to help debug your agent.

You have the option of turning action groups and knowledge bases on and off. Use this feature to debug your agent and to isolate which actions or knowledge bases need to be updated by assessing its behavior with different settings.

To turn an action group or knowledge base on or off

1. Choose an agent in the **Agents** section.
2. Select the working draft in the **Working draft**.
3. In the **Action groups** or **Knowledge base** section, hover over the **State** of the action group whose state you want to change.
4. An edit button appears. Select it and then choose from the dropdown menu whether the action group or knowledge base is **Enabled** or **Disabled**.
5. If an action group is **Disabled**, the agent doesn't try to carry it out. If a knowledge base is **Disabled**, the agent doesn't use it in orchestration. Turn action groups on and off and use the test window to debug your agent.

Deploying an agent: versioning and aliases

After you have sufficiently iterated on your working draft and are satisfied with the behavior of your agent, you can set it up for deployment and integration into your application by creating *aliases* of your agent.

To deploy your agent, you create an *alias*. During alias creation, Amazon Bedrock automatically creates a version of your agent. The alias points to this newly created version. You can point the alias to previously created version if necessary. You then configure your application to make API calls to that alias.

The version is like a snapshot that preserves the resource as it exists at the time it was created. You can keep modifying the working draft and create new aliases (and consequently, versions) of your agent as necessary. In Amazon Bedrock, you create a new version of your agent by creating an alias that points to the new version by default. Amazon Bedrock creates versions in numerical order, starting from 1. Because a version acts as a snapshot of your agent at the time you created it, it is immutable.

Aliases let you efficiently switch between different versions of your agent without requiring the application to keep track of the version. For example, you can change an alias to point to a previous version of your agent if there are changes that you need to quickly revert.

The working draft version is `DRAFT` and the alias that points to it is the `AgentTestAlias`.

To manage versions and aliases of an agent, select **Agents** from the left navigation pane and choose the agent from the **Agents** section.

To create a new alias (and optionally a new version)

1. Select **Create alias** at the top right corner. Alternatively, select the **Deploy** tab and choose **Create** in the **Aliases** section.
2. Entire a unique name for the alias and provide an optional description.
3. Choose one of the following options
   - Create a new version and to associate the alias with it
   - Associate the alias with an existing version. From the dropdown menu, choose the version you want to associate the alias to.
4. Select **Create alias**. A green success banner appears at the top.

To manage versions of the agent, check that you are in the **Build** tab and that the **Versions** section displays underneath.

To view the details of a version

1. Select the version to view from the **Versions** section.
2. You can't modify any part of a version, but you can view details about the model, action groups, or Lambda function by choosing the name of the information you want to view.

To manage aliases for the agent, select the **Deploy** tab and check that the **Aliases** section displays underneath.

To associate an alias to a different version

1. Select the radio button next to the alias you want to edit.
2. Select the **Edit** button.
3. Choose one of the following options.
   - Create a new version and to associate the alias with it
   - Associate the alias with an existing version. From the dropdown menu, choose the version you want to associate the alias to.

Using the API

The following are the service endpoints for the Agents for Amazon Bedrock service. To connect programmatically to an AWS service, you use an endpoint. For information about endpoints for other AWS services, see AWS service endpoints in the AWS General Reference.

The following table provides a list of Region-specific endpoints that Agents for Amazon Bedrock supports.

Region Name	Region	Endpoint	Protocol
US East (N. Virginia)	us-east-1	invoke-agent-bedrock.us-east-1.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	invoke-agent-bedrock.us-west-2.amazonaws.com	HTTPS

Currently, the only API available for Agents for Amazon Bedrock is `InvokeAgent`.

Invoke your agent

To interact with your agent, send an `InvokeAgent` request. Use `TSTALIASID` as the `agentAliasId` to invoke the draft version of your agent. You must use the Agents for Amazon Bedrock runtime endpoint to call this operation. `PRODUCTION RUNTIME ENDPOINT`

- `agentId` – The unique identifier for your agent. You can find the ID in your agent's details page.
- `agentAliasId` – The unique identifier for your agent's alias. You can find the ID in the details page for your agent's alias. To invoke the draft version of your agent, use `TSTALIASID`.
- `sessionId` – The unique identifier for the session. If you reuse this value, you continue an existing session with the agent if the value you set for the idle session timeout hasn't been exceeded.
- `inputText` – The prompt to provide the agent.

The following shows a sample request.

```
{
    "agentId": "AGENT12345",
    "agentAliasId": "TSTALIASID",
    "sessionId": "session",
    "inputText": "what color are bananas in Timbuktu?"
}
```

The response returns a string responding to the prompt you provided in `inputText`.

Security in Agents for Amazon Bedrock

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Agents for Amazon Bedrock, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Agents for Amazon Bedrock. The following topics show you how to configure Agents for Amazon Bedrock to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Agents for Amazon Bedrock resources.

Topics

Data protection in Agents for Amazon Bedrock

The AWS shared responsibility model applies to data protection in Agents for Amazon Bedrock. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center (successor to AWS Single Sign-On) or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.

- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Agents for Amazon Bedrock or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

Amazon Bedrock uses encryption to protect data at rest and data in transit.

Encryption at rest

To protect your data, Amazon Bedrock encrypts your prompts and their responses. Amazon Bedrock encrypts the data using AWS-owned keys.

Encryption in transit

Within AWS, all internetwork data in transit supports TLS 1.2 encryption.

Requests to the Amazon Bedrock API and console are made over a secure (SSL) connection. You pass AWS Identity and Access Management roles to Amazon Bedrock to provide permissions to access resources on your behalf for training and deployment.

Identity and access management for Agents for Amazon Bedrock

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Agents for Amazon Bedrock resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Agents for Amazon Bedrock.

Service user – If you use the Agents for Amazon Bedrock service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Agents for Amazon Bedrock features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Agents for Amazon Bedrock, see Troubleshooting Agents for Amazon Bedrock identity and access (p. 45).

Service administrator – If you're in charge of Agents for Amazon Bedrock resources at your company, you probably have full access to Agents for Amazon Bedrock. It's your job to determine which Agents for Amazon Bedrock features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Agents for Amazon Bedrock, see How Agents for Amazon Bedrock works with IAM (p. 37).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Agents for Amazon Bedrock. To view example Agents for Amazon Bedrock identity-based policies that you can use in IAM, see Identity-based policy examples for Agents for Amazon Bedrock (p. 41).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *AWS Account Management Reference Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center (successor to AWS Single Sign-On). You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Agents for Amazon Bedrock in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.

Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Agents for Amazon Bedrock works with IAM

Before you use IAM to manage access to Agents for Amazon Bedrock, learn what IAM features are available to use with Agents for Amazon Bedrock.

IAM features you can use with Agents for Amazon Bedrock

IAM feature	Agents for Amazon Bedrock support
Identity-based policies (p. 37)	Yes
Resource-based policies (p. 38)	No
Policy actions (p. 38)	Yes
Policy resources (p. 39)	Yes
Policy condition keys (p. 39)	No
ACLs (p. 40)	No
ABAC (tags in policies) (p. 40)	No
Temporary credentials (p. 40)	Yes
Principal permissions (p. 41)	Yes
Service roles (p. 41)	Yes
Service-linked roles (p. 41)	No

To get a high-level view of how Agents for Amazon Bedrock and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Agents for Amazon Bedrock

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Agents for Amazon Bedrock

To view examples of Agents for Amazon Bedrock identity-based policies, see Identity-based policy examples for Agents for Amazon Bedrock (p. 41).

Resource-based policies within Agents for Amazon Bedrock

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for Agents for Amazon Bedrock

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Agents for Amazon Bedrock actions, see Actions Defined by Agents for Amazon Bedrock in the *Service Authorization Reference*.

Policy actions in Agents for Amazon Bedrock use the following prefix before the action:

```
bedrock
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "bedrock:action1",
      "bedrock:action2"
         ]
```

To view examples of Agents for Amazon Bedrock identity-based policies, see Identity-based policy examples for Agents for Amazon Bedrock (p. 41).

Policy resources for Agents for Amazon Bedrock

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To view examples of Agents for Amazon Bedrock identity-based policies, see Identity-based policy examples for Agents for Amazon Bedrock (p. 41).

Policy condition keys for Agents for Amazon Bedrock

Supports service-specific policy condition keys	No

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

All Amazon Bedrock actions support condition keys using Bedrock models as the resource.

To view examples of Agents for Amazon Bedrock identity-based policies, see Identity-based policy examples for Agents for Amazon Bedrock (p. 41).

ACLs in Agents for Amazon Bedrock

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Agents for Amazon Bedrock

Supports ABAC (tags in policies)	No

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Agents for Amazon Bedrock

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Agents for Amazon Bedrock

Supports principal permissions	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Agents for Amazon Bedrock in the *Service Authorization Reference*.

Service roles for Agents for Amazon Bedrock

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
> Changing the permissions for a service role might break Agents for Amazon Bedrock functionality. Edit service roles only when Agents for Amazon Bedrock provides guidance to do so.

When you create an agent resource, you must create or choose a role to allow Agents for Amazon Bedrock to access resources on your behalf. If you have previously created a service role or service-linked role, Agents for Amazon Bedrock provides you with a list of roles to choose from. It's important to choose a role that allows access to the required actions. For more information, see Create a service role and configure IAM permissions (p. 19).

Service-linked roles for Agents for Amazon Bedrock

Supports service-linked roles	No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Identity-based policy examples for Agents for Amazon Bedrock

By default, users and roles don't have permission to create or modify Agents for Amazon Bedrock resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Agents for Amazon Bedrock, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for Agents for Amazon Bedrock in the *Service Authorization Reference*.

> **Note**
> The Amazon Bedrock service is available as a limited preview release, so its information is not included in the Service Authorization Reference.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Agents for Amazon Bedrock resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Agents for Amazon Bedrock console

To access the Agents for Amazon Bedrock console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Agents for Amazon Bedrock resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To provide access to the Agents for Amazon Bedrock console, attach the following policy to the roles or entities that need access.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BedrockConsole",
            "Effect": "Allow",
            "Action": [
               "bedrock:CreateAgent",
               "bedrock:UpdateAgent",
               "bedrock:GetAgent",
               "bedrock:ListAgents",
               "bedrock:CreateActionGroup",
               "bedrock:UpdateActionGroup",
               "bedrock:GetActionGroup",
               "bedrock:ListActionGroups",
               "bedrock:CreateAgentDraftSnapshot",
               "bedrock:GetAgentVersion",
               "bedrock:ListAgentVersions",
               "bedrock:CreateAgentAlias",
               "bedrock:UpdateAgentAlias",
               "bedrock:GetAgentAlias,
               "bedrock:ListAgentAliases"
               "bedrock:InvokeAgent"
            ],
            "Resource": "*"
        }
    ]
```

For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow users to perform actions on agent and alias resources

You can provision identities with permissions to perform actions on agent and alias resources. The ARNs of these resources are formatted as follows.

- Agents `arn:aws:bedrock:`*`region`*`:`*`account-id`*`:agent/`*`AGENTID`*
- Aliases `arn:aws:bedrock:`*`region`*`:`*`account-id`*`:agent-alias/`*`AGENTID`*`/`*`ALIASID`*

A role can call API operations on specific resources. For example, the `InvokeAgent` operation can only be used on alias resources and the `UpdateAgent` operation can only be used on agent resources, If you specify an operation in a policy that can't be used on the resource specified in the policy, Amazon Bedrock returns an error. For a list of operations and the resources that they can be used with, see the following table. `CreateAgent` and `ListAgents` are not performed on a specific resource.

Operation	Resource
CreateAgent	N/A
UpdateAgent	Agent
GetAgent	Agent
ListAgents	N/A
CreateActionGroup	Agent
UpdateActionGroup	Agent
GetActionGroup	Agent
ListActionGroups	Agent
CreateAgentDraftSnapshot	Agent
GetAgentVersion	Agent
ListAgentVersions	Agent

Operation	Resource
CreateAgentAlias	Agent
UpdateAgentAlias	Alias
GetAgentAlias	Alias
ListAgentAliases	Agent
InvokeAgent	Alias

The following is a sample policy that you can attach to an IAM role to allow it to call Amazon Bedrock API operations to get information about an agent, update an agent alias, and interact with an agent. Replace the *`sid`* with a policy identifier of your choice, the *`account-id`* with the account ID to which the agent belongs, the `AGENTID` with the ID of the agent, and the `ALIASID` with the alias of the agent. You can find agent and alias IDs with the `ListAgents` API operation or in the agent and alias details pages in the console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "sid",
            "Effect": "Allow",
            "Action": "bedrock:GetAgent",
            "Resource": "arn:aws:bedrock:region:account-id:agent/AGENTID"
        },
        {
            "Sid": "sid",
            "Effect": "Allow",
            "Action": [
                "bedrock:UpdateAgentAlias",
                "bedrock:InvokeAgent"
            ],
            "Resource": [
                "arn:aws:bedrock:region:account-id:agent-alias/AGENTID/ALIASID"
            ]
        },
    ]
}
```

Troubleshooting Agents for Amazon Bedrock identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Agents for Amazon Bedrock and IAM.

Topics

I am not authorized to perform an action in Agents for Amazon Bedrock

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `bedrock:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 bedrock:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `bedrock:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Agents for Amazon Bedrock.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Agents for Amazon Bedrock. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Agents for Amazon Bedrock resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Agents for Amazon Bedrock supports these features, see How Agents for Amazon Bedrock works with IAM (p. 37).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.

- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Compliance validation for Agents for Amazon Bedrock

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

  Note
  Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Agents for Amazon Bedrock

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Agents for Amazon Bedrock offers several features to help support your data resiliency and backup needs.

Infrastructure Security in Agents for Amazon Bedrock

As a managed service, Agents for Amazon Bedrock is protected by the AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well‐Architected Framework*.

You use AWS published API calls to access Agents for Amazon Bedrock through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.