Administrator Guide

# AWS Chatbot

# AWS Chatbot: Administrator Guide

# Table of Contents

# What is AWS Chatbot?

AWS Chatbot is an AWS service that enables DevOps and software development teams to use messaging program chat rooms to monitor and respond to operational events in their AWS Cloud. AWS Chatbot processes AWS service notifications from Amazon Simple Notification Service (Amazon SNS), and forwards them to chat rooms so teams can analyze and act on them immediately, regardless of location.

You can also run AWS CLI commands in chat channels using AWS Chatbot.

**Topics**

## Features of AWS Chatbot

AWS Chatbot enables ChatOps for AWS. *ChatOps* speeds software development and operations by enabling DevOps teams to use chat clients and chatbots to communicate and execute tasks. AWS Chatbot notifies chat users about events in their AWS services, so teams can collaboratively monitor and resolve issues in real time, instead of addressing emails from their SNS topics. AWS Chatbot also allows you to format incident metrics from Amazon CloudWatch as charts for viewing in chat notifications.

Important features of the AWS Chatbot service include the following:

- **Supports Amazon Chime, Microsoft Teams, and Slack** – You can add AWS Chatbot to your Amazon Chime chat rooms, Microsoft Teams channel, or Slack channel in just a few clicks.
- **Predefined AWS Identity and Access Management (IAM) policy templates** – AWS Chatbot provides chat room-specific permission controls through AWS Identity and Access Management (IAM). AWS Chatbot's predefined templates make it easy to select and set up the permissions you want associated with a given channel or chat room.

- **Receive notifications –** Use AWS Chatbot to receive notifications about operational incidents and other events from supported sources, such as operational alarms, security alerts, or budget deviations. To set up notifications in the AWS Chatbot console, you simply choose the channels or chat rooms you want to receive notifications and then choose which Amazon Simple Notification Service (Amazon SNS) topics should trigger notifications.
- **Customize notifications –** You can define and receive customized AWS service and application notifications directly in your chat channels. Custom notifications can be as succint or comprehensive you desire and use the same Amazon SNS-based mechanisms as default notifications.
- **Create custom actions –** Custom actions transform your notifications into actionable items. A custom action appears as a button on your notifications. This button represents a Lambda function or CLI command that you define. You can use custom actions to retrieve telemetry information, run Lambda functions, run an automation runbook, and notify team members. When an issue arises, you can easily take action directly from your notifications.
- **Monitor and manage AWS resources through the AWS CLI with Microsoft Teams and Slack** – AWS Chatbot supports CLI commands for most AWS services, making it easy to monitor and manage your AWS resources from your chat clients on desktop and mobile devices. Your teams can retrieve diagnostic information in real-time, change your AWS resources, run AWS SSM runbooks, and start long running jobs from a centralized location. AWS Chatbot commands use the standard AWS Command Line Interface syntax.
- **Search and discover AWS information –** You can search and discover information about AWS services and your AWS resources by asking AWS Chatbot natural language questions. The answers provided in your chat channels are pulled directly from your AWS environments, AWS product documentation, and support articles. This makes it easier to locate your resources, find product information, and troubleshoot issues.
- **Ask Amazon Q –** You can get Amazon Q, Generative Artifical Intelligence (AI) assistant powered answers to your AWS service questions directly in your chat channels.

# How AWS Chatbot works

AWS Chatbot uses Amazon Simple Notification Service (Amazon SNS) topics to send event and alarm notifications from AWS services to your chat channels. Once an SNS topic is associated with a configured chat client, events and alarms from various services are processed and notifications are delivered to the specified chat channels and webhooks. For Microsoft Teams and Slack, after an administrator approves AWS Chatbot support for the workspace or tenant, anyone in the workspace or team can add AWS Chatbot to their chat channels. For Amazon Chime, users with AWS Identity and Access Management (IAM) permissions to use Amazon Chime can add AWS Chatbot to their webhooks. You use the AWS Chatbot console to configure chat clients to receive notifications from SNS topics.

AWS Chatbot supports a number of AWS services, including Amazon CloudWatch, AWS Billing and Cost Management, and AWS Security Hub. For a complete list of supported services, see Monitoring AWS services.

You can also run AWS CLI commands directly in chat channels using AWS Chatbot. You can retrieve diagnostic information, configure AWS resources, and run workflows. To run a command, AWS Chatbot checks that all required parameters are entered. If any are missing, AWS Chatbot prompts you for the required information. AWS Chatbot then confirms if the command is permissible by checking the command against what is allowed by the configured IAM roles and the channel guardrail policies. For more information, see Running AWS CLI commands from chat channels and Understanding permissions.

# Regions and quotas for AWS Chatbot

AWS Chatbot is a global service and can be used in all commercial AWS Regions. You can combine Amazon SNS topics from multiple Regions in a single AWS Chatbot configuration.

AWS Chatbot currently supports service endpoints, however there are no adjustable quotas. For more information about AWS Chatbot AWS Region availability and quotas, see AWS Chatbot endpoints and quotas. AWS Chatbot supports using all supported AWS services in the Regions where they are available.

# Supported services for AWS Chatbot

AWS Chatbot uses Amazon EventBridge to send most notifications. For a list of supported services, see Events from AWS services in Amazon EventBridge in the *Amazon EventBridge User Guide*.

AWS Chatbot also supports notifications for the following services:

- Amazon CloudWatch
- Amazon CodeCatalyst
- AWS CodeStar
- AWS Cost Anomaly Detection

# AWS Chatbot requirements

To use AWS Chatbot, you need the following:

- An AWS account to associate with Amazon Chime, Microsoft Teams, or Slack chat clients during AWS Chatbot setup.
- Administrative privileges for your Amazon Chime chat room, Microsoft Teams tenant, or Slack workspace. You can be the Slack workspace owner or have the ability to work with workspace owners to get approval for installing AWS Chatbot.
- Familiarity with AWS Identity and Access Management (IAM) and IAM roles and policies. For more information about IAM, see What is IAM? in the *IAM User Guide*.
- Experience with the AWS services supported by AWS Chatbot, including experience configuring those services to subscribe to Amazon Simple Notification Service (Amazon SNS) topics to send notifications. For information about supported services, see Using AWS Chatbot with Other AWS Services.

To access Amazon CloudWatch metrics, AWS Chatbot requires an AWS Identity and Access Management (IAM) role with a permissions policy and a trust policy. You create this IAM role, with the required policies, using the AWS Chatbot console. You can use an existing IAM role, but it must have the required policies.

# Accessing AWS Chatbot

You access and configure AWS Chatbot through the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.

You can also access the AWS Chatbot app from the Slack app directory.

# Getting started with AWS Chatbot

To get started using AWS Chatbot to help manage your AWS infrastructure, follow the steps below to set up AWS Chatbot with chat rooms and Amazon SNS topic subscriptions.

**Topics**

## Setting up AWS Chatbot

To use AWS Chatbot, you authorize a chat client configuration with AWS Chatbot, and optionally configure AWS Chatbot to use an Amazon Simple Notification Service (Amazon SNS) topic to deliver notifications to the chat rooms. Before you can get started, you must complete the following setup tasks.

### Prerequisites

With AWS Chatbot, you can use chat rooms to monitor and respond to events in your AWS Cloud.

Below are some prerequisites you should have before you begin using AWS Chatbot:

- You have started using some AWS services. For more information about AWS services you can use with AWS Chatbot, see Monitoring AWS services using AWS Chatbot.
- You have administrator privileges with your chosen chat client chat room, tenant, or workspace.
- You understand AWS Chatbot's permission schemes. For more information about AWS Chatbot's permission schemes, see Understanding permissions.

If you have an existing AWS administrator user, you can access the AWS Chatbot console with no additional permissions. AWS recommends that you grant only the permissions required to perform a task for other users. For more information, see Apply least-privilege permissions in the *AWS Identity and Access Management 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 a 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 a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and 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 a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.
2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

### Sign in as the user with administrative access

- 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*.

### Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

## Setting up IAM permissions for AWS Chatbot

If you would like to add AWS Chatbot access to an existing user or group, you can choose from allowed Chatbot actions in IAM.

If you need to customize an IAM role to work with AWS Chatbot, you can use the procedure in this topic.

If you want to allow AWS Chatbot to answer questions about your AWS resources, make sure to add the `AWSResourceExplorerReadOnlyAccess` policy to your IAM role. You must also make sure that your channel guardrail policies allow `AWSResourceExplorerReadOnlyAccess` permissions.

**To create a policy to configure AWS Chatbot**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Policies** from the navigation pane.
3. Choose **Create policy**.
4. Expand **Service** and find **Chatbot**.
5. Under **Actions**, expand the **Read** and **Write** sections to see the available actions.

   **Read** actions include **DescribeChimeWebhookConfigurations**, **DescribeSlackChannelConfigurations**, **DescribeTeamsChannelConfiguration**, and more.

   **Write** actions include **CreateChimeWebhookConfiguration**, **DeleteChimeWebhookConfiguration**, and more.
6. After selecting the actions you want to include, choose **Review policy**.
7. Give your policy a name and description, then choose **Create policy**. You can now add your new policy to any of your users or groups.

For more information on updating the permissions of existing users, see Adding Permissions to a User (Console) in the *IAM User Guide*.

**Note**

AWS Chatbot is a global service that requires access to all AWS Regions. If there is a policy in place that prevents access to services in certain Regions, you must change the policy to allow global AWS Chatbot access. For more information about policy types that might limit how IAM roles can be assumed and how to override them, see Other policy types.

# Setting up Amazon SNS topics

To use AWS Chatbot, you must have Amazon SNS topics set up. If you don't have any Amazon SNS topics yet, follow the steps to get started in Getting Started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

> **Note**
>
> AWS Chatbot doesn't support SNS FIFO topics because these topics can't deliver messages to HTTPS endpoints. For more information, see Message delivery for FIFO topics in the *Amazon Simple Notification Service Developer Guide*.

If you have server-side encryption enabled for your Amazon SNS topics, you must give permissions to the sending services in your AWS KMS key policy to post events to the encrypted SNS topics. The following policy is an example for Amazon EventBridge.

```
{
  "Sid": "Allow CWE to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

In order to successfully test the configuration from the console, your role must also have permission to use the AWS KMS key.

AWS managed service keys don’t allow you to modify access policies, so you will need AWS KMS/CMK for encrypted SNS topics. You can then update the access permissions in the AWS KMS key policy to allow the service that sends messages to publish to your encrypted SNS topics (for example, EventBridge).

# Tutorial: Get started with Slack

To get started using AWS Chatbot to help manage your AWS infrastructure, follow the steps below to set up AWS Chatbot with chat rooms and Amazon SNS topic subscriptions.

**Topics**

## Prerequisites

Before you get started, make sure you've completed the tasks in Setting up AWS Chatbot. You will need to choose a permissions scheme in the following procedure. This scheme determines the permissions your channel members will have and what AWS Chatbot can do on your behalf. For more information about AWS Chatbot permissions, see Understanding permissions. You must also create or choose a Slack channel to be used in your AWS Chatbot configuration. This channel is used to monitor and operate your AWS resources.

## Step 1: Configure a Slack client

To allow AWS Chatbot to send notifications or run commands, you must configure AWS Chatbot with Slack. Workspace administrators must approve the use of the AWS Chatbot app in the workspace. Members can request to install apps if app approval is turned on by the workspace administrator. For more information, see Add apps to your Slack workspace.

**To configure a Slack client**

1. Add AWS Chatbot to the Slack workspace:

    a. In Slack, on the left navigation pane, choose **Automations**.

> **Note**
>
> If you do not see **Automations** in the left navigation pane, choose **More**, then choose **Automations**.

b. If AWS Chatbot is not listed, choose the **Browse Apps Directory** button.

c. Browse the directory for the AWS Chatbot app and then choose **Add** to add AWS Chatbot to your workspace.

2. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.

3. Under **Configure a chat client**, choose **Slack**, then choose **Configure**.

> **Note**
>
> After choosing **Configure**, you'll be redirected to Slack's authorization page to request permission for AWS Chatbot to access your information. For more information, see Chat client application permissions for AWS Chatbot.

4. From the dropdown list at the top right, choose the Slack workspace that you want to use with AWS Chatbot.

   There's no limit to the number of workspaces that you can set up for AWS Chatbot, but you can set up only one at a time.

5. Choose **Allow**.

## Step 2: Configure a Slack channel

To allow AWS Chatbot to send notifications or run commands in your Slack channel, you must also configure AWS Chatbot with a Slack channel. Configuring a channel consists of:

- Adding AWS Chatbot to your Slack channel
- Associating a channel with the configuration
- Defining user permissions, which dictate what tasks users can perform in a channel
- (Optional) Adding Amazon SNS topics, which AWS Chatbot uses to send notifications to your channel

**To configure a Slack channel**

1. Add AWS Chatbot to the Slack channel:

   a. In your Slack channel, enter **`invite @aws`**.

   b. Choose **Invite Them**.

2. Associate a channel with your configuration:

   a. On the **Workspace details** page in the AWS Chatbot console, choose **Configure new channel**.

   b. Under **Configuration details**, enter a name for your configuration. The name must be unique across your account and can't be edited later.

   c. If you want to enable logging for this configuration, choose **Publish logs to Amazon CloudWatch Logs**. For more information, see Amazon CloudWatch Logs for AWS Chatbot.

   > **Note**
   >
   > There is an extra charge for using CloudWatch Logs.

   d. For **Slack channel**, choose the channel you used in step 1. AWS Chatbot supports both public and private channels.

   To configure a private channel with AWS Chatbot:

   i. In Slack, copy the Channel ID of the private channel by right-clicking on the channel name in the left pane and choosing **Copy Link**. The Channel ID is the string at the end of the URL (for example, `AB3BBLZZ8YY`).

   ii. In AWS Chatbot, paste the ID into the **Channel URL** field. (If you copy the URL of the private Slack channel, the AWS Chatbot console shows only the Channel ID value when you paste it into the field.)

3. Define user permissions:

   a. Choose your **Role Setting**.

   > **Tip**
   >
   > Your role setting dictates what permissions your channel members have. A channel role gives all members the same permissions. This is useful if your channel

members typically perform the same actions in Slack. A user role requires your channel members to choose their own roles. As such, different users in your channels can have different permissions. This is useful if your channel members are diverse or you don’t want new channel members to perform actions as soon as they join the channel. For more information, see Role setting.

Channel role

1. For **Role setting**, choose **Channel role**.
2. For **Channel role**, choose **Create new role**. If you want to use an existing role instead, choose **Use an existing role**. To use an existing IAM role, you will need to modify it for use with AWS Chatbot. If you want your users to be able to use Amazon Q, attach the AmazonQDeveloperAccess policy. For more information, see Configuring an IAM Role for AWS Chatbot.
3. For **Role name**, enter a name. Valid characters: a-z, A-Z, 0-9, .\w+=,.@-_.
4. For **Policy template**, select **Amazon Q permissions** and any other templates you wish to use.

User roles

1. For **Role setting**, choose **User roles**.

b. Select the policies that will make up your channel guardrails. Your channel guardrails control what actions are available to your channel members. Add the **AmazonQDeveloperAccess policy** as a channel guardrail to allow your users to use Amazon Q in your Slack channel.

4. (Optional) Add Amazon SNS topics:

> **Note**
>
> If you want to receive notifications in your Slack channel, complete these steps.

- Choose your notification settings:

   i. For **SNS Region**, choose the AWS Region that hosts the SNS topics for this AWS Chatbot subscription.

   ii. For **SNS topic**, choose the Amazon SNS topic for the client subscription. This topic determines the content that's sent to the Slack channel. If the region has additional SNS topics, you can choose them from the same dropdown list. The SNS topics you choose must be configured in the services for which you want to receive notifications. For more information, see Monitoring AWS services using AWS Chatbot.

   iii. To add an Amazon SNS topic from another AWS Region to the notification subscription, choose **Add another Region**.

   > **Note**
   >
   > For a tutorial on subscribing existing Amazon SNS topics to AWS Chatbot, see Tutorial: Subscribing an Amazon SNS topic to AWS Chatbot.

   Notifications from supported services that publish to the chosen Amazon SNS topics will now appear in the Slack channel.

5. Choose **Save**.

> **Note**
>
> You can configure a Slack channel to run commands to your AWS account. For more information, see Running AWS CLI commands from chat channels.

You can configure as many channels with as many topics as you need.

If you want to allow AWS Chatbot to answer questions about your AWS resources, turn on AWS Resource Explorer in the Resource Explorer Console. For more information, see Getting started with Resource Explorer in the *AWS Resource Explorer User Guide*.

## Step 3: Test notifications from AWS services to Slack

To verify that an Amazon Simple Notification Service (Amazon SNS) topic sends notifications to your Slack channel, you can test your setup by sending a notification. Ensure your AWS Chatbot configuration is subscribed to at least one Amazon SNS topic and that your topics are assigned to a service supported by AWS Chatbot. For a list of supported services, see Monitoring AWS services using AWS Chatbot. You can also test notifications by using CloudWatch. For more information, see Test notifications from AWS services to Amazon Chime or Slack using CloudWatch.

**Testing notifications with configured clients**

1. Open the AWS Chatbot console.
2. Choose the configured client you want to test.
3. In the configured client, choose the channel to send a test notification to.
4. Choose **Send test message**.
5. View the confirmation message at the top of the screen that shows a message was sent to your Amazon SNS topic.
6. Confirm the test message in your Slack channel.

## Configuring Slack channels using AWS CloudFormation

You can automate Slack channel configuration by using an AWS CloudFormation template. To use an AWS CloudFormation template, you need the **Workspace ID** found under **Workspace details** in the AWS Chatbot console. For more information, see AWS::Chatbot::SlackChannelConfiguration in the *AWS CloudFormation User Guide*.

## Next steps

After you configure your chat clients and test that your notifications are working, you might want to explore some of the following topics:

- Learn about which other AWS services you can integrate with AWS Chatbot in Monitoring AWS services using AWS Chatbot.
- Learn about what you can customize using AWS Chatbot in Customizing AWS Chatbot.
- Learn about what actions you can perform using AWS Chatbot in Performing actions.
- Learn what questions you can ask AWS Chatbot and Amazon Q in Asking questions from chat channels in AWS Chatbot.

# Tutorial: Get started with Amazon Chime

To get started using AWS Chatbot to help manage your AWS infrastructure, follow the steps below to set up AWS Chatbot with chat rooms and Amazon SNS topic subscriptions.

**Topics**

## Prerequisites

Before you get started, make sure you've completed the tasks in Setting up AWS Chatbot. You will need to choose a permissions scheme in the following procedure. This scheme determines the permissions your channel members will have and what AWS Chatbot can do on your behalf. For more information about AWS Chatbot permissions, see Understanding permissions.

## Step 1: Setting up AWS Chatbot with Amazon Chime

To set up AWS Chatbot for Amazon Chime, get the webhook URL for your team's chat room from Amazon Chime.

**Prerequisite**

You must be an Amazon Chime chat room admin and have the ability to manage webhooks.

**To configure an Amazon Chime client**

1. Open Amazon Chime.
2. For **Amazon Chime**, choose the chat room that you want to set up to receive notifications through AWS Chatbot.
3. Choose the Room settings icon on the top right and choose **Manage Webhooks and Bots**.

   Amazon Chime displays the webhooks associated with the chat room.

> **Note**
>
> You can have multiple webhooks in a single Amazon Chime chat room.
> For example, in an **Amazon Chime** chat room, one webhook could send notifications for Amazon CloudWatch alarms and another webhook could send AWS Security Hub security alerts. Each webhook receives notifications only for the SNS topics subscribed to it. All chat room members can see all of the notifications from each of the SNS topics.

4. For the webhook, choose **Copy URL** and choose **Done**.

   If you need to create a new webhook for the chat room, choose **Add webhook**, enter a name for the webhook in the **Name** field, and choose **Create**.
5. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
6. Choose **Configure new client**.
7. Choose **Amazon Chime** and choose **Configure**.
8. Under **Configuration details**, enter a name for your configuration. The name must be unique across your account and can't be edited later.
9. If you want to enable logging for this configuration, choose **Send logs to CloudWatch**. For more information, see Amazon CloudWatch Logs for AWS Chatbot.

   > **Note**
   >
   > There is an extra charge for using CloudWatch Logs.
10. For **Configure Amazon Chime webhook**, do the following.

    a. Paste the webhook URL that you copied from Amazon Chime.

    b. For **Webhook description**, use the following naming convention to describe the purpose of the webhook: **`Chat_room_name/Webhook_name`**. This helps you associate Amazon Chime webhooks with their AWS Chatbot configurations.
11. For **IAM permissions**, set the IAM permissions for AWS Chatbot.

    a. For **Role**, choose **Create a new role from template**. If you want to use an existing role instead, choose it from the **IAM Role** list. To use an existing IAM role, you might need to modify it for use with AWS Chatbot. For more information, see Configuring an IAM Role for AWS Chatbot.

   b. For **Policy templates**, choose **Notification permissions**. This is the IAM policy provided by AWS Chatbot. It provides the necessary Read and List permissions for CloudWatch alarms, events and logs, and for Amazon SNS topics.

   c. For **Role name**, enter a name. Valid characters: a-z, A-Z, 0-9.

12. Set up the SNS topics that will send notifications to the Amazon Chime webhook.

    a. For **SNS Region**, choose the AWS Region that hosts the SNS topics for this AWS Chatbot subscription.

    b. For **SNS topic**, choose the SNS topic for the client subscription. This topic determines the content that's sent to the Amazon Chime webhook. If the region has additional SNS topics, you can choose them from the same dropdown list.

    c. If you want to add an SNS topic from another Region to the notification subscription, choose **Add another Region**.

    > **Note**
    >
    > For a tutorial on subscribing existing Amazon SNS topics to AWS Chatbot, see Tutorial: Subscribing an Amazon SNS topic to AWS Chatbot.

13. Choose **Configure**.

Notifications from supported services that publish to the chosen SNS topics will now appear in the Amazon Chime chat room.

You can configure as many webhooks as you need. The SNS topics that you choose also must be configured in the services for which you want to receive notifications. For more information, see Monitoring AWS services using AWS Chatbot.

If you want to allow AWS Chatbot to answer questions about your AWS resources, turn on AWS Resource Explorer in the Resource Explorer Console. For more information, see Getting started with Resource Explorer in the *AWS Resource Explorer User Guide*.

## Step 2: Test notifications from AWS services to Amazon Chime

To verify that an Amazon Simple Notification Service (Amazon SNS) topic sends notifications to your Amazon Chime chat room, you can test your setup by sending a notification. To test your notifications, ensure your topics are assigned to a service supported by AWS Chatbot. For a list of supported services, see Monitoring AWS services using AWS Chatbot. You can also test notifications by using CloudWatch. For more information, see Test notifications from AWS services to Amazon Chime using CloudWatch.

**Testing notifications with configured clients**

1. Open the AWS Chatbot console.
2. Choose the configured client you want to test.
3. In the configured client, choose the webhook to send a test notification to.
4. Choose **Send test message**.
5. View the confirmation message at the top of the screen that shows a message was sent to your Amazon SNS topic.
6. Confirm the test message in your Amazon Chime chat room.

## Next steps

After you configure your chat clients and test that your notifications are working, you might want to explore some of the following topics:

- Learn about which other AWS services you can integrate with AWS Chatbot in Monitoring AWS services using AWS Chatbot.
- Learn about what you can customize using AWS Chatbot in Customizing AWS Chatbot.

# Tutorial: Get started with Microsoft Teams

To get started using AWS Chatbot to help manage your AWS infrastructure, follow the steps below to set up AWS Chatbot with chat rooms and Amazon SNS topic subscriptions. Note that AWS Chatbot is approved by your Microsoft Teams administrator.

**Topics**

- Step 1: Configure a Microsoft Teams client
- Step 2: Configure a Microsoft Teams channel
- Step 3: Test notifications from AWS services to Microsoft Teams
- Configuring Microsoft Teams channels using AWS CloudFormation
- Next steps

## Prerequisites

Before you get started, make sure you've completed the tasks in Setting up AWS Chatbot. You should also ensure Microsoft Teams is installed and approved by your organization administrator. You will need to choose a permissions scheme in the following procedure. This scheme determines the permissions your channel members will have and what AWS Chatbot can do on your behalf. For more information about AWS Chatbot permissions, see Understanding permissions You must also create or choose a channel to be used in your AWS Chatbot configuration. This channel is used to monitor and operate your AWS resources.

> **Note**
>
> The following IAM permissions are required to create a Microsoft Teams configuration:
>
> - GetMicrosoftTeamsOauthParameters
> - RedeemMicrosoftTeamsOauthCode
> - CreateMicrosoftTeamsChannelConfiguration
>
> If you have less than administrative permissions, ensure you have the aforementioned permissions to create a configuration.

## Step 1: Configure a Microsoft Teams client

To allow AWS Chatbot to send notifications or run commands in your Microsoft Teams channel, you must configure AWS Chatbot with Microsoft Teams.

**To configure a Microsoft Teams client**

1. Add AWS Chatbot to your team:

   a. In Microsoft Teams, find your team name and choose **...**, then choose **Manage team**.

   b. Choose **Apps**, then choose **More apps**.

   c. Enter **aws** in the search bar to find AWS Chatbot.

   d. Select the bot.

   e. Choose **Add to a team** and complete the prompt.

2. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.

3. Under **Configure a chat client**, choose **Microsoft Teams**, then choose **Configure client**.

4. Copy and paste your Microsoft Teams channel URL.

> **Tip**
>
> Your channel URL contains your tenant, team, and channel IDs. You can find your channel URL by right clicking on the channel in your Microsoft Teams channel list and copying the link. Your channel ID is the portion of your channel URL after the path `/channel/`, starting with `19%3` and likely ending with either `thread.tacv2` or `thread.skype`.
> For example, the bolded portion of the following channel URL is its channel ID: https://teams.microsoft.com/l/channel/**19%3Ae5eace25j32023jga835103358eapge3t8235%40thread.tacv2**/ChannelName?groupId=0d36500a-6023-419c-8c36-7e21f19b0135&tenantId=5fe61832-9f46-403b-a7db-cf9cf2e38199.

5. Choose **Configure**.

> **Note**
>
> After choosing **Configure**, you'll be redirected to Microsoft Team's authorization page to request permission for AWS Chatbot to access your information. For more information, see Chat client application permissions for AWS Chatbot.

6. On the Microsoft Teams authorization page, choose **Accept**.

# Step 2: Configure a Microsoft Teams channel

To allow AWS Chatbot to send notifications or run commands in your Microsoft Teams channel, you must also configure AWS Chatbot with a Microsoft Teams channel. Channel configuration consists of:

- Associating a channel with the configuration
- Defining user permissions, which dictate what tasks users can perform in a channel
- (Optional) Adding Amazon SNS topics, which AWS Chatbot uses to send notifications to your channel

> **Note**
>
> Microsoft Teams doesn't currently support AWS Chatbot in private channels. For more information, see Private channel limitations.

**To configure a Microsoft Teams channel**

1. Associate a channel with your configuration:

   a. On the **Team details** page in the AWS Chatbot console, choose **Configure new channel**.

   b. Under **Configuration details**, enter a name for your configuration. The name must be unique across your account and can't be edited later.

   c. If you want to enable logging for this configuration, choose **Publish logs to Amazon CloudWatch Logs**. For more information, see Amazon CloudWatch Logs for AWS Chatbot.

   > **Note**
   >
   > There is an extra charge for using CloudWatch Logs.

   d. For **Team channel**, paste your Microsoft Teams channel URL.

2. Define user permissions:

   a. Choose your **Role Setting**.

> **Tip**
>
> Your role setting dictates what permissions your channel members have. A channel role gives all members the same permissions. This is useful if your channel members typically perform the same actions in Microsoft Teams. A user role requires your channel members to choose their own roles. As such, different users in your channels can have different permissions. This is useful if your channel members are diverse or you don't want new channel members to perform actions as soon as they join the channel. For more information, see Role setting.

Channel role

1. For **Role setting**, choose **Channel role**.
2. For **Channel role**, choose **Create new role**. If you want to use an existing role instead, choose **Use an existing role**. To use an existing IAM role, you will need to modify it for use with AWS Chatbot. If you want your users to be able to use Amazon Q, attach the AmazonQDeveloperAccess policy. For more information, see Configuring an IAM Role for AWS Chatbot.
3. For **Role name**, enter a name. Valid characters: a-z, A-Z, 0-9, .\w+=,.@-_.
4. For **Policy template**, select **Amazon Q permissions** and any other templates you wish to use.

User roles

1. For **Role setting**, choose **User roles**.

b. Select the policies that will make up your channel guardrails. Your channel guardrails control what actions are available to your channel members. Add **AmazonQDeveloperAccess** as a channel guardrail to allow your users to use Amazon Q in your Microsoft Teams channel.

3. (Optional) Add Amazon SNS topics:

> **Note**
>
> If you want to receive notifications in your Slack channel, complete these steps.

- Choose your notification settings:

  i. For **SNS Region**, choose the AWS Region that hosts the SNS topics for this AWS Chatbot subscription.

  ii. For **SNS topic**, choose the Amazon SNS topic for the client subscription. This topic determines the content that's sent to the Microsoft Teams channel. If the region has additional SNS topics, you can choose them from the same dropdown list. The SNS topics you choose must be configured in the services for which you want to receive notifications. For more information, see Monitoring AWS services using AWS Chatbot.

  iii. To add an Amazon SNS topic from another AWS Region to the notification subscription, choose **Add another Region**.

  > **Note**
  >
  > For a tutorial on subscribing existing Amazon SNS topics to AWS Chatbot, see Tutorial: Subscribing an Amazon SNS topic to AWS Chatbot.

  Notifications from supported services that publish to the chosen Amazon SNS topics will now appear in the Microsoft Teams channel.

4. Choose **Configure**.

> **Note**
>
> You can configure a Microsoft Teams channel to run commands to your AWS account. For more information, see Running AWS CLI commands from chat channels.

You can configure as many channels with as many topics as you need.

If you want to allow AWS Chatbot to answer questions about your AWS resources, turn on AWS Resource Explorer in the Resource Explorer Console. For more information, see Getting started with Resource Explorer in the *AWS Resource Explorer User Guide*.

## Step 3: Test notifications from AWS services to Microsoft Teams

To verify that an Amazon Simple Notification Service (Amazon SNS) topic sends notifications to your Microsoft Teams channel, you can test your setup by sending a notification. Ensure your AWS Chatbot configuration is subscribed to at least one Amazon SNS topic and that your topics are assigned to a service supported by AWS Chatbot. For a list of supported services, see Monitoring AWS services using AWS Chatbot. You can also test notifications by using CloudWatch. For more information, see Test notifications from AWS services to Microsoft Teams using CloudWatch.

**Testing notifications with configured clients**

1. Open the AWS Chatbot console.
2. Choose the configured client you want to test.
3. In the configured client, choose the channel to send a test notification to.
4. Choose **Send test message**.
5. View the confirmation message at the top of the screen that shows a message was sent to your Amazon SNS topic.
6. Confirm the test message in your Microsoft Teams channel.

## Configuring Microsoft Teams channels using AWS CloudFormation

You can automate Microsoft Teams channel configuration by using an AWS CloudFormation template. To use an AWS CloudFormation template, you need the **Team ID** and **Tenant ID** found under **Team details** in the AWS Chatbot console. For more information, see AWS::Chatbot::MicrosoftTeamsChannelConfiguration in the *AWS CloudFormation User Guide*.

## Next steps

After you configure your chat clients and test that your notifications are working, you might want to explore some of the following topics:

- Learn about which other AWS services you can integrate with AWS Chatbot in Monitoring AWS services using AWS Chatbot.
- Learn about what you can customize using AWS Chatbot in Customizing AWS Chatbot.
- Learn about what actions you can perform using AWS Chatbot in Performing actions.

- Learn what questions you can ask AWS Chatbot and Amazon Q in Asking questions from chat channels in AWS Chatbot.
- Learn how to receive AWS CodeStar notifications in your channels in Tutorial: Receive Developer Tools notifications in Microsoft Teams.

# Tutorial: Create or configure an Amazon Simple Notification Service topic as a notification target for Microsoft Teams and AWS CodeStar

The notifications feature in the Developer Tools console is a notifications manager for subscribing to events in AWS CodeBuild, AWS CodeCommit, AWS CodeDeploy, and AWS CodePipeline. It has its own API, AWS CodeStar. Notification rule targets defined within AWS CodeStar are currently Amazon SNS topics or AWS Chatbot clients configured for Slack channels, but not AWS Chatbot clients configured for Microsoft Teams. However, to receive notifications about resources of interest in Microsoft Teams, you can use an Amazon SNS topic as a target in both your Microsoft Teams channel configuration and your AWS CodeStar notification rules. This tutorial describes two methods to achieve this:

- Configure an existing Amazon SNS topic to use as a notification target.
- Create a new Amazon SNS topic as a target by editing notification rules.

## Prerequisites

This tutorial assumes you already have some familiarity with Amazon SNS topics, AWS CodeStar, and AWS Chatbot. It also assumes that you've already created at least one notification rule in AWS CodeStar and that you've configured at least one AWS Chatbot client for Microsoft Teams.

For more information see the following topics:

- Tutorial: Get started with Microsoft Teams
- What is the Developer Tools console? in the *Developer Tools console User Guide*.
- What are notifications? in the *Developer Tools console User Guide*.

# Step 1: Configure or create an Amazon SNS topic

## Configure an existing Amazon SNS topic to use as a notification target

In this procedure you edit an existing Amazon SNS topic used in your Microsoft Teams channel configuration to be used as a notification rule target in AWS CodeStar.

**To configure an existing Amazon SNS topic to use as a notification rule target**

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Select your configured Microsoft Teams chat channel.
3. Choose the channel you want to receive notifications in.
4. Choose **Edit**.
5. In **Notifications**, identify the name of the Amazon SNS topic you want to use.
6. Follow the steps in To configure an Amazon Simple Notification Service topic to use as a target for AWS CodeStar notification rules in the *Developer Tools User Guide*.
7. The Amazon SNS topic is now associated with both AWS CodeStar and your chosen Microsoft Teams channel configuration.

## Create a new Amazon SNS topic as a target by editing notification rules

In this procedure you create a new Amazon SNS topic by editing an existing notification rule. You then use this Amazon SNS topic in your Microsoft Teams channel configuration.

**To create a new Amazon SNS topic to use as a notification rule target**

1. Edit your notification rule:

   > **Note**
   >
   > The Region in which this notification rule is created must be added as a notification Region in your Microsoft Teams configuration.

   a. Open the Developer Tools console.

   b. In the navigation pane, choose **Settings** and then choose **Notification rules.**

   c. Select the appropriate rule and choose **Edit**.

d. In **Targets**, choose **Create target**.

e. Choose **Amazon SNS topic** as your target type.

f. Enter a topic name.

> **Note**
>
> This topic will be used in your Microsoft Teams configuration.

2. Edit your Microsoft Teams configuration:

a. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.

b. Select your configured Microsoft Teams chat client.

c. Select the channel in which you want to receive notifications.

d. Choose **Edit**.

e. (Optional) If you have no Regions or you don't currently have the Region you used for Developer Tools added, choose **Add another Region** in **Notifications** and select the appropriate Region.

f. In **Topics**, search for and select the Amazon SNS topic you created while editing your notification rule.

g. Choose **Configure**.

## Tutorial: Subscribing an Amazon SNS topic to AWS Chatbot

You can quickly subscribe existing Amazon SNS topics to the AWS Chatbot service. You associate the new subscriptions to a chat channel. After doing so, the messages from those topics will appear in the channel. The Amazon SNS topics must be associated with AWS services that AWS Chatbot supports, and may also require further configuration, such as association with a CloudWatch rule. This procedure is most useful if you have SNS topics that are already doing significant work with CloudWatch Events and CloudWatch alarms in AWS cloud services supported by AWS Chatbot.

> **Note**
>
> You can set up each supported AWS service to *target* one or more SNS topics to send notifications to AWS Chatbot. You do this using each AWS service's console, or using AWS CloudFormation. If you already have Amazon SNS topics set as targets for supported

services, you can configure AWS Chatbot to use those topics. Notifications from subscribed topics will automatically appear in your chat channels without further configuration.

> **Note**
>
> If your SNS topic is encrypted, you must add a section to your AWS KMS key policy to give the sending service permissions to post events to the encrypted SNS topics. For more information, see Setting up Amazon SNS topics.

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Under **Configured clients**, choose your chat client.
3. Choose any channel in your chat client configuration.
4. Choose **Edit**. The configuration page for the channel appears. Note that the **Region** Notifications is already configured.
5. In the **Notifications** panel:
   - If you need to apply an Amazon SNS topic from another region, choose **Add another Region**.
6. For each **Region** in the chat channel, select the Amazon SNS topic you want to add.
7. When finished, choose **Save**.
8. To check for the subscription, click on any subscription entry in the AWS Chatbot console. The Amazon SNS console opens, showing the list of subscriptions for the selected topic.

## Test notifications from AWS services to chat channels using CloudWatch

To verify that an Amazon Simple Notification Service (Amazon SNS) topic sends notifications to your chat channels, you can test your setup by sending a notification. Any SNS topic can send notifications to your chat rooms, but the topic must be assigned to a service supported by AWS Chatbot. For a complete list of supported services, see Monitoring AWS services using AWS Chatbot.

> **Note**
>
> CloudWatch alarms and events are separately configured and have different characteristics for use with AWS Chatbot.

The following procedure uses a CloudWatch alarm because most AWS services supported by AWS Chatbot send their event and alarm data to CloudWatch.

You configure CloudWatch alarms using performance metrics from the services that are active in your account. When you associate CloudWatch alarms with an Amazon SNS topic that is mapped to AWS Chatbot, the Amazon SNS topic sends the CloudWatch alarm notifications to the chat rooms. For more information, see Monitoring AWS services using AWS Chatbot and the Troubleshooting topic.

**To test notifications to configured chat clients**

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create alarm**.
3. Select the correct AWS **Region** at the top right of the AWS console, that contains the Amazon SNS topic you need. (**Tip:** to make sure you have the right region for your SNS topics for testing alarms, you can check the AWS Chatbot configuration to see the regions for all configured SNS topics in each channel or webhook.)
4. Choose **Select metric**, and choose the **SNS** service namespace. (All CloudWatch alarms use service *metrics* to generate their notifications, and you need to select one for this example.)

   a. Choose **Topic metrics**.

   b. Choose the check box for the SNS topic next to its **Topic Name** and **Metric Name**. Any SNS topics that you configured with AWS Chatbot appear in this list.

      *Important*: if you do not see your desired Amazon SNS topic in the SNS Topic list, make sure to select the correct AWS Region in the AWS console when you begin configuring the new CloudWatch alarm.

   c. Choose **Select metric**.

   The **Specify metric and conditions** page shows a graph and other information about the metric and statistic.

5. For **Conditions** (the circumstances under which the CloudWatch alarm fires and an action takes place), choose the following options:

    a. For **Threshold type**, choose **Static**.

    b. For **Whenever *metric* is**, choose **Lower/Equal <=threshold**.

    c. For **than...**, specify a threshold value of **1**. This setting ensures you will trigger the test notification within one minute.

    d. Under **Additional configuration**, do the following:

        i. For **Datapoints to alarm**, select **1 out of 1**.

        ii. For **Missing data treatment**, select **Treat missing data as bad**.

    e. Choose **Next**.

6. Choose **Configure actions**. Here, you set the *action* to create SNS notifications when the metric threshold is exceeded.

    For **Notification**, choose the following options.

    a. For **Whenever this alarm state is...**, choose **In Alarm**.

    b. For **Select an SNS topic**, choose **Select an existing SNS topic**.

    c. For **Send a notification to...**, choose your SNS topic that has a subscription to AWS Chatbot. If the SNS topic is subscribed in AWS Chatbot, the endpoint value for AWS Chatbot appears in the **Email (endpoints)** field.

        > **Note**
        >
        > If the endpoint value doesn't appear in the **Email (endpoints)** field, make sure that the SNS topic is set up correctly in the Microsoft Teams channel, Slack channel or Amazon Chime webhook. For more information, see Setting up AWS Chatbot with Microsoft Teams, Setting up AWS Chatbot with Slack, or Setting up AWS Chatbot with Amazon Chime.

    d. Choose **Next**.

7. Enter a name and description for the alarm. The name must contain only ASCII characters. Then, choose **Next**.

8. For **Preview and create**, confirm that the information and conditions are correct, then choose **Create alarm**.

When the alarm triggers for the first time, you should receive the first test notification in your chat room, confirming that AWS Chatbot is working correctly and receiving alarm notifications from Amazon CloudWatch.

## Next steps

Once you've taken the necessary steps to set up AWS Chatbot, you can get started configuring the chat client of your choice. For a step-by-step guide on how to do this, choose the appropriate tutorial below:

- Tutorial: Get started with Slack
- Tutorial: Get started with Amazon Chime
- Tutorial: Get started with Microsoft Teams

# Understanding AWS Chatbot permissions

AWS Chatbot requires an AWS Identity and Access Management (IAM) role to perform actions. Actions you can perform in your chat channels include running commands and responding to interactive messages. AWS Chatbot uses organization policies, service policies, channel roles, user roles, and channel guardrail policies to control the actions channel members can take. What your users can do is the intersection of your guardrail policies and what is allowed by their roles.

**Topics**

## Organization policies

### AWS Chatbot organization policies (Chatbot policies)

Organization administrators can manage multiple AWS Chatbot settings across all accounts within an organization using an AWS Chatbot organization policy (Chatbot policy). Chatbot policies define where AWS Chatbot can deliver notifications and if it can respond to AWS Chatbot mention events. For more information, see ???.

### Service control policies (SCPs)

Service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for the IAM users and IAM roles in your organization. For more information, Service control policies (SCPs) in the *AWS Organizations User Guide*.

# Role setting

## Channel role

A channel role gives all channel members the same permissions. This is useful if your channel members are similar users or they typically perform the same actions. You can use an existing role as your channel role or you can create a new role using templates. If you use a channel role, your channel members can still choose their own user roles. Your channel role is restricted by your guardrail policies. You can set your channel role in channel configurations from the AWS Chatbot console.

### Channel role templates

There are eight templates that can be used to create a channel role:

- Notification permissions
- Read-only command permissions
- Lambda-invoke command permissions
- AWS Support command permissions
- Incident Manager permissions
- Resource Explorer permissions
- Amazon Q permissions
- Amazon Q operations assistant permissions

You can use any and all combinations of these templates to suit your needs. For example, if you want to create a configuration that only delivers notifications, choose **Notification permissions** as your policy template. If you want your channel members to run read-only commands exclusively and you want notifications to be delivered, choose **Read-only command permissions** and **Notification permissions** as your policy templates. For more information, see ???.

## User roles

User roles require channel members to choose their own roles. As a result, different users in your channel can have different permissions. If you have a diverse set of channel members or you don't want new channel members to perform actions as soon as they join your channel, user roles are appropriate. Under this schema, your channel members must have applied a user role to perform actions. When channel members apply a user role, it is mapped to their chat client ID.

Administrators can unmap user roles from chat client IDs in the AWS Chatbot console. Your channel member's actions are limited by your guardrail policies, despite what user roles they may have applied. For more information on managing user roles, see Managing user roles in AWS Chatbot.

### User role requirement

Administrators can require user roles for all current channel members and channels and all channels created in the future by enabling a user role requirement in the AWS Chatbot console. Individual channels can't override this requirement. This can be done at the account level in **User permissions**, if you want to require every workspace and channel to use user roles. It can also be done at the channel configuration level wherein a channel level administrator can enable the user role requirement.

> **Note**
>
> This feature is enforced at the account level.

## Channel guardrail policies

Guardrail policies provide detailed control over what actions are available to your channel members and what actions AWS Chatbot can perform on your behalf. They constrain and take precedence over both user roles and channel roles. For example, if a user has a user role that allows administrator access, and they belong to a channel where the channel role or the guardrail policies limit permissions on one or more services, the user will have less than administrator-level access. You can set, view, and edit your guardrail policies in the AWS Chatbot console. If you had an AWS Chatbot configuration before the expansion of available commands on 11/28/2021, you may have a protection policy applied as one of your guardrail policies.

> **Note**
>
> AWS Service Roles IAM policies can't be used as guardrail policies.

## Non-supported operations

AWS Chatbot doesn't support running commands for operations in the following JSON policy:

```
{
  "Statement": [
    {
      "Action": [
        "appsync:ListApiKeys",
        "chatbot:*",
        "codecommit:GetFile",
        "codecommit:GetCommit",
        "codecommit:GetDifferences",
        "cognito-idp:*",
        "cognito-identity:*",
        "connect:GetFederationToken",
        "dynamodb:BatchGetItem",
        "dynamodb:GetItem",
        "ec2:GetPasswordData",
        "ecr:GetAuthorizationToken",
        "ecr:GetLogin",
        "gamelift:RequestUploadCredentials",
        "gamelift:GetInstanceAccess",
        "identitystore:*",
        "lightsail:DownloadDefaultKeyPair",
        "lightsail:GetInstanceAccessDetail",
        "lightsail:GetKeyPair",
        "lightsail:GetKeyPairs",
        "lightsail:UpdateRelationalDatabase",
        "iam:*",
        "kms:*",
        "redshift:GetClusterCredentials",
        "sdb:*",
        "secretsmanager:*",
        "sso:*",
        "sso-admin:*",
        "sso-oidc:*",
        "storagegateway:DescribeChapCredentials",
        "sts:*",
        "s3:GetObject",
        "s3:HeadObject",
        "s3:PutObject",
        "s3:GetBucketPolicy",
        "snowball:GetJobUnlockCode"
      ],
      "Effect": "Deny",
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

# Securing your AWS organization in AWS Chatbot

You can secure your AWS organization or organizational units (OUs) using organization policies. 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, you can apply organization policies such as a Chatbot organization policy (Chatbot policy) and service control policies (SCPs) to any or all of your accounts. A Chatbot policy defines which permissions models, chat platforms, and chat workspaces can be used to access your accounts. SCPs limit permissions for entities in member accounts, including each AWS account root user. Effective chat application permissions are the intersection between organization level controls (organization policies) and account level controls (the section called “User role requirement”, AWS Chatbot configuration resources). For more information about organization policies, see Managing policies with AWS Organizations in the *AWS Organizations User Guide*.

**Topics**

- AWS Chatbot organization policies
- Service control policies (SCPs) for AWS Chatbot

## AWS Chatbot organization policies

Organization administrators can manage multiple AWS Chatbot settings across all accounts within an organization using a Chatbot policy. Chatbot policies define where AWS Chatbot can deliver notifications and if it can respond to AWS Chatbot mention events. Using Chatbot policies, administrators can:

- Enforce which chat platforms can be used across your organization (Amazon Chime, Microsoft Teams, and Slack)
- Restrict chat client access to specific workspaces and teams.
- Restrict Slack channel visibility to either public or private channels.
- Set and enforce specific role settings.

Chatbot policies restrict and take precedence over account level settings like role settings and ???. Administrators can define rules in a policy and apply those rules to an entire organization or a group of accounts, referred to as OUs. For more information, see Managing organizational units in the *AWS Organizations User Guide*. You can access and modify these policies from the AWS Chatbot console or the AWS Organizations console. For more information about organization policies, see Managing policies in AWS Organizations *AWS Organizations User Guide*.

If your users try to perform an action restricted by your Chatbot policy, they are informed via error message that they are disallowed due to the policy and we recommend that they contact their organization administrator.

> **Note**
>
> AWS Chatbot organization policies are validated at runtime, so existing resources are continuously checked for compliance. There is no overlap with existing IAM permissions as there aren't currently any runtime-based IAM permissions for sending notifications or interacting with AWS Chatbot.

**Topics**

- Example AWS Chatbot organization policy
- Enabling Chatbot policies
- Disabling Chatbot policies
- Tutorial: Creating a Chatbot policy in AWS Chatbot
- Editing a Chatbot policy in AWS Chatbot
- Deleting a Chatbot policy in AWS Chatbot

## Example AWS Chatbot organization policy

The following policy allows restricted AWS Chatbot access for selected Slack workspaces and a Microsoft Teams tenant.

```
{
    "chatbot":{
       "platforms":{
          "slack":{
             "client":{
                "@@assign":"enabled"
             },
             "workspaces": { // limit 255
```

```
                  "@@assign":[
                     "Slack-Workspace-Id1",
                     "Slack-Workspace-Id2"
                  ]
            },
            "default":{
               "supported_channel_types":{
                  "@@assign":[
                     "private"
                  ]
               },
               "supported_role_settings":{
                  "@@assign":[
                     "user_role"
                  ]
               }
            },
            "overrides":{ // limit 255
               "Slack-Workspace-Id2":{
                  "supported_channel_types":{
                     "@@assign":[
                        "public",
                        "private"
                     ]
                  },
                  "supported_role_settings":{
                     "@@assign":[
                        "channel_role",
                        "user_role"
                     ]
                  }
               }
            }
         },
         "microsoft_teams":{
            "client":{
               "@@assign":"enabled"
            },
            "tenants":{ // limit 36
               "Microsoft-Teams-Tenant-Id":{ // limit 36
                  "@@assign":[
                     "Microsoft-Teams-Team-Id"
                  ]
               }
            },
            "default":{
               "supported_role_settings":{
                  "@@assign":[
                     "user_role"
                  ]
               }
            },
            "overrides":{ // limit 36
               "Microsoft-Teams-Tenant-Id":{
                  "Microsoft-Teams-Team-Id":{
                     "supported_role_settings":{
                        "@@assign":[
                           "channel_role",
                           "user_role"
                        ]
                     }
                  }
               }
            }
         }
      },
      "default":{
         "client":{
            "@@assign":"disabled"
         }
      }
   }
}
```

### For Slack

- The Slack client is enabled.
- The allowed Slack workspaces are *Slack-Workspace-Id1* and *Slack-Workspace-Id2*.
- The default settings for Slack are to only allow private channels and User level IAM roles.
- There is an override for the workspace *Slack-Workspace-Id2* that allows both public and private channels as well as both Channel level IAM roles and User level IAM roles.

### For Microsoft Team

- The Microsoft Teams is enabled.

- The allowed Teams tenants are *Microsoft-Teams-Tenant-Id* with the team *Microsoft-Teams-Team-Id*.
- The default settings are to only allow User level IAM roles.
- There is an override for the tenant *Microsoft-Teams-Tenant-Id* that allows both Channel level IAM roles and User level IAM roles for the team *Microsoft-Teams-Team-Id*.

### Additional details

- The `default` block at the bottom sets the client to be disabled, which disables AWS Chatbot across the organization unless overridden at a lower level. This means Amazon Chime is disabled in this example. This default also disables any new chat platform that AWS Chatbot supports. For example, if AWS Chatbot supports a new chat platform, this default disables that newly supported chat platform as well.

## Enabling Chatbot policies

Before you can create Chatbot policies, you must first enable them using the AWS Organizations console. For more information, see Enabling a policy type in the *AWS Organizations User Guide*.

## Disabling Chatbot policies

If you no longer want to use Chatbot policies in your organization, you can disable them to prevent accidental use. For more information, see Disabling a policy type in the *AWS Organizations User Guide*.

## Tutorial: Creating a Chatbot policy in AWS Chatbot

In this tutorial, you use the AWS Chatbot console to create a Chatbot policy that:

- Restricts chat client access to Slack
- Specifies usable Slack workspaces
- Restricts usage to private channels
- Requires user-level roles

Subsequently, all AWS Chatbot configurations in your organization must adhere to these specifications.

### Topics

- Prerequisites
- Step 1: Create a new Chatbot policy
- (Optional) Step 2: Testing your Chatbot policy

## Prerequisites

You must have already created an organization using AWS Organizations. For more information, see Managing an organization with AWS Organizations in the *AWS Organizations User Guide*.

## Step 1: Create a new Chatbot policy

### To create a new Chatbot policy

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. In the left sidebar menu, choose **Organization settings**.
3. Choose **Chatbot policies**.
4. Choose **Create Chatbot policy**.
5. - **Enable Chatbot Orgs policies:**

     > **Note**
     >
     > Before you can create and attach a policy to your organization, you must enable that policy type for use. This is a one-time task on the organization root. You can enable a policy type from only the organization's management account. For more information, see Enabling and disabling policy types in the *AWS Organizations User Guide*.

     On the Chatbot policies page, choose **Enable**.
6. a. **Enter your policy Details:**

   Enter a policy name.

   b. (Optional) Enter a policy description.
7. (Optional) Add tags.

8.
   a. **Configure chat client access:**

      In **Set Amazon Chime chat client access**, choose **Deny Chime access**.
   b. In **Set Microsoft Teams client access**, choose **Deny access to all Teams**.
   c. In **Set Slack chat client access**, choose **Restrict access to named Slack workspaces**:

      i. Enter a Slack workspace ID.

         > **Tip**
         >
         > You can find your workspace ID in the AWS Chatbot console by choosing the configured client in the left sidebar and looking under **Workspace details**.

      ii. (Optional) Choose **Add new workspace ID** to add another Slack workspace.

      iii. Choose **Add**.
   d. Select **Enable usage to only private Slack channels**.
9.
   - **Set IAM permission types:**

     Select **Enable User level IAM role**.
10. Choose **Create policy**.

### (Optional) Step 2: Testing your Chatbot policy

If you already have an AWS Chatbot configuration, you can sign in as a user in any of your member accounts and try to perform any of the following actions:

- Create an AWS Chatbot configuration for Microsoft Teams
- Create a Slack AWS Chatbot configuration for a workspace you didn't specify in your policy
- Create a Slack AWS Chatbot configuration that uses a channel role

When you try to perform these actions, you should receive an error message that explains why you’re disallowed.

## Editing a Chatbot policy in AWS Chatbot

If you need to make changes to your Chatbot policy, you can edit it.

**To edit a Chatbot policy**

1. Sign in to the AWS Chatbot console;
2. In the left sidebar meny, choose **Organization settings**.
3. Choose **Chatbot Policies**.
4. Select the name of the policy.
5. Choose **Edit policy**.
6. Make your edits.
7. Choose **Save changes**.

## Deleting a Chatbot policy in AWS Chatbot

If you no longer need a Chatbot policy, you can delete it.

**To delete a Chatbot policy**

1. Sign in to the AWS Chatbot console;
2. In the left sidebar meny, choose **Organization settings**.
3. Choose **Chatbot Policies**.
4. Select the name of the policy.
5. Choose **Delete policy**.
6. Confirm your deletion by entering the policy name.
7. Choose **Delete**.

# Service control policies (SCPs) for AWS Chatbot

Service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for the IAM users and IAM roles in your organization. For more information, Service control policies (SCPs) in the *AWS Organizations User Guide*.

SCPs in AWS Chatbot function similarly to channel guardrail policies, but are implemented on the organization level. You can use SCPs to secure your organizations by restricting what APIs can be used to configure AWS Chatbot and which services and operations can be run using AWS Chatbot in chat channels. This doesn’t impact resources that are already created or the ability to respond to commands in chat channels. The global condition key, `aws:ChatbotSourceArn`, is attached to all sessions created through AWS Chatbot. You can use this condition key to restrict which AWS Chatbot API operations can be run using AWS Chatbot as opposed to other platforms such as the CLI or console.

**Topics**

- Example Service control policies

## Example Service control policies

### Example 1: Deny all IAM operations

The following SCP denies all IAM operations invoked through all AWS Chatbot configurations.

```
{
    "Effect": "Deny",
    "Action": "iam:*",
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:ChatbotSourceArn": "arn:aws:chatbot::*"
        }
    }
}
```

### Example 2: Deny S3 bucket put requests from a specified Slack channel

The following SCP denies S3 put requests on the specified bucket for all requests originating from a Slack channel.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleS3Deny",
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringLike": {
                      "aws:ChatbotSourceArn": "arn:aws:chatbot::*:chat-configuration/
slack-channel/*"
                }
            }
        }
    ]
}
```

# Chat client application permissions for AWS Chatbot

When you install AWS Chatbot on Microsoft Teams and Slack applications, each authorization process requests approval to grant AWS Chatbot app permissions. The following permissions are requested for each chat client.

## Microsoft Teams permissions

- Team.ReadBasic.All
- Channel.ReadBasic.All
- ChannelMember.Read.All
- User.ReadBasic.All

For more information, see Microsoft Graph permissions reference.

## Slack permissions

- app_mentions:read
- channels:read
- chat:write
- chat:write.public
- groups:read
- team:read
- users:read

For more information, see Permission scopes.

# Managing user roles in AWS Chatbot

All users can manage user roles, however how you can manage these roles depends on your user type. This topic describes how each user type can manage user roles.

**Topics**

## Managing user roles as a channel member in AWS Chatbot

Channel members can switch their user roles from their chat channels. Additionally, channel members can unmap user roles from chat client IDs using the AWS Chatbot console.

**Topics**

### Adding a user role from a chat channel using AWS Chatbot

If you are a new channel member or your channel permission approach changes, AWS Chatbot will prompt you to add a user role.

**To add a user role from a chat channel**

1. Choose **Let's get started**.
2. Choose an account to add a role.

   > **Note**
   >
   > This link will take you directly to the AWS Chatbot console.

3. In **User role**, choose a role.
4. Choose **Save**.

> **Note**
>
> Choosing **Save** takes you to an authorization page to fetch your chat client identity. This identity is mapped to your chosen role.

5. Choose **Allow**.

## Switching user roles from a chat channel using AWS Chatbot

If you find that your current user role doesn't have the right permissions to achieve your desired task, you can switch roles directly from Microsoft Teams and Slack.

> **Note**
>
> If you are unable to run a particular command after switching roles, contact your administrator regarding the channel guardrails in place.

### To switch a user role from a chat channel

1. In your chat channel, enter `@aws switch-role`.
2. Choose the account that you want to switch roles for.

   > **Note**
   >
   > This link will take you directly to the AWS Chatbot console.

3. In the AWS Chatbot console, choose **Choose user role**.
4. In **User role**, choose a user role.
5. Choose **Save**.

   > **Note**
   >
   > Choosing **Save**, takes you to an authorization page. This is so your chat client identity can be retrieved and associated with your chosen role.

6. On the authorization page, choose **Allow**.

## Unmapping a user role using AWS Chatbot

If you have a user role applied that you no longer need, you can unmap it.

**To unmap a user role**

1. Open the AWS Chatbot console.
2. Choose a configured client.
3. In **User role**, choose **Clear role**.

# Managing user roles as an administrator in AWS Chatbot

Administrators can unmap user roles from channel members' chat client IDs from the **User permissions** page in the AWS Chatbot console. Administrators can also require user roles by enabling a user role requirement in the **User permissions** page. This requirement can be applied to all workspaces and channels or to individual channel configurations. For more information on user role requirements, see User role requirement.

> **Note**
>
> Administrators can't map user roles. Only channel members have this ability.

**Topics**

## Unmapping a user role in AWS Chatbot

You can unmap a user role from a chat client ID. When you unmap a user role, it will no longer appear your **Mapped roles** table.

**To unmap a user role**

1. Open the AWS Chatbot console.
2. Under **Account settings**, choose **User permissions**.
3. In **Mapped roles**, select the roles you want to unmap.

4. Choose **Unmap**.

### Enabling a user role requirement in AWS Chatbot

You can enable a user role requirement to force users to apply a user role before running commands in Microsoft Teams and Slack.

**To enable a user role requirement**

1. Open the AWS Chatbot console.
2. Under **Account settings**, choose **User permissions**.
3. In **User role requirement**, enable a user role requirement.

## Editing an IAM role for AWS Chatbot

You can create new IAM roles in the AWS Chatbot console, which provides a convenient way to deploy the AWS Chatbot service. You associate these roles with your chat channels or Amazon Chime webhooks. The AWS Chatbot console does not allow editing of IAM roles, including any roles that you've already created in the AWS Chatbot console.

> **Note**
>
> AWS requires that you use the IAM console to edit IAM roles. If you create roles in the AWS Chatbot console, you must use the IAM console to edit them. This might happen, for example, when you are using the AWS Chatbot service and a new release comes out that supports new features.

Use the IAM console to edit AWS Chatbot roles. You can use the entire set of IAM console features to specify permissions for your AWS Chatbot users.

**To edit roles**

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Choose the configured client, and choose the name of the configured channel or webhook.
3. Choose a role to edit:

Channel role

1. Choose the role you want to edit. When you choose a role, the IAM console opens, automatically showing role configuration page, with the Permissions tab displaying the selected role.

   > **Note**
   >
   > You can attach AWS managed policies and customer managed policies. AWS Chatbot roles support both types of IAM policies.

2. Choose **Add permissions** and then select **Attach Policies**.

User roles

1. Choose the **User role** tab.
2. Choose **Edit**.

   > **Note**
   >
   > You can attach AWS managed policies and customer managed policies. AWS Chatbot roles support both types of IAM policies.

3. Choose **Selected role information** then choose a role. When you choose a role, the IAM console opens automatically showing role configuration page.
4. Choose **Add permissions** and then select **Attach Policies**.

4. Choose the name of the policy that you want. You can use the **Search** box to search for the policy by name or by a partial string of characters. For example, all IAM policies associated with AWS Chatbot include the character string **Chatbot** as part of the policy name. If you want your users to be able to use Amazon Q, attach the **AmazonQFullAccess policy**.

5. You can attach any of three AWS managed policies to any role. You can use these policies as templates to create your own policies.

   - **ReadOnlyAccess**
   - **CloudWatchReadOnlyAccess**
   - **AWSSupportAccess**

The **ReadOnlyAccess** policy is automatically attached to any role that you create in the AWS Chatbot console.

The **AWSSupportAccess** policy is the only AWS managed policy that appears in the AWS Chatbot console when you configure new roles there.

You can use these policies to create your own policies that are less permissive and specify the resources their users can access. You can substitute these custom policies for the ones listed here.

6. Choose each of the policies that you want to attach to the role and choose **Attach policy**. If needed, use the Search box to locate the policies you're looking for.

   After you click **Attach policy**, the role's **Permissions** page opens and shows the change in the **Permissions** list.

> **Note**
>
> For more information about the customer managed policies and AWS managed policies described in this section, see IAM Policies for AWS Chatbot.
> For more information about editing IAM policies, see Editing IAM Policies. Exercise caution at all times when editing policies, and avoid overwriting existing customer managed policies.

## Managing IAM role permissions for running commands in AWS Chatbot

With AWS Identity and Access Management (IAM), you can use *identity-based policies*, which are JSON permissions policy documents, and attach them to an *identity*, such as a user, role, or group. These policies work with your guardrail policies to control what actions a user can perform. AWS Chatbot provides three IAM policies in the AWS Chatbot console that you can use to set up AWS CLI commands support for chat channels. Those policies include:

- **ReadOnly Command Permissions**
- **Lambda-Invoke Command Permissions**

- **AWS Support Command Permissions**

You can use any or all of these policies, based on your organization's requirements. To use them, create a new channel IAM role in your channel configuration using the AWS Chatbot console, and attach the policies there. You can also attach the policies to the AWS Chatbot IAM roles using the IAM console. The policies simplify AWS Chatbot role configuration and enable you to set up quickly.

You can use these IAM policies as templates to define your own policies. For example, all policies described here use a wildcard ("*") to apply the policy's permissions to all resources:

```
                "Resource": [
                 "*"
             ]
```

You can define custom permissions in a policy to limit actions to specific resources in your AWS account. These are called *resource-based permissions*. For more information on defining resources in a policy, see the section IAM JSON Policy Elements: Resource in the *IAM User Guide*.

For more information on these policies, see Configuring an IAM Role for AWS Chatbot.

## Using the AWS Chatbot read-only command permissions policy

The AWS Chatbot **ReadOnly Command Permissions** policy controls access to several important AWS services, including IAM, AWS Security Token Service (AWS STS), AWS Key Management Service (AWS KMS), and Amazon S3. It disallows all IAM operations when using AWS commands in Microsoft Teams and Slack. When you use the **ReadOnly Command Permissions** policy, you allow or deny the following permissions to users who run commands in chat channels:

- IAM (Deny All)
- AWS KMS (Deny All)
- AWS STS (Deny All)
- Amazon Cognito (allows Read-Only, denies `GetSigningCertificate` commands)
- Amazon EC2 (allows Read-Only, denies `GetPasswordData` commands)
- Amazon Elastic Container Registry (Amazon ECR) (allows Read-Only, denies `GetAuthorizationToken` commands)
- Amazon GameLift (allows Read-Only, denies requests for credentials and `GetInstanceAccess` commands)

- Amazon Lightsail (allows List, Read, denies several key pair operations and `GetInstanceAccess`)
- Amazon Redshift (denies `GetClusterCredentials` commands)
- Amazon S3 (allows Read-Only commands, denies `GetBucketPolicy` commands)
- AWS Storage Gateway (allows Read-Only, denies `DescribeChapCredentials` commands)

The **ReadOnly Command Permissions** policy JSON code is shown following:

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iam:*",
                "kms:*",
                "sts:*",
                "cognito-idp:GetSigningCertificate",
                "ec2:GetPasswordData",
                "ecr:GetAuthorizationToken",
                "gamelift:RequestUploadCredentials",
                "gamelift:GetInstanceAccess",
                "lightsail:DownloadDefaultKeyPair",
                "lightsail:GetInstanceAccessDetails",
                "lightsail:GetKeyPair",
                "lightsail:GetKeyPairs",
                "redshift:GetClusterCredentials",
                "s3:GetBucketPolicy",
                "storagegateway:DescribeChapCredentials"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

## Using the AWS Chatbot Lambda-Invoke policy

The AWS Chatbot **Lambda-Invoke Command Permissions** policy allows users to invoke AWS Lambda functions in chat channels. This policy is an AWS managed policy that is not specific to AWS Chatbot, though it appears in the AWS Chatbot console.

By default, invoked Lambda functions can perform *any operation*. You might need to define a more restrictive inline IAM policy that allows permissions to invoke specific Lambda functions, such as functions specifically developed for your DevOps team that only they should be able to invoke, and deny permissions to invoke Lambda functions for any other purpose.

The following example shows the **Lambda-Invoke Command Permissions** policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:invokeAsync",
                "lambda:invokeFunction"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

You can also define resource-based permissions to allow invoking of Lambda functions only against specific resources, instead of the "*" wildcard that applies the policy to all resources. Always follow the IAM practice of granting only the permissions required for your users to do their jobs.

# Protection policy

The expansion of usable CLI commands occurred on 11/28/2021. This expansion can allow channel members to create, read, update, and delete your AWS resources. To prevent this, a protection policy is applied as a guardrail policy to existing AWS Chatbot configurations by default. Specifically, the protection policy restricts permissions and actions to what was available before all CLI commands were usable. This policy is detachable, but we strongly recommend it stay in place until you've verified that all your guardrails, channel IAM roles, and user-level roles align with your governance policy or channel requirements. You can detach this policy from:

- Individual workspaces.
- Individual channels in the channel configurations page.
- A selection of channels using the **Set guardrails** button.
- All channel configurations in the **User permissions** page of the AWS Chatbot console.

The protection policy contains the ReadOnlyAccess policy and the following JSON code:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:Invoke*",
                "support:*",
                "ssm-incidents:*"
            ],
            "Resource": "*"
        }
    ]
}
```

# Monitoring AWS services using AWS Chatbot

You can use AWS Chatbot to monitor and receive notifications about other AWS services. AWS Chatbot works with a number of AWS services, including Amazon CloudWatch, AWS Security Hub, and Amazon GuardDuty. All services that work with AWS Chatbot use Amazon SNS topics as targets to send event and alarm notifications. You may already have established Amazon SNS topics that send notifications to DevOps and development personnel as emails. Because AWS Chatbot redirects those Amazon SNS topics' notifications to chat rooms, you can map those Amazon SNS topics to a Amazon Chime webhook, Microsoft Teams channel, or Slack channel in the AWS Chatbot console.

> **Note**
>
> Not all service messages sent via Amazon SNS are supported. For more information about supported services, see ???.

When you create a new Amazon SNS topic, your services will require additional configuration.

If you want to customize the message content of default service notifications or customize messages for your application events, you can use custom notifications. For more information, see Custom notifications in AWS Chatbot.

**Topics**

- AWS Systems Manager
- AWS Systems Manager Runbooks
- AWS Systems Manager Incident Manager

You can set up the following AWS services to forward notifications to Amazon Chime, Microsoft Teams, or Slack chat rooms.

## AWS Billing and Cost Management

AWS Billing and Cost Management helps AWS account holders plan service usage, service costs, and instance reservations. You do this using several specific types of budgets, which track your unblended costs, subscriptions, refunds, and Reserved Instances. The service sends AWS Budget Alerts to an Amazon SNS topic. You then map the Amazon SNS topic in AWS Chatbot to send those notifications to your chat rooms.

For information about setting up Amazon SNS topics for AWS budgets, see Creating an Amazon SNS Topic for Budget Notifications in the *AWS Billing and Cost Management User Guide*.

## AWS CloudFormation

AWS CloudFormation is an infrastructure management service that helps you model and set up Amazon Web Services resources so you can spend less time managing those resources and more time focusing on the applications that you run in AWS. You create a template that describes all of the AWS resources (for example, Amazon EC2 instances or Amazon RDS DB instances) that you want, and AWS CloudFormation provisions and configures those resources for you.

AWS Chatbot supports AWS CloudFormation notifications through Amazon SNS topics. You enable support for SNS topics that are enabled for use with AWS Chatbot by selecting them in each AWS CloudFormation stack configuration. For more information, see Setting AWS CloudFormation Stack Options in the *AWS CloudFormation User Guide*.

## Notifications for AWS developer tools

AWS provides a suite of cloud-based developer tools for creating, managing, and working with software development projects. The AWS development tools suite includes AWS services such as AWS CloudFormation stacks, AWS CodeBuild, AWS CodeCommit, AWS CodeDeploy, AWS CodePipeline, and more. You can redirect Amazon SNS topic subscriptions for these services to AWS Chatbot. For example, if you want notifications about events in an AWS CodeCommit repository or in a pipeline in AWS CodePipeline to appear in a Microsoft Teams or Slack channel for your development teams, you can set up notifications for those resources in the Developer Tools console, and then integrate the SNS topic used for those notifications with AWS Chatbot. For more information, see Configure Integration Between Notifications and AWS Chatbot in the *Developer Tools Console User Guide*.

## Amazon CloudWatch alarms

To monitor performance and operating metrics for AWS services, and send notifications when thresholds are breached, you can create alarms in Amazon CloudWatch. CloudWatch sends an Amazon SNS notification or performs an action when the alarm changes state.

CloudWatch also features composite alarms. Composite alarms allow you to combine multiple alarms to reduce alarm noise and focus on critical operational issues. You can easily combine multiple alarms together into alarm hierarchies that only trigger once, when multiple alarms fire at the same time. Composite alarms are currently supported by AWS Chatbot.

> **Note**
>
> Parent composite alarms can have multiple triggering children however, the AWS Chatbot notification will only display a maximum of 3 of the total triggering metric children's alarm states. For example, if you have 10 total children alarms and 5 are currently triggered, the AWS Chatbot notification will display 3 of those 5.

Any metric, for any AWS service, that CloudWatch alarm actions can report can also be shared by an SNS topic to chat rooms through AWS Chatbot. This includes alarms for services such as Amazon Elastic Compute Cloud (Amazon EC2).

For information about setting up SNS topics to forward CloudWatch alarms, see Set Up Amazon SNS Notifications in the *Amazon CloudWatch User Guide*.

Because CloudWatch alarms use SNS topics to forward alarm notifications, you need to map only the associated Amazon SNS topic to your Slack channel or Amazon Chime webhook configuration in AWS Chatbot.

AWS Chatbot also supports several AWS services through CloudWatch Events. For more information, see the following section.

## Amazon EventBridge

AWS Chatbot supports multiple AWS services through Amazon EventBridge rules. EventBridge uses rules to help manage AWS service events and how you respond to them. You can use these rules to associate an Amazon SNS topic (or other actions) with an event type from any AWS service.

You map the Amazon SNS topic to the EventBridge rule, and then map it to a chat channel or Amazon Chime webhook in the AWS Chatbot console. When a service event matches the rule, the rule's target Amazon SNS topic sends an event to the AWS Chatbot for processing. The AWS Chatbot then sends a notification to the chat room. You can also customize the content of your notifications by using the custom notifications event schema and EventBridge InputTransformers. For more information, see Custom notifications in AWS Chatbot and Creating an EventBridge Rule that sends notifications to AWS Chatbot.

AWS Chatbot can process most AWS service events handled by EventBridge. This includes AWS Config, Amazon GuardDuty, AWS Health, AWS Security Hub, and AWS Systems Manager. AWS Chatbot only supports EventBridge events from AWS services. For an exhaustive list of supported service events, see EventBridge Event Examples from Supported AWS Services in the *EventBridge User Guide*.

> **Note**
>
> Event notifications from: CloudWatch Alarms, CodeBuild, CodeCommit, CodeDeploy, and CodePipeline are not currently supported via EventBridge rules. If you want to receive notifications for one of these services, you can go to its console, and configure Amazon SNS notifications that you can then map to your chat channel or Amazon Chime webhook configuration in AWS Chatbot. For more information, see Amazon CloudWatch alarms or Notifications for AWS developer tools.

## Tutorial: Creating an Amazon EventBridge rule that sends notifications to AWS Chatbot

AWS Chatbot currently supports notifications for most service events that are handled by Amazon EventBridge. When you create a rule for events, you tell EventBridge what action to take for events that match the rule. In this tutorial, you create a rule to use AWS Chatbot to generate an Amazon SNS topic notification that will appear in your Microsoft Teams channel, Slack channel or Amazon Chime chatroom.

> **Tip**
>
> You might create an EventBridge rule that unintentionally sends too many notifications to your chat channels. This typically happens with EventBridge rules that trigger on API calls via AWS CloudTrail. To better control the number of notifications you generate, write your EventBridge rules with event pattern based filtering. For more information about EventBridge event patterns, see Content-based filtering in Amazon EventBridge event patterns in the *EventBridge User Guide*.

## Prerequisites

For this tutorial, you need a Amazon Chime, Microsoft Teams, or Slack client for AWS Chatbot. For more information, see Getting started with AWS Chatbot in the *AWS Chatbot Administrator Guide*.

You also need to set up Amazon Simple Notification Service. Your Amazon SNS topic is used in the creation of your EventBridge rule and should be identical to the Amazon SNS topic used in your AWS Chatbot configuration. If you don't have any Amazon SNS topics yet, follow the steps in Getting Started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

For more information about EventBridge, see What Is Amazon EventBridge? in the *EventBridge User Guide*.

## Create an Amazon EventBridge Rule

Receiving notifications about events of interest in your Amazon Chime chat room, Microsoft Teams channel, or Slack channel is a convenient way to monitor service processes. In this tutorial, you create an EventBridge rule to use AWS Chatbot to generate an Amazon SNS topic notification that will appear in your chat channel or chat room. It is important to note that you only receive notifications from Amazon SNS topics that are used in your AWS Chatbot configuration.

**To create an EventBridge rule**

1. Open the EventBridge console.
2. In the navigation pane, choose **Rules**.

3. Choose **Create rule**.
4. Enter a name and description for the rule. A rule must have a unique name from other rules in the same Region.
5. Choose **Next**.
6. For **Event source**, choose **AWS events or EventBridge partner events**.
7. For **Creation method**, choose **Use pattern form**.
8. In **Event Pattern**, for **Event source**, choose **AWS services**.
9. For **AWS service**, select the service that emits the event.
10. For **Event type**, select the event you're interested in.

   ⚠ **Important**

   Choosing **All Events** as your service provider can result in increased costs and notifications.

   ⓘ **Tip**

   You can edit event patterns by choosing **Edit pattern** and then choosing **Save**.

   ⓘ **Note**

   Currently only AWS Services are supported.

11. Choose **Next**.
12. In **Target types**, select **AWS service**.
13. In **Select a target**, select **SNS Topic**.
14. For **Topic**, choose the appropriate topic.

   ⓘ **Note**

   This topic should be the same topic used in your AWS Chatbot configuration.

15. **(Optional) To create a custom notification**

    a. Choose **Additional settings**.

    b. In **Configure target input**, select **Input transformer**.

    c. Choose **Configure input transformer**.

    d. (Optional) In **Sample events**, search for an event type of interest and note the parameter names.

    e. In **Target input transformer**, enter your key-value pairs.

    f. In **Template**, enter a JSON object using the custom notifications event schema.

    g. (Optional) Choose **Generate output** to see what your custom notification will look like.

    h. Choose **Confirm**.

    > **Note**
    >
    > For more information on input transformers, see Amazon EventBridge input transformation in the *Amazon EventBridge User Guide*.

16. (Optional) To add additional targets, choose **Add target**.
17. Choose **Next**.
18. (Optional) Add tags.
19. Choose **Next**.
20. Choose **Create rule**.

After the rule is created, you can view, edit, or delete it in the console under **Rules**. When an event occurs that matches the rule, you receive a notification in your chat channel or Amazon Chime chat room from AWS Chatbot.

For information about testing your rule, see Test notifications from AWS services to chat channels using CloudWatch.

## Receive Amazon EventBridge event notifications between AWS accounts and Regions

You can receive EventBridge event notifications between AWS accounts and Regions in your chat channels and chat rooms using one AWS Chatbot and one Amazon SNS topic. To do this, use the Amazon SNS topic in your AWS Chatbot configuration as the target for your *receiver* account's EventBridge rule. With this mechanism, you don't have to configure AWS Chatbot in each AWS account you want to receive notifications from. If you have multiple accounts with multiple resources you want to monitor, you can configure AWS Chatbot in one account and have all other accounts send their events to the account with AWS Chatbot using EventBridge. For more information about sending and receiving events across accounts and Regions, see Sending and receiving EventBridge events between AWS accounts and Regions in the *Amazon EventBridge User Guide*.

## Delete an Amazon EventBridge rule

You can remove any resources created for this tutorial by navigating to the EventBridge console and deleting the resource.

**To delete or disable an EventBridge rule**

- To delete or disable an EventBridge rule, see Deleting or Disabling a Rule in the *EventBridge User Guide*.

# AWS Config

AWS Config performs resource oversight and tracking for auditing and compliance, config change management, troubleshooting, and security analysis. It provides a detailed view of AWS resources configuration in your AWS account. The service also shows how resources relate to one another and how they were configured in the past, so you can see how configurations and relationships change over time.

For AWS Config monitoring, you configure Amazon CloudWatch Events rules to forward AWS Config events notifications to an Amazon SNS topic. You can then map that topic to AWS Chatbot to track those event notifications in chat rooms.

For more information, see  Notifications for AWS Config in the *AWS Config Developer Guide*.

# Amazon GuardDuty

Amazon GuardDuty is a security threat monitoring service that detects and reports on potential security threats in your AWS account. It uses threat intelligence feeds, such as lists of malicious IPs and domains, and machine learning to identify possible unauthorized and malicious activity in your AWS environment.

GuardDuty reports its security incidents and threats through *findings*. Findings appear in the GuardDuty console and automatically appear as CloudWatch Events. You then create Amazon CloudWatch Events rules, so these events appear as notifications to a selected SNS topic. You then map that SNS topic to a chat channel or Amazon Chime webhook in AWS Chatbot.

For more information, see Monitoring Amazon GuardDuty Findings with Amazon CloudWatch Events in the *Amazon GuardDuty User Guide*.

## AWS Health

AWS Health provides visibility into the state of your AWS resources, services, and accounts. It provides information about the performance and availability of resources that affect your applications running on AWS and guidance for remediation. AWS Health provides this information in a console called the AWS Health Dashboard.

AWS Health directly supports EventBridge notifications. You configure CloudWatch Events rules for AWS Health, and specify an SNS topic mapped in AWS Chatbot.

For more information, see Monitoring AWS Health Events with Amazon CloudWatch Events in the *AWS Health User Guide*.

## AWS Security Hub

AWS Security Hub provides a comprehensive view of high-priority security alerts and compliance status across your AWS accounts. Security Hub aggregates, organizes, and prioritizes security findings from multiple AWS services, including Amazon GuardDuty, Amazon Inspector, and Amazon Macie. Security Hub reduces the effort of collecting and prioritizing security findings across accounts, from AWS services, and from AWS partner tools.

Security Hub supports two types of integration with CloudWatch Events rules, both of which AWS Chatbot supports:

- **Standard CloudWatch Events**. Security Hub automatically sends all findings to CloudWatch Events. You can define CloudWatch Events rules that automatically route generated findings to an Amazon Simple Storage Service (Amazon S3) bucket, a remediation workflow, or an SNS topic. Use this method to automatically send all Security Hub findings, or all findings with specific characteristics, to an SNS topic to which AWS Chatbot subscribes.

- **Security Hub Custom Actions**. Define custom actions in Security Hub and configure CloudWatch Events rules to respond to those actions. The event rule uses its SNS topic setting to forward its notifications to the SNS topic to which AWS Chatbot subscribes.

## AWS Systems Manager

AWS Systems Manager lets you view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances, and reporting or taking corrective action on detected policy violations.

AWS Chatbot supports the following Systems Manager events.

### Configuration compliance

- Status change for association compliance.
- Status change for instance patch compliance.

### Automation

- Status change for an automation execution.
- Status change for a single step in an automation execution.

### Run command

- Status change for a command (applies to one or more instances).
- Status change for a command invocation (applies to one instance only).

### State manager

- Status change for an association.
- Status change for an instance association.

**Parameter store**

- A parameter is created.
- A parameter is updated.
- A parameter is deleted.

For information about monitoring Systems Manager events with CloudWatch, see Monitoring Systems Manager Events with Amazon CloudWatch Events in the *AWS Systems Manager User Guide*.

## AWS Systems Manager Runbooks

SM runbooks define the actions that Systems Manager performs on your managed instances and other AWS resources when an automation runs. A runbook contains one or more steps that run in sequential order. The process of running these actions and their steps is called the automation. AWS Chatbot supports the ability to run SM runbooks directly from Microsoft Teams or Slack using CLI commands. You can type a command to list your runbooks and choose a runbook to run. Runbooks can require one or more input parameters before running (for example, Amazon EC2 instances can require inputs such as instance id). Once the runbook begins, it runs in its entirety. For an example of running a runbook using a CLI command, see Run an Automation runbook.

For more information about SM runbooks, see Working with runbooks in the *AWS Systems Manager User Guide*.

## AWS Systems Manager Incident Manager

AWS Systems Manager Incident Manager is an incident management console designed to help users mitigate and recover from incidents affecting their AWS-hosted applications. An incident is any unplanned interruption or reduction in quality of services.

AWS Chatbot allows you to communicate through chat channels and receive notifications and incident updates during an incident. You can also interact with the incident directly using chat commands. For more information, see Chat channels in the *Incident Manager User Guide*.

# Monitoring investigations with Amazon Q Developer in AWS Chatbot

You can use Amazon Q Developer operational investigations in your Microsoft Teams and Slack chat channels to investigate and identify the cause of application issues when they occur. An Amazon Q operational investigation traverses and analyzes volumes of data, such as logs, metrics, deployments, and configuration changes. It can then identify anomalies and the root cause of issues. Investigations can be initiated automatically from Amazon CloudWatch Alarm actions. When a root cause is identified, the assistant recommends high-confidence runbooks curated by AWS to help mitigate issues where applicable. You can update colleagues about the investigation using Jira and ServiceNow integration and one-click investigation status updates.

**Topics**

## Configuring AWS Chatbot with Amazon Q Developer operational investigations

To configure AWS Chatbot with Amazon Q operational investigations, you must add the following policies to enable two-way communication between investigations and AWS Chatbot. You can add these policies during step 2 of the AWS Chatbot channel configuration process for Slack and Microsoft Teams when you define your user permissions or by editing your configurations' **Permissions** in the AWS Chatbot console.

- Add **Notification permissions** and **Amazon Q operations assistant permissions** as policy templates when you define your user permissions. For more information about Channel role templates, see the section called "Role setting".
- Attach the **AIOpsOperatorAccess** managed IAM policy to your guardrail policies in AWS Chatbot. This grants permissions to AWS Chatbot to interact with Amazon Q operational investigations and perform required actions on your behalf.

## Connecting AWS Chatbot with an investigation group

You can integrate Amazon Q Developer operational investigations with your Microsoft Teams and Slack channels using Amazon SNS topics. Once integrated, you can receive and act on operational investigation notifications from your chat channel.

> **Tip**
>
> You can make investigations in your chat channels easier by adding custom actions to your notifications and by creating command aliases for frequently used tasks to fetch telemetry information.

**To connect AWS Chatbot with an investigation group**

1. Follow the steps in Get started with Amazon Q Developer operational investigations to create an investigation group.
2. Follow the steps in Integration with third-party chat systems to integrate Amazon Q operational investigations with your chat channel.

   > **Note**
   >
   > When selecting an Amazon SNS topic, select the same topic configured in your AWS Chatbot channel configuration.

# Customizing AWS Chatbot

Items you can customize include:

- Notifications
- Actions
- Command aliases
- Amazon Bedrock agents

**Topics**

## Custom notifications in AWS Chatbot

You can customize messages for your application events or customize default AWS service notifications in AWS Chatbot using custom notifications. By customizing notification content, you can promptly receive important application updates with relevant contextual information in your chat channels. This increases visibility for your team and facilitates quicker responses. You can also thread custom notifications using the `threadId` parameter shown in the section called “Parameter details”. Threaded notifications help you organize notification updates by grouping them in a thread in your chat channel.

> **Note**
>
> To thread notifications, your channel preferences must allow sending notification updates in a thread. You can update your channel preferences by entering `@aws preferences` in your chat channel.

You can generate custom notifications from Lambda functions, your applications, or by using Amazon EventBridge input transformers to modify existing EventBridge events into an AWS Chatbot compatible format. If using EventBridge, you map the Amazon SNS topic to the

EventBridge rule target and map the topic to a channel used in your AWS Chatbot configuration. Custom notifications use the same Amazon Simple Notification Service-based mechanisms as AWS Chatbot default notifications delivered in your chat channels. There are no additional costs to use custom notifications.

**Topics**

- Event schema
- Creating custom notification content in AWS Chatbot
- Sending a custom notification in AWS Chatbot
- Sample custom notifications
- OpenAPI schema

## Event schema

AWS Chatbot custom notifications must use the following event format:

```
{
    "version": String,
    "source": String,
    "id": String,
    "content": {
        "textType": String,
        "title": String,
        "description": String,
        "nextSteps": [ String, String, ... ],
        "keywords": [ String, String, ... ]
    },
    "metadata": {
        "threadId": String,
        "summary": String,
        "eventType": String,
        "relatedResources": [ String, String, ... ],
        "additionalContext" : {
            "customerProvidedKey1": String,
            "customerProvidedKey2": String
            ...
        }
    }
}
```

## Parameter details

| Parameter | Required | Description |
|---|---|---|
| `version` | Yes | Must be `1.0`. |
| `source` | Yes | Must be `custom`. |
| `id` | No | Unique event identifier. |
| `textType` | No | Only `client-markdown` textType is currently supported. When `textType=client-markdown` , AWS Chatbot renders notifications in the target chat platform' s markdown. For example, "Example text *123*" in Slack notification content would be formatted using Slack's markdown style. |
| `title` | No | Supports markdown content, including emojis and Slack @mentions using user IDs, for example @userID.<br>Maximum length is 250 characters. |
| `description` | Yes | Makes up the message content of your notification. Supports markdown content, including emojis and Slack @mentions using user IDs, for example @userID. |

| Parameter | Required | Description |
|---|---|---|
| | | Maximum length is 8,000 characters. |
| `nextSteps` | No | Used to communicate next step recommendations. Rendered as a bulleted list and supports markdown.<br>Each individual step in the `nextSteps` list has a maximum length of 350 characters. |
| `keywords` | No | Used to communicate event tags and categories. Rendered as an inline list of tags.<br>Each individual keyword in the `keywords` list has a maximum length of 75 characters. |
| `metadata` | No | Additional metadata about the event. |
| `threadId` | No | Used to thread messages in Slack and Microsoft Teams. Custom notifications with the same threadId value are grouped together. |
| `summary` | No | Displays a summary on the top-level message when notifications are threaded. |

| Parameter | Required | Description |
|---|---|---|
| `eventType` | No | Specifies the type of event for the custom notification (future release). |
| `relatedResources` | No | An array of strings describing resources related to the custom notification (future release). |
| `additionalContext` | No | A String-to-String dictionary used to provide additional information about your custom notification. |

**Note**

- `eventType` and `relatedResources` will be available in a future release.
- @mentions for Microsoft Teams aren't currently supported.

## Creating custom notification content in AWS Chatbot

Content created for custom notifications should utilize chat client specific syntax. For example, if you want text in your custom notification in Slack to be bold, use Slack markdown. For more information on Slack and Microsoft Teams markdown, see Format your messages and Use Markdown formatting in Microsoft Teams respectively.

You can add chat platform compatible emojis in your custom notifications. You can also tag team members using @mentions in your custom notifications for Slack. Tagged team members are notified when the custom notification is delivered to the chat channel. To tag a team member in Slack, use their user ID. For example, `<@userID>`. Tagging team members in Microsoft Teams isn't currently supported.

**Note**

All custom notifications contain the footer “# Custom notification delivered by AWS Chatbot.” to differentiate them from default notifications.

## Sending a custom notification in AWS Chatbot

You can create custom notification content and post the message to the Amazon SNS topic used in your AWS Chatbot configuration. AWS Chatbot monitors the Amazon SNS topic and sends the custom notification to your configured channel.

**Note**

You can also customize notifications generated by Amazon EventBridge by using AWS Chatbot's event schema in Input transformers. For more information, see Tutorial: Use input transformer to customize what Amazon EventBridge passes to the event target in the *EventBridge User Guide*.

**To send a custom notification to a chat channel**

1. Open the Amazon SNS console.
2. Select the topic used in your AWS Chatbot configuration.
3. Choose **Edit**.
4. Ensure your topic is standard.
5. Choose **Save changes**.
6. Choose **Publish message**.
7. In **Message body**, enter a JSON object using the custom notifications event schema.
8. Choose **Publish message**.
9. View your custom notification in your chat channel.

# Sample custom notifications

## Minimalist custom notification

The following custom notification example creates a simple notification that contains only mandatory schema parameters when published to an Amazon SNS topic.

```
{
    "version": "1.0",
    "source": "custom",
    "content": {
        "description": ":warning: EC2 auto scaling refresh failed for ASG
 *OrderProcessorServiceASG*! \ncc: @SRE-Team"
    }
}
```

## Complex custom notification

The following custom notification example creates a complex notification when published to an Amazon SNS topic.

```
  {
    "version": "1.0",
    "source": "custom",
    "id": "c-weihfjdsf",
    "content": {
      "textType": "client-markdown",
      "title": ":warning: Banana Order processing is down!",
      "description": "Banana Order processor application is no longer processing
 orders. OnCall team has beeen paged.",
      "nextSteps": [
        "Refer to <http://www.example.com|*diagnosis* runbook>",
        "@googlie: Page Jane if error persists over 30 minutes",
        "Check if instance i-04d231f25c18592ea needs to receive an AMI rehydration"
      ],
      "keywords": [
        "BananaOrderIntake",
        "Critical",
        "SRE"
      ]
    },
    "metadata": {
      "threadId": "OrderProcessing-1",
      "summary": "Order Processing Update",
      "eventType": "BananaOrderAppEvent",
      "relatedResources": [
        "i-04d231f25c18592ea",
        "i-0c8c31affab6078fb"
      ],
      "additionalContext": {
        "priority": "critical"
      }
    }
  }
```

## OpenAPI schema

If you use OpenAPI, you can use the following OpenAPI schema to generate custom notification resources for use in source code.

```
openapi: 3.0.0
info:
  title: Custom Notifications Payload Schema
  version: 1.0.0

components:
  schemas:
    CustomNotifications:
      type: object
      required:
        - version
        - source
        - content
      properties:
        version:
          type: string
          description: Must be "1.0"
          enum:
            - "1.0"
        source:
          type: string
          description: Must be "custom"
          enum:
            - "custom"
        id:
          type: string
          description: Unique event identifier
        content:
          type: object
          required:
            - description
          properties:
            textType:
              type: string
              description: Only "client-markdown" textType is currently supported.
 When textType=client-markdown, AWS Chatbot renders notifications in the target chat
 platform's markdown. For example, "Example text 123" in Slack notification content
 would be formatted using Slack's markdown style.
              enum:
                - "client-markdown"
            title:
              type: string
              description: Supports markdown content, including emojis and @mentions.
            description:
              type: string
              description: Makes up the message content of your notification. Supports
 markdown content, including emojis and @mentions.
            nextSteps:
              type: array
              description: Used to communicate next step recommendations. Rendered as a
 bulleted list and supports markdown.
              items:
                type: string
            keywords:
              type: array
              description: Used to communicate event tags and categories. Rendered as
 an inline list of tags.
              items:
                type: string
        metadata:
          type: object
          properties:
            threadId:
              type: string
              description: Used for threading messages for chat clients that support
 it. Custom notifications with the same threadId value will be grouped together.
            summary:
              type: string
```

```
              description: Used for displaying a summary on the top-level message when 
notifications are threaded.
            eventType:
              type: string
              description: Specifies the type of event for the custom notification. 
(Future release)
            relatedResources:
              type: array
              description: An array of strings describing resources related to the 
custom notification. (Future release)
              items:
                type: string
            additionalContext:
              type: object
              description: A String-to-String dictionary customers can use to provide 
additional information about their custom notification.
              additionalProperties:
                type: string
```

# Custom actions in AWS Chatbot

Custom actions are preconfigured buttons you add to custom and default notifications. These actions allow you to automate commonly used DevOps processes and incident response tasks. When you create a custom action, you configure your action button to run either a CLI command or a Lambda function. Action targets can be paramaterized by using the parameters available in your notification metadata. You can use custom actions to retrieve telemetry information, run an automation runbook, and notify team members. When an issue arises, you can take action directly from your notifications. Custom actions are available in AWS Chatbot configurations for Microsoft Teams and Slack. Custom actions can't be created using AWS CDK at this time.

No additional permissions are needed to configure or run custom actions. When your channel members choose the custom action button, the action target is invoked using the configured IAM permissions in your channel configuration.

**Topics**

- Creating a custom action using AWS Chatbot
- Sample use cases

# Creating a custom action using AWS Chatbot

You can create custom actions using Lambda functions in your account, using the AWS Chatbot API, or by entering AWS CLI commands. AWS CLI command Shortcuts function similarly to Command aliases.

Lambda action

**To create a Lambda action**

1. Choose the vertical ellipsis button on the bottom of a notification in your chat channel.
2. In **Manage actions**, choose **Create**.
3. Enter a custom action name. This name is a unique identifier for your custom action.
4. Enter a name for your custom action button. This name is shown on a button on your notification. This name should be 20 characters or less and can incorporate emojis.
5. For **Custom action type**, select **Lambda action**.
6. Choose **Next**.
7. Select an AWS account.
8. Select a Region.

   > **Note**
   >
   > The available Lambda functions are populated from this account and Region.

9. Choose **Load Lambdas**.
10. In **Define Lambda Function**, select a Lambda function.
11. Enter your payload.

    > **Note**
    >
    > We expose predefined variables from your custom notifications as useable variables.

12. (Optional) Add more variables by choosing **+ Add more variables**.
13. Choose **Next**.
14. (Optional) Add additional display criteria:

   a. Select a variable.

   b. Select a condition.

   c. Choose **Add**.

15. Choose **Save**.

### AWS CLI command action

**To create an AWS CLI command action**

1. Choose the vertical ellipsis button on the bottom of a notification in your chat channel.

2. In **Manage actions**, choose **Create**.

3. Enter a custom action name. This name is a unique identifier for your custom action.

4. Enter a name for your custom action button. This name is shown on a button on your notification. This name should be 20 characters or less and can incorporate emojis.

5. For **Custom action type**, select **CLI action**.

6. Enter a command.

   > **Note**
   >
   > We expose predefined variables from your custom notifications as useable variables.

7. (Optional) Add more variables by choosing **+ Add more variables**.

8. Choose **Next**.

9. (Optional) Add additional display criteria:

   a. Select a variable.

   b. Select a condition.

   c. Choose **Add**.

10. Choose **Save**.

# Sample use cases

This page includes examples of functional use cases for custom actions that you can leverage for your specific needs.

## Custom notifications metadata

You can use custom notifications to specify metadata for your custom actions. Custom actions display this information as variables that you define in the payload. Before you run a custom action, you're shown a complete summary of that action and its payload. In the following example, a custom notification for allow listing an IP address contains an IP address as metadata:

```
{
    "version": "1.0",
    "source": "custom",
    "content": {
        "title": "IP Address Allowlist Request",
        "description": "We have received a request to allows list an IP address from
 'sample@sample.com'."
    },
    "metadata": {
        "additionalContext": {
            "IPAddress": "192.168.0.1"
        }
    }
}
```

If you create a custom action based on this notification, `{"message": "I'ved allow listed IP address: $IPAddress"}` is shown as an available notification variable that you can use in your payload. For example, the following Lambda action payload displays the message and the IP address variable.

```
{"message": "I'ved allow listed IP address: $IPAddress"}
```

## Recent Amazon CloudWatch Logs errors

The following example shows how you can use custom actions to display recent errors from an Amazon CloudWatch Logs group in your channel from an Amazon CloudWatch Logs notification.

The following Lambda function returns a list of the most common Amazon CloudWatch Logs errors:

```
import boto3
from collections import Counter
import json
import datetime

def extract_message(value):
  if value.startswith('{'):
    try:
      structured_message = json.loads(value)

      return structured_message.get('message', value)
    except Exception:
      pass

  return value

def take_ellipsis(value, length):
  if len(value) <= length:
    return value
  else:
    return value[:length - 1] + '…'

def lambda_handler(event, context):
  log_group_name = event['logGroup']
  lookback_minutes = int(event.get('minutes', 60))
  filter = event.get('filter', 'ERROR')
  limit = int(event.get('limit', 10))

  logs_client = boto3.client('logs')

  now = datetime.datetime.now()
  offset = now - datetime.timedelta(minutes=lookback_minutes)
  print(offset)
  start_time = int(offset.timestamp() * 1000)
  end_time = int(now.timestamp() * 1000)

  response = logs_client.filter_log_events(
    logGroupName=log_group_name,
    startTime=start_time,
    endTime=end_time,
    filterPattern=filter
  )
```

```
 messages = [extract_message(event['message']) for event in response['events']]

 message_counts = Counter(messages)

 top_messages = message_counts.most_common(limit)

 if top_messages:
   formatted_messages = [
     f'{index + 1}. `{take_ellipsis(message[0], 100)}` ({message[1]} occurrences)' for
index, error in enumerate(top_messages)
   ]
   message_summary = '\n'.join(formatted_messages)

   return f'*Most common errors in `{log_group_name}` within the last hour*\n\n' +
message_summary
 else:
   return f'Found no errors matching filter within the last {lookback_minutes} minutes
in {log_group_name}'
```

You can create a Lambda action that invokes this Lambda function and view errors for specific log groups by choosing this function while creating your action and entering the following as the payload:

```
{
    "logGroup": "$LogGroup"
}
```

# Creating and using command aliases in chat channels

AWS Chatbot supports the creation of command aliases to make running commonly used CLI commands easier. A command alias is a custom shortcut or shorthand representation of a CLI command that you define. A command alias can be configured to include one or more custom parameters. Additional target actions can be included as part of the alias definition.

Command aliases are defined for a single channel. If a channel is configured to work with more than one AWS account, the aliases work across all AWS accounts. All channel members share the aliases defined in the channel.

> **Note**
>
> When a channel member runs a command alias, the target is run with the configured member's permissions (channel role, user role) depending on the permissions schemas in place. For more information about permissions, see Understanding AWS Chatbot permissions.

We strongly recommend that you don't add any personally identifiable information (PII) or other confidential or sensitive information to your command aliases.

For more information about running CLI commands in Slack, see Running AWS CLI commands from chat channels using AWS Chatbot.

**Topics**

- Creating a command alias in AWS Chatbot
- Listing command aliases using AWS Chatbot
- Running command aliases using AWS Chatbot
- Getting help using AWS Chatbot

## Creating a command alias in AWS Chatbot

You can create command aliases using the CreateCustomAction action and providing an `AliasName`, or by running:

`@aws alias create` *`alias_name mapped_action`*.

> **Note**
>
> The command alias definition can contain additional parameters. Also note that *`alias_name`* has a 100 character limit and *`mapped_action`* has a 5,000 character limit.

For example, in this alias:

```
@aws alias create automation ssm start-automation-execution
--document-name $name --parameters { "AutomationAssumeRole":
["arn:aws:iam::123456789012:role/$role-name"] }
```

The parameters `$name` and `$role-name` are input variables that are required to run the alias. When the alias is run, your supplied parameter values are used for the parameters.

So, if you run:

`@aws run automation val1 val2`.

`$name` is assigned the value val1 and `$role-name` the value of val2. These values are assigned by position.

Alternatively, you can explicitly name the alias parameters:

`@aws run automation --name val1 --role-name val2`.

## Listing command aliases using AWS Chatbot

To list all available command aliases in a channel, run:

`@aws alias list`.

## Running command aliases using AWS Chatbot

To run a command alias, use:

`@aws run` *`alias_name`* or `@aws alias run` *`alias_name`*.

## Getting help using AWS Chatbot

To get help with command aliases, run:

`@aws help` or `@aws alias help`.

# Invoking Amazon Bedrock Agents from chat channels in AWS Chatbot

You can invoke Amazon Bedrock Agents directly from your chat channels using AWS Chatbot. Amazon Bedrock Agents are fully managed capabilities that make it easier for you to create generative AI-based applications that can complete complex tasks for a wide range of use cases and deliver up-to-date answers based on proprietary knowledge sources. You can use agents to increase productivity, improve your customer experience, and automate workflows. For more information, see Automate tasks in your application using conversational agents in the *Amazon Bedrock User Guide*.

**Topics**

## Setting up your chat channel

To invoke an agent from your chat channel with AWS Chatbot, you must ensure that both:

- The channel or user role has permission to invoke the agent.
- The channel guardrail policies include permission to invoke the agent.

> **Tip**
>
> The most minimal way to achieve this is to create a new IAM policy with just the required `InvokeAgent` permissions and add it to your channel guardrail policies.

The AWS Chatbot Bedrock Agent connector requires the `bedrock:InvokeAgent` IAM action.

### Example role policy

The following policy is a minimal policy statement with the permissions required to invoke a specific agent. This policy statement is added to the channel or user role used to run AWS SDK commands in your AWS Chatbot channel.

```
{
  "Sid": "AllowInvokeBedrockAgent",
  "Effect": "Allow",
  "Action": "bedrock:InvokeAgent",
  "Resource": [
    "arn:aws:bedrock:aws-region:111122223333:agent-alias/AGENT12345/ALIAS12345"
  ]
}
```

**Note**

The Agent and Alias IDs are located on the Agent details page in the Amazon Bedrock console. For more information, see [View information about aliases of agents in Amazon Bedrock](#) in the *Amazon Bedrock User Guide*.

## Connecting Amazon Bedrock to chat channels

To connect an agent to a chat channel, you must set up a connector. Connector commands are used to manage connectors in your chat channel. AWS Chatbot connectors work by assigning a friendly name to a specific alias of an Amazon Bedrock agent. Once this configuration is defined, you can start conversing with this agent directly in your chat channel.

**To connect an agent to your chat channel**

1. In your chat channel, enter `@aws connector`. AWS Chatbot responds with a help message, listing the available connector management commands.
2. Add your Amazon Bedrock agent by running `@aws connector add` *`connector_name`* `arn:aws:bedrock:aws-region:111122223333:agent/`*`AgentID AliasID`*.
3. (Optional) View all registered connectors by running `@aws connector list`.
4. (Optional) Remove connectors by running `@aws connector delete` *`connector_name`*.

**Important**

- Exposing agents to chat channels: By granting a chat channel or user role permission to invoke an agent, they can perform any action that the agent provides. Take careful consideration when exposing agents to chat channels.
- Shared usage of Amazon Bedrock sessions: The threads that connectors start are visible to all members of the the chat channel, and any member can participate in the session

(user roles permitting). Consider this when exposing actions that depend on previous session context, as subsequent requests can be from different users.

- Amazon Bedrock session timeouts: Agents have a configurable idle session TTL after which any session data is forgotten. This is true for interactions with agents within a thread and if the idle session TTL is exceeded, previous contents of the conversation are lost. Further interactions with the agent within this thread maintain the same session ID but may lose the previous session context.
- Amazon Bedrock feature coverage: Connectors in AWS Chatbot are currently intended for use in textual conversations with agents. The use of the **Return Control** features in action groups for richer controls is currently not supported. Any trace context, observations, or linked sources are also currently not included in the final message. You can let us know if these features would be desirable by sending us feedback in your chat channel.

## Conversing with your Amazon Bedrock Agent connectors using AWS Chatbot

To start a conversation with your agent, run:

`@aws ask` *`connector_name your message`*. This invokes your configured agent with your message within a new session. Your agent's response starts a new thread in your chat channel under the initial message.

Any subsequent mention of `@aws` in this thread sends the provided message directly to the agent and all interactions in this thread share the same agent and session ID. As such, you can continue to ask questions in this thread without specifying the name of your connector.

## Updating a connector using AWS Chatbot

If you're publishing a new alias for a Amazon Bedrock, you must replace the connector to converse with this new version. Existing threads can no longer use the old agent alias. You may also need to update the roles and policies for your channel to allow the new alias.

**To update a connector**

1. In your chat channel, run `@aws connector delete` *`connector_name`*.
2. Run `@aws connector add` *`connector_name`* `arn:aws:bedrock:aws-region:111122223333:agent/`*`AgentID AliasID`* `.`

## Quotas for Amazon Bedrock in AWS Chatbot

Your AWS account has the following default quotas, formerly referred to as limits, for Amazon Bedrock.

| Name | Default | Adjustable | Description |
|---|---|---|---|
| Connectors per channel | 10 connectors. | No | The maximum number of registered connectors you can have in a channel. |
| InvokeAgent request timeout | 60 seconds. | No | The amount of time connectors allow the invokeAgent request to complete |
| InvokeAgent response truncation | 2,500 characters. | No | The maximum number of characters a response message from an agent can contain. |
| Connector names | 1-20 letters, digits, dashes or underscores | No | The maximum number of characters a valid connector name can contain. Each connector must have a unique name. |

# Performing actions

Actions you can perform include:

- Running CLI commands
- Asking questions about AWS services and troubleshooting guidance
- Finding your AWS resources
- Responding to interactive messages
- Getting help
- Managing your AWS Support cases
- Running CLI commands to use different AWS services from chat channels
  - For example, you can retrieve diagnostic information, invoke Lambda functions, and create support cases for your AWS resources

**Topics**

## Running AWS CLI commands from chat channels using AWS Chatbot

You can run commands using AWS CLI syntax directly in chat channels. AWS Chatbot enables you to retrieve diagnostic information, configure AWS resources, and run workflows.

When you interact with AWS Chatbot in your chat channels, it prompts you for any missing parameters before it runs the command.

**Note**

To perform actions in your chat channels, you must first have the appropriate permissions. For more information about AWS Chatbot's permissions, see Understanding permissions.

**Topics**

## Non-supported operations in AWS Chatbot

AWS Chatbot doesn't support running commands for operations in the following JSON policy:

```
{
  "Statement": [
    {
      "Action": [
        "appsync:ListApiKeys",
        "chatbot:*",
        "codecommit:GetFile",
        "codecommit:GetCommit",
        "codecommit:GetDifferences",
        "cognito-idp:*",
        "cognito-identity:*",
        "connect:GetFederationToken",
        "dynamodb:BatchGetItem",
        "dynamodb:GetItem",
        "ec2:GetPasswordData",
        "ecr:GetAuthorizationToken",
        "ecr:GetLogin",
        "gamelift:RequestUploadCredentials",
        "gamelift:GetInstanceAccess",
        "identitystore:*",
        "lightsail:DownloadDefaultKeyPair",
        "lightsail:GetInstanceAccessDetail",
        "lightsail:GetKeyPair",
        "lightsail:GetKeyPairs",
        "lightsail:UpdateRelationalDatabase",
        "iam:*",
        "kms:*",
        "redshift:GetClusterCredentials",
        "sdb:*",
        "secretsmanager:*",
        "sso:*",
        "sso-admin:*",
        "sso-oidc:*",
        "storagegateway:DescribeChapCredentials",
        "sts:*",
        "s3:GetObject",
        "s3:HeadObject",
        "s3:PutObject",
        "s3:GetBucketPolicy",
        "snowball:GetJobUnlockCode"
      ],
      "Effect": "Deny",
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

## Using commands in AWS Chatbot

After you set up the AWS Chatbot, you run commands with the following prefix:

```
@aws
```

> **Note**
>
> If you are using Slack and AWS is not listed as a valid member of the channel, you need to add the AWS Chatbot app to the Slack workspace and invite it to the channel. For more information, see the Getting started guide for AWS Chatbot.

The AWS Chatbot command syntax is the same as you would use in a terminal:

```
@aws service command --options
```

**Note**

> You can specify parameters with either a double hyphen (`--option`) or a single hyphen (`-option`). This allows you to use a mobile device to run commands without running into issues with the mobile device automatically converting a double hyphen to a long dash.

**Note**

> AWS CLI commands run from AWS Chatbot have an execution timeout of 15 seconds. If a command response is not received within 15 seconds, you receive a timeout error message. If you have longer running jobs, such as AWS Lambda functions, you should invoke them asynchronously from AWS Chatbot. The maximum allowable Lambda function execution timeout is 900 seconds (15 minutes). For more information about asynchronous invocation, see Asynchronous invocation in the *AWS Lambda Developer Guide*.

For example, enter the following read-only command to view a list of your Lambda functions:

```
@aws lambda list-functions
```

Enter the following commands to list and chart CloudWatch alarms:

```
@aws cloudwatch describe-alarms --state ALARM
```

You can also use CLI commands to change you AWS resources. For example, enter the following command to change your Kinesis shards:

```
@aws kinesis update-shard-count --stream-name samplestream --scaling-type
UNIFORM_SCALING --target-shard-count 6
```

You can enter a complete AWS CLI command with all the parameters, or you can enter the command without parameters and AWS Chatbot prompts you for missing parameters.

For more information on commonly used CLI commands, see Using CLI commands with AWS Chatbot - Common use cases. For an exhaustive list of CLI commands, see the AWS CLI Command Reference.

**Note**

If you find you are unable to run commands, you may need to switch your user role or contact your administrator to find out what actions are permissible.

The following limitations apply to running AWS CLI commands in your chat rooms:

- You may experience some latency when invoking commands through AWS Chatbot.
- Regardless of their AWS Chatbot role permissions, users cannot run IAM, AWS Security Token Service, or AWS Key Management Service commands within chat channels.
- Amazon S3 service commands support Linux-style command aliases such as **ls** and **cp**. AWS Chatbot does not support Amazon S3 command aliases for commands in Slack.
- Users cannot display or decrypt secret keys or key pairs for any AWS service, or pass IAM credentials.
- You can't use AWS CLI command memory (that is, recent commands appear when the user presses up arrow or down arrow keys) in the chat channel. You must enter, or copy and paste each AWS CLI command in the chat channel.
- You can create AWS support cases through your chat channels. You cannot add attachments to these cases from the chat channel.
- Chat channels do not support standard AWS CLI pagination.

## Running commands using AWS Chatbot

AWS Chatbot tracks your use of command options and prompts you for any missing parameters before it runs the command you want.

For example, if you enter `@aws lambda get-function` with no further arguments, the Chatbot requests the function name. Then, run the `@aws lambda list-functions` command, find the function name you need, and re-run the first command with the corrected option. Add more parameters for the initial command with `@aws function-name` *`name`*. AWS Chatbot parses your commands and helps you complete the correct syntax so it can run the complete AWS CLI command.

**Topics**

- Formatting data and viewing logs in AWS Chatbot
- Displaying Amazon CloudWatch Logs information using AWS Chatbot
- Creating an AWS Support case using AWS Chatbot

## Getting help for AWS services in AWS Chatbot

To get help about commands for any AWS service, enter **@aws** followed by the service name, as shown following:

```
@aws lambda --help
```

```
@aws cloudwatch describe-alarms --help
```

## Formatting data and viewing logs in AWS Chatbot

To ensure data from Amazon CloudWatch alarms is correctly formatted, attach the **Lambda-Invoke Command Permissions** and **ReadOnly Commands Permissions** IAM policies to the role in the AWS Chatbot console for users in the chat channel.

Run the `cloudwatch describe-alarms` command to show CloudWatch alarms in chart form as follows:

```
@aws cloudwatch describe-alarms
```

You can change the command to only include notifications in the alarm state, filtering out other notifications, by adding the following option:

```
@aws cloudwatch describe-alarms --state ALARM
```

To see alarms from a different AWS Region, include that Region in the command:

```
@aws cloudwatch describe-alarms --state ALARM --region us-east-1
```

You can also filter AWS CLI output by using the optional `query` parameter. A query uses JMESPath syntax to create an expression to filter your output to your specifications. For more information about filtering, see Filtering AWS CLI output in the *AWS Command Line Interface User Guide*. For more information about JMESPath syntax, see their website. The following example shows how to limit AWS CLI output for the `cloudwatch describe-alarms` command to just the alarm name, description, state, and reason attributes.

```
@AWS cloudwatch describe-alarms --query
 @.{MetricAlarms:MetricAlarms[*].
 {AlarmName:AlarmName, AlarmDescription:AlarmDescription, StateValue:StateValue,
 StateReason:StateReason, Namespace:Namespace, MetricName:MetricName,
 Dimensions:Dimensions, ComparisonOperator:ComparisonOperator, Threshold:Threshold,
 Period:Period, EvaluationPeriods:EvaluationPeriods, Statistic:Statistic}}
 --region us-east-2
```

## Displaying Amazon CloudWatch Logs information using AWS Chatbot

CloudWatch alarm notifications show buttons in chat client notifications to view logs related to the alarm. These notifications use the CloudWatch Log Insights feature. There may be service charges for using this feature to query and show logs.

You can view CloudWatch logs, including error logs, that are associated with the CloudWatch alarm by choosing **Show logs** at the bottom of the alarm notification. AWS Chatbot displays the first 30 log entries from the start of the alarm evaluation period. AWS Chatbot uses CloudWatch Log Insights to query for logs. The query results contain a link to the CloudWatch Log Insights console, where a user can dive deeper into logs details.

Choose **Show error logs** to filter search results to log entries containing Error, Exception, or Fail terms.

The log shows a command that a user can copy, paste, and edit to re-run the query for viewing logs.

## Creating an AWS Support case using AWS Chatbot

The **AWS Support Command Permissions** policy appears in the AWS Chatbot console when you configure resources. It's provided in the AWS Chatbot console so that you can set up new roles for users in your chat client to create AWS support tickets through their chat channels.

You can quickly create a new AWS support case by entering the following:

```
@aws support create-case
```

Follow the prompts from AWS Chatbot to fill out the support case with its needed parameters. When you complete the case information entry, AWS Chatbot asks for confirmation. You will not be able to use file attachments.

**Note**

AWS Chatbot requires `UpperCamelCase` for the `--query` parameter. In `UpperCamelCase`, the first letter of every word is capitalized.

For any AWS Chatbot role that creates AWS Support cases, you need to attach the **AWS Support command permissions** policy to the role. For existing roles, you will need to attach the policy in the IAM console.

In the IAM console, this policy appears as **AWSSupportAccess**.

It is an AWS managed policy. Attach this policy in IAM to any role for AWS Chatbot usage. You can define your own policy with greater restrictions, using this policy as a template.

The **Support Command Permissions** policy applies only to the AWS Support service.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "support:*"
            ],
            "Resource": "*"
        }
    ]
}
```

## Configuring commands support on an existing chat channel in AWS Chatbot

If you have existing chat channels using the AWS Chatbot, you can reconfigure them in a few steps to support the AWS CLI.

1. Open the AWS Chatbot console.

2. In the **Configured Clients** page, select the chat client. If you have only one, its contents (the list of chat channels) appear on the page.

   > **Note**
   >
   > In this procedure, we assume use of an existing AWS Chatbot chat channel configuration. The process is very similar if you need to create a new chat client configuration by choosing **Configure new client**.

3. Choose a channel from the **Configured channels** list, and choose **Edit**. The selected channel can be public or private.

4. Define your **Role setting** by choosing a **Channel role** or **User roles**. For more information about role types, see Role setting:

   Channel role

   1. For **Role setting**, choose **Channel role**.
   2. For **Channel role**, choose **Create new role**. If you want to use an existing role instead, choose **Use an existing role**. To use an existing IAM role, you will need to modify it for use with AWS Chatbot. For more information, see Configuring an IAM Role for AWS Chatbot.
   3. For **Role name**, enter a name. Valid characters: a-z, A-Z, 0-9, .\w+=,.@-_.
   4. For **Role policy template**, choose **Read Only command permissions** and **Lambda-Invoke command permissions**.

      > **Note**
      >
      > - If you plan to have users of the role submit AWS Support cases, also attach the **AWS Support command permissions** policy.
      > - If you want the role to allow users to manage incidents, add the **Incident Manager Permissions** policy.

   User roles

   1. For **Role setting**, choose **User roles**.

5. Select the policies that will make up your channel guardrail policies. Your channel guardrail policies control what actions are available to your channel members.

   **Note**

   If you initially had permission to run Lambda invoke, it is contained in **All actions permitted**.

   **Note**

   To run most CLI commands from your Slack channel, ensure you select **All actions permitted**.

   **Note**

   You do not need to edit or change the Amazon SNS topics configuration for the chat channel.

6. Choose **Save**.

   You can use the IAM console to modify an existing IAM role. By simply attaching the three additional AWS Chatbot policies to the IAM role, users of that role can immediately begin using commands in the chat channel. To do so, see Configuring an IAM Role for AWS Chatbot.

**Important**

If you have a large number of chat channels and you want to have the same command permissions across multiple channels, you can apply the configured AWS Chatbot role to any of your other chat channels without further modification. The IAM policies will be consistent across chat channels that support commands in your AWS Chatbot service.

## Enabling multiple accounts to use commands in AWS Chatbot

You can configure AWS Chatbot for multiple AWS accounts in the same chat channel. When you work with AWS Chatbot for the first time in that channel, it will ask you which account you want to use. AWS Chatbot will remember the account selection for 7 days.

To change the default account in the channel, enter `@aws set default-account` and select the account from the list.

# Asking questions from chat channels in AWS Chatbot

You can search and discover information about AWS services and your AWS resources by asking AWS Chatbot natural language questions. AWS Chatbot answers service related questions directly in your chat channels by providing relevant AWS documentation and support article excerpts. AWS Chatbot answers resource related questions by using AWS Resource Explorer to search and find your resources.

You can also get Artificial Intelligence (AI) powered answers to your questions about AWS services and your AWS resources by using Amazon Q Developer. Amazon Q Developer is a generative AI-powered conversational assistant that can help you to understand, build, extend, and operate AWS applications. To accelerate your building on AWS, the model that powers Amazon Q is augmented with high-quality AWS content to produce more complete, actionable, and referenced answers. For more information, see What is Amazon Q Developer? in the *Amazon Q Developer User Guide*. Amazon Q is available in Slack and Microsoft Teams channels configured with AWS Chatbot.

**Topics**

- Asking Amazon Q questions in AWS Chatbot
- Asking AWS Chatbot questions in chat channels

## Asking Amazon Q questions in AWS Chatbot

You can ask Amazon Q AWS service and AWS resource related questions in your Microsoft Teams and Slack chat channels. Amazon Q must be enabled to answer questions. We recommend using Amazon Q for AWS resource related queries.

**Topics**

- Enabling Amazon Q Developer in AWS Chatbot

- Asking Amazon Q AWS service questions in AWS Chatbot
- Asking Amazon Q AWS resource questions

## Enabling Amazon Q Developer in AWS Chatbot

You can get AI powered answers to your questions about AWS and your AWS account resources by activating Amazon Q in your Slack and Microsoft Teams channels configured with AWS Chatbot. You can ask Amazon Q

**To get started with Amazon Q**

1. Add the AmazonQDeveloperAccess managed policy to your IAM role:

   > **Note**
   >
   > If you require administrator access, you can use the AmazonQFullAccess managed policy.

   a. Open the IAM console.
   b. In the navigation pane of the IAM console, choose **Roles**.
   c. Choose the name of the role you want to modify.
   d. In **Permissions policies**, choose **Add permissions** and **Attach policies**.
   e. Enter `AmazonQDeveloperAccess` in the search.
   f. Select **AmazonQDeveloperAccess**.
   g. Choose **Add permissions**.
2. Add the `AmazonQDeveloperAccess` managed policy to your channel guardrails:

   a. Open the AWS Chatbot console.
   b. Choose a configured client.
   c. Select a configured channel.
   d. Choose **Set guardrails**.
   e. Enter `AmazonQDeveloperAccess` in the search.
   f. Select **AmazonQDeveloperAccess**.
   g. Choose **Save**.

3. In your chat channel, enter `@aws` and your question.

## Asking Amazon Q AWS service questions in AWS Chatbot

If Amazon Q is enabled in AWS Chatbot, you can chat about best practices, recommendations, step-by-step instructions for AWS tasks, and architecting your AWS resources and workflows. Additionally, Amazon Q can generate short scripts or code snippets to help you get started using AWS SDKs and AWS CLI. To enable Amazon Q in AWS Chatbot, see ???. For more information, see Chatting with Amazon Q Developer about AWS in the *Amazon Q Developer User Guide*.

### Commonly asked questions

You can ask Amazon Q these service questions directly from your chat channels.

```
@aws what is fargate?
```

```
@aws what’s the maximum runtime for a Lambda function?
```

```
When should I put my resources in a VPC?
```

```
@aws what’s the best container service to use to run my workload if I need
to keep my costs low?
```

```
@aws how do I list my Amazon S3 buckets?
```

## Asking Amazon Q AWS resource questions

If Amazon Q is enabled, you can ask it about your AWS account resources. Amazon Q can perform get, list, and describe actions to retrieve your AWS resources. Amazon Q can’t answer questions about the data stored in your resources, such as listing objects in an Amazon S3 bucket, or questions related to your account security, identity, credentials, or cryptography. To enable Amazon Q in AWS Chatbot, see ???. For more information, see Chatting about your resources in the *Amazon Q Developer User Guide*.

We recommend using Amazon Q to answer your resource related queries. Alternatively, you can use Resource Explorer for multi-region resource based queries. If Amazon Q is disabled, we attempt to answer your question using resource explorer based search. If you have both Resource Explorer and Amazon Q enabled, Amazon Q is used to answer your resource questions.

**Note**

Amazon Q chat makes calls from the US East (N. Virginia) Region. Amazon Q may need to make cross-Region calls to access your resources in other Regions, including opt-in Regions. Amazon Q requires cross-Region consent to retrieve resources. For more information, see Cross-Region consent in the *Amazon Q Developer User Guide*.

**Commonly asked questions**

You can ask Amazon Q these resource questions directly from your chat channels.

```
@aws get the configuration for my lambda function <name>?
```

```
@aws what is the size of the auto scaling group <name> in us-east-2?
```

```
@aws can you show ec2 instances running in us-east-1?
```

## Asking AWS Chatbot questions in chat channels

You can ask AWS Chatbot AWS service and AWS resource related questions in your Microsoft Teams and Slack chat channels.

**Topics**

- Asking AWS Chatbot AWS service questions
- Asking AWS Chatbot AWS resource questions in chat channels

### Asking AWS Chatbot AWS service questions

You can ask AWS Chatbot questions about AWS services to find relevant product information directly in your chat channels. For example, you can ask:

```
@aws How do I set ec2 autoscaling group's max limit?
```

If AWS Chatbot finds an answer, it presents an excerpt with a direct link to the information source.

### Asking AWS Chatbot AWS resource questions in chat channels

AWS Chatbot uses Resource Explorer to search and discover your resources. When appropriate, AWS Chatbot shows its search results in a list. This list shows the top 5 matching resources and includes the ability to filter results further by resource type, AWS Region, and tag.

> **Note**
>
> We recommend using Amazon Q for AWS account resource related questions. For more information, see ???. Alternatively, you can use Resource Explorer for multi-region resource based queries. If Amazon Q is disabled, we attempt to answer your question using resource explorer based search. If you have both Resource Explorer and Amazon Q enabled, Amazon Q is used to answer your resource questions.

### Prerequisities

To ask AWS Chatbot resource related questions, you must:

- Sign up for Resource Explorer to allow AWS Chatbot to search your resources. For more information, see Getting started with Resource Explorer in the *AWS Resource Explorer User Guide*. To search your resources across Regions from chat channels, we recommend turning on cross-Region search by creating an aggregator index. For more information, see Turning on cross-Region search by creating an aggregator index in the *AWS Resource Explorer User Guide*.

  > **Note**
  >
  > You can also create Resource Explorer resources using an AWS CloudFormation template. For more information, see Creating Resource Explorer resources with AWS CloudFormation in the *AWS Resource Explorer User Guide*.

- Make sure you have active indexes and views with at least one default view in your Region. Indexes and views allow Resource Explorer to catalog and query your resources. For more information, see Terms and concepts for Resource Explorer in the *AWS Resource Explorer User Guide*.
- Add the `AWSResourceExplorerReadOnlyAccess` policy to your channel role or reach appropriate user role, depending on your channel's permissions scheme.
- Verify that your channel guardrail policies allow `AWSResourceExplorerReadOnlyAccess` permissions.

### Commonly asked questions

These questions can be asked directly from your chat channels. Replace the *`italicized placeholder text`* in these examples with your own information.

```
@aws What services am I using in Region?
```

```
@aws What are the resources in my account with tags?
```

```
@aws What lambda functions do I have?
```

## Getting help from AWS Chatbot

You can ask AWS Chatbot for help by entering `@aws help`. You can choose any of the following buttons to receive additional information about their respective topics.

**Topics**

- Home (#)
- Commands
- Aliases
- Built-ins
- Q&A
- More help

### Home (#)

Choosing the home icon (#) returns you to the main menu.

### Commands

Provides helpful tips about CLI command syntax and parameters.

For example, you dont have to remember parameters to use CLI commands with AWS Chatbot. AWS Chatbot prompts you for all required prameters for a CLI command. Commands with one parameter don't require a parameter flag and unique AWS service operations don't require you to enter a service name. For more information, see the section called “Running AWS CLI commands”.

### Aliases

Provides helpful tips about command aliases. Command aliases are short-hand representations of CLI commands. For more information, see the section called “Command aliases”.

# Built-ins

Lists and provides details about AWS Chatbot commands.

## Setting preferences

To manage AWS Chatbot preferences for your chat channel, enter `@aws set preferences`.

You can manage your threading preferences and communication preferences after running this command. This includes where AWS Chatbot notifications are displayed, how frequently new threads are created, and what AWS Chatbot related updates you want to receive.

## AWS accounts

To see a list of AWS accounts configured for this channel or select a new account to use for commands, enter `@aws set-default account`.

If you would like to add an account to the channel, sign-in to the AWS Chatbot console with that account and configure AWS Chatbot for your channel. For more information, see the section called “Setting up AWS Chatbot”.

## Providing feedback

To provide feedback to AWS Chatbot, enter `feedback` *`your comment`*.

## Switching user roles

To access a link for mapping user roles, enter `@aws switch-roles`.

If your current user role doesn’t have the right permissions, you can switch roles directly from your chat channel. For more information, see ???.

# Q&A

Provides additional information about the types of questions AWS Chatbot can answer and how to use Amazon Q Developer with AWS Chatbot. For more information, see the section called “Asking questions from chat channels”.

## More help

To get help with services or operations, use the `--help` parameter. For example, you can enter `@aws ec2 --help` to get help with EC2. For more information, see the section called “Running AWS CLI commands”.

# Managing AWS Support cases from chat channels in AWS Chatbot

You can use AWS Chatbot to monitor and respond to your AWS Support cases in your Microsoft Teams and Slack chat channels. A support case is a way for you to connect with technical support and get help with AWS service-related technical issues. You can use the on-screen action buttons to interact with your support cases. Actions you can perform include viewing correspondence (case history), resolving your case, and replying to your case. You can create support cases using AWS Chatbot and the AWS Management Console. For more information, see ??? and Creating support cases and case management in the *AWS Support User Guide* respectively.

AWS Support case management in AWS Chatbot is available at no additional cost in Regions where AWS Chatbot is offered.

> **Note**
>
> To interact with support cases in chat channels, you must have a Business, Enterprise On-Ramp, or Enterprise Support plan. If you attempt to take action on a support case without one of these plans, you will receive a `SubscriptionRequiredException` error message. For information about changing your support plan, see AWS Support.

## Prerequisities

To manage your support cases in your chat channels, you must:

- Create an Amazon EventBridge rule for AWS Support case events and choose an Amazon SNS topic as your target. For more information, see Creating an EventBridge rule for AWS Support cases in the *AWS Support User Guide*.
  - Subscribe that Amazon SNS topic to your AWS Chatbot configuration. For more information, see ???.

- Add the managed role `AWSSupportAccess` to your AWS Chatbot role. For more information, see ???.

# Using CLI commands with AWS Chatbot - Common use cases

Common use cases for using AWS Chatbot in your chat channels involve running CLI commands. This topic also includes an example use case for invoking a Lambda function in your chat channel, written in Python 3.8.

For more information about running CLI commands in chat channels see Running AWS CLI commands from chat channels using AWS Chatbot.

For a tutorial that walks you through how to invoke Lambda functions from AWS Chatbot, see the Tutorial: Using AWS Chatbot to run an AWS Lambda function remotely.

**Topics**

## Restart an Amazon EC2 instance

The following example shows how CLI commands can be used to restart your specified Amazon EC2 instance from your chat channel. The parameters you include here are your instance id, min, and max size. For more information about restarting Amazon EC2 instances, see the reboot-instances command in the *AWS CLI Command Reference*.

```
@aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

## Change Auto Scaling limits

The following example shows how CLI commands can be used to change your Auto Scaling limits directly from your chat channel. The parameters you include are the name of your Auto Scaling group and the minimum and maximum sizes. For more information about changing autoscaling limits, see update-autoscaling-group command in the *AWS CLI Command Reference*.

```
@aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg
--min-size 2 --max-size 10
```

## Run an Automation runbook

The following example shows how CLI commands can be used to run an Automation runbook. In this command you specify your document name and your parameters. For more information, see the start-automation-execution command in the *AWS CLI Command Reference*.

```
@aws ssm start-automation-execution --document-name "AWS-UpdateLinuxAmi"
--parameters "AutomationAssumeRole=arn:aws:iam::123456789012:role/
SSMAutomationRole,SourceAmiId=ami-
EXAMPLE,IamInstanceProfileName=EC2InstanceRole"
```

## Use a Lambda function to approve an AWS CodePipeline action

The code example in this section demonstrates how you can use a Lambda function to perform activities, spefically how you can manually approve a pipeline action. This function enables you to approve or reject a pipeline action from your chat channel by entering the status and a summary. The function gets the required token using the get_pipeline_status method. It then uses the token value when applying the approval decision by using the put_approval_result method. For more information about these methods, see the CodePipeline section of the *Boto3 docs 1.14.10 documentation*. The Lambda code uses Python 3.8.

```
import boto3

def lambda_handler(event, context):
    client = boto3.client('codepipeline')

    getToken = client.get_pipeline_state(
        name = 'mypipeline1'
    )

    myToken=getToken['stageStates']['actionStates']['latestExecution']['token']

    response = client.put_approval_result(
        pipelineName='mypipeline1',
        stageName='beta',
        actionName='Approval',
        result={
            'summary': ['summary'],
            'status': ['status']
        },
        token=myToken

    )
```

The following AWS Chatbot command invokes the Lambda function. For more information about CodePipeline and pipeline actions, see the *AWS CodePipeline User Guide*.

```
@aws invoke mypipeline1-beta-Approval --payload {"summary": "the design looks good, ready to release",“status”: "Approved"}
```

# Tutorial: Using AWS Chatbot to run an AWS Lambda function remotely

In this tutorial you use AWS Chatbot to run a Lambda function remotely and check the status of the Lambda function using Amazon CloudWatch. A Lambda function is a self contained block of organized resuable code that you write. Lambda functions are useful because they are run without provisioning or managing servers. Additionally, they are only invoked when needed based on your specifications. There are steps at the end of this tutorial to delete the resources you created.

**Topics**

- Prerequisites
- Step 1: Create a Lambda function
- Step 2: Create an SNS topic
- Step 3: Configure a CloudWatch alarm
- Step 4: Configure a Slack client for AWS Chatbot
- Step 5: Invoke a Lambda function from Slack
- Step 6: Test the CloudWatch alarm
- Step 7: Clean up resources

## Prerequisites

This tutorial assumes that you have some familiarity with the Lambda, AWS Chatbot, and CloudWatch consoles.

For more information, see the following topics:

- Getting started with AWS Lambda in the *AWS Lambda Developer Guide*.
- Setting up AWS Chatbot in the *AWS Chatbot Administrator Guide.*
- Getting Set Up with CloudWatch in the *Amazon CloudWatch User Guide.*

The AWS Region that you select while setting up these consoles should be the same Region you specify in your Slack channel when your first AWS Command Line Interface (AWS CLI) command in Step 5: Invoke a Lambda function from Slack.

## Step 1: Create a Lambda function

In this procedure you create a Lambda function in the console and test it.

**To create a Lambda function**

1. Sign in to the AWS Management Console and open the Lambda console at console.aws.amazon.com/lambda.
2. Choose **Create function**.
3. Choose **Author From Scratch**.
4. In **Function Name**, enter: **`myHelloWorld`**
5. Choose **Create Function**.
6. Copy and paste the following example code into `index.js`.

```
export const handler = async (event) => {
  // TODO implement
  const response = 'Hello World!'
  return response;
};
   };
```

7. Choose **Deploy**.
8. Choose **Test**.
9. In **Event Name**, enter: **`myHelloWorld`**
10. Choose **Save**.

11. Choose **Test** and then verify that the **Execution results** tab displays Response: "Hello World!"

## Step 2: Create an SNS topic

CloudWatch uses Amazon SNS to send notifications. First, you create an SNS topic and subscribe to it using your email. Later in the tutorial you use this SNS topic to configure AWS Chatbot.

### To create an SNS topic

1. Open the Amazon SNS console.
2. In the left navigation pane, choose **Topics**.
3. Choose **Create Topic**.
4. Create a topic with the following settings:
   a. **Type** – Standard
   b. **Name** – **`myHelloWorldNotifications`**
   c. **Display name** – **`myHelloWorld`**
5. Choose **Create topic**.
6. Choose **Create subscription**.
7. Create a subscription with the following settings:
   a. **Protocol** – **`Email`**
   b. **Endpoint** – Your email address
8. Confirm subscription to the SNS by checking your email and choosing the link.

## Step 3: Configure a CloudWatch alarm

A CloudWatch alarm monitors your Lambda function and sends a notification if an error occurs.

### To create a CloudWatch alarm

1. Open the CloudWatch console.
2. Choose **Alarms**.
3. Choose **Create alarm**.
4. Choose **Select metric**.

5. Choose **Lambda**.
6. Choose **By Function Name**.
7. Choose **myHelloWorld errors**.
8. Change the following settings:

    a. **Period – 1 minute**

    b. **Whenever Errors is `Greater` than `0`**

    c. **Send notifications to – `myHelloWorldNotifications`**

    d. **Alarm name – `myHelloWorld-alarm`**

    e. **Alarm description – `Lambda myHelloWorld alarm`**
9. Choose **Create alarm**.

## Step 4: Configure a Slack client for AWS Chatbot

You can configure a Slack client using AWS Chatbot to to run different commands in Slack using the AWS CLI. In this tutorial you use AWS CLI to invoke your Lambda function from Slack.

### To create a Slack client

1. Open the AWS Chatbot console.
2. Under **Configure a Chat client** choose **Slack**, and then choose **Configure**.

    > ⚠ **Important**
    >
    > When you choose **Configure**, you are momentarily navigated away from the AWS Chatbot console.
3. In the upper right corner, choose the dropdown list, and then choose the Slack workspace that you want to use with AWS Chatbot.

    > ⓘ **Note**
    >
    > There's no limit to the number of workspaces that you can set up for AWS Chatbot, but you must set up each workspace one at a time.
4. Choose **Allow**.

5. Choose **Configure new channel**.
6. Under **Configuration details**, for **Name**, enter **`myHelloWorld`**.
7. Under **Channel type**, choose **Private**.

    a. Navigate to Slack and create a private channel by choosing the **+** button to the right of **Channels**.

    b. Choose **Create a channel**.

    c. Name the channel **`myHelloWorld`**.

    d. Choose to make the channel private.

    e. Choose **Create**.

    f. When prompted to add people, choose **x**.

    g. Navigate back to the AWS Chatbot console and enter the private channel ID.
8. Define the **IAM permissions** that the chatbot uses for messaging your Slack chat room as shown following:

    a. For **Role name**, enter **`myHelloWorldRole`**.

    b. For **Policy Templates**, select **Read-only command permissions** and **Lambda-invoke command permissions.**
9. In the SNS topics section, choose the appropriate AWS Region under **Region**.
10. Under **Topics**, select the **myHelloWorldNotifcations** topic.
11. Choose **Configure**.

## Step 5: Invoke a Lambda function from Slack

After you configure a chatbot in AWS Chatbot, you can invoke Lambda functions from Slack using AWS CLI syntax. To interact with AWS Chatbot in Slack, enter **`@aws`** followed by an AWS CLI command. For more information, see Running AWS CLI commands from chat channels using AWS Chatbot in the *AWS Chatbot Administrator Guide.*

**To invoke a Lambda function**

1. Invite AWS Chatbot to your channel by doing the following in Slack:

    a. Enter `@AWS`.

    b. Choose **Invite to Channel.**

> **Tip**
>
> You only have to invite AWS Chatbot to the channel once.
>
> If AWS is not listed as a valid member of the channel, you need to add the AWS Chatbot app to the Slack workspace. For more information, see the Getting started guide for AWS Chatbot.

2. Enter the following command in Slack:

```
@aws lambda invoke --function-name myHelloWorld --region <your region>
```

> **Important**
>
> Replace *<your region>* with the same AWS Region you set while using the Lambda, CloudWatch, and AWS Chatbot consoles. You only need to specify the AWS Region in the channel once when you type your first AWS CLI command in Slack.

> **Tip**
>
> AWS Chatbot also supports certain simplified AWS CLI syntaxes. For example, the simplified version of the previous command is shown following:
>
> ```
> @aws invoke myHelloWorld --region <your region>
> ```

3. Choose **Yes**.

4. The following output is shown:

```
ExecutedVersion: $LATEST
Payload: \"Hello World\"
StatusCode: 200
```

## Troubleshooting

If you try to run your Lambda function in Slack and you encounter errors referring to the following permissions, revisit step 8 of the Step 4: Configure a Slack client for AWS Chatbot procedure and verify that you have the correct permissions assigned to your role:

- **Lambda-invoke command permissions**
- **Read-only command permissions**

## Step 6: Test the CloudWatch alarm

In this step, you update the myHelloWorld function so that it returns an error, which triggers the CloudWatch alarm. By testing the alarm you can confirm that it's configured correctly and that you can view CloudWatch alarms in Slack (in addition to logs).

### To test the CloudWatch alarm

1. Open the Lambda console Functions page.
2. Choose **myHelloWorld**.
3. Copy and paste the following example code into the Lambda function code:

```
exports.handler = async (event) => {
    throw new Error('this is an error');
};
```

4. Choose **Deploy** and confirm your changes have been deployed by viewing the label next to the **Deploy** button.
5. Return to your Slack channel and then enter the following command:

```
@aws invoke myHelloWorld
```

6. An error appears in your output, and you receive a CloudWatch alarm notification in Slack and an email. It might take a few minutes for you to receive the notifications.
7. To view logs, choose **Show logs** or **Show error logs**.

### Troubleshooting

If you don't receive a notification in Slack or an email from CloudWatch, navigate to the CloudWatch console and on the left of the screen. Under **Alarms**, choose **In alarm** to confirm that your alarm has triggered. Your alarm name should appear on this page if it has been triggered successfully.

## Step 7: Clean up resources

You can remove any resources created for this tutorial that you don't want to keep by navigating to the specific service's console and deleting the resource. Removing unwanted or unused resources is beneficial because it lowers overall costs to you.

### To delete the Lambda function

1. Open the Lambda console.
2. Choose **myHelloWordFunction**.
3. Choose **Actions** and then choose **delete**.

### To delete the CloudWatch alarm

1. Open the CloudWatch console.
2. In the left navigation pane, choose **Insufficient**.
3. Choose myHelloWorld-alarm by selecting the check box.
4. Choose **Actions** and then choose **delete**.

### To delete the AWS Chatbot configuration

1. Open the AWS Chatbot console.
2. Choose **Slack.**
3. Choose the radio button next to the channel you created and then choose **Delete**.

# Tagging your AWS Chatbot resources

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value, both of which you define. Tags help you manage, search for, and filter resources.

Tags help you categorize your AWS resources in different ways. For example, you can tag your resources by purpose, owner, or environment. This is useful when you have many resources of the same type. You can quickly identify a specific resource based on the tags you assigned to it. You can assign one or more tags to your AWS resources. Each tag has an associated value.

We recommend that you create a set of tag keys that meet your needs for each resource type. Use a consistent set of tags to more efficiently manage your AWS resources. You can search and filter the resources based on the tags you add.

Tags are interpreted strictly as a string of characters. They aren't automatically assigned to your resources. You can edit tag keys and values, as well as remove tags from a resource, at any time. You can set the value of a tag to an empty string. However, you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the previous value. If you delete a resource, any tags for the resource are also deleted.

## Managing tags

Tags consist of the `Key` and `Value` properties on a resource. You can use the AWS Chatbot console, the AWS CLI, or the AWS Chatbot API to add, edit, or delete the values for these properties. For more information about working with tags, see the following:

- TagResource, UntagResource, and ListTagsForResource API actions.
- TagResource, UntagResource, and ListTagsForResource in the *AWS Chatbot CLI Reference*.
- Using Tag Editor in the *Resource Groups User Guide.*

## Tagging restrictions

The following basic restrictions apply to tags.

| Restriction | Description |
|---|---|
| Maximum number of tags per resource | 50 |

| Restriction | Description |
|---|---|
| Maximum key length | 128 Unicode characters in UTF-8 |
| Maximum value length | 256 Unicode characters in UTF-8 |
| Prefix restriction | Don't use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix don't count against the number of tags you can assign to a resource. |
| Character restrictions | Tags may only contain Unicode letters, digits, white space, or these symbols: `_ . : / = + - @` |

# Monitoring AWS Chatbot

Monitoring is an important part of maintaining the availability of AWS Chatbot and your other AWS solutions. AWS provides the following monitoring tools to watch AWS Chatbot, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

## Monitoring AWS Chatbot with Amazon CloudWatch

You can monitor AWS Chatbot using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

### Enabling CloudWatch Metrics

Amazon CloudWatch metrics are enabled by default.

### Available metrics and dimensions

The metrics and dimensions that AWS Chatbot sends to Amazon CloudWatch are listed below.

The `AWS/Chatbot` namespace includes the following metrics.

> **Note**
>
> To get AWS Chatbot metrics, you must specify **US East (N. Virginia)** for the Region.

| Metric | Description | |
|---|---|---|
| `EventsThrottled` | The number of throttled notifications.<br>Events may be throttled if the number of events received exceeds 10 per second.<br>Units: Count | |
| `EventsProcessed` | The number of event notifications received by AWS Chatbot.<br>Units: Count | |
| `UnsupportedEvents` | The number of unsupported events or messages attempted.<br>For a full list of AWS services supported by AWS Chatbot, see *Monitoring AWS services*.<br>Units: Count | |
| `MessageDeliverySuccess` | The number of messages successfully delivered to the chat client.<br>Units: Count | |
| `MessageDeliveryFailure` | The number of messages that failed to deliver to the chat client.<br>Units: Count | |

AWS Chatbot sends the following dimensions to CloudWatch.

| Dimension | Description |
|---|---|
| `ConfigurationName` | This dimension filters the data you request by the name of your configuration. |

## Viewing AWS Chatbot metrics

You can view metrics in the CloudWatch console, which provides a fine-grained and customizable display of your resources, as well as the number of running tasks in a service.

### Viewing AWS Chatbot metrics in the CloudWatch console

AWS Chatbot metrics can be viewed in the CloudWatch console. The CloudWatch console provides a detailed view of AWS Chatbot metrics, and you can tailor the views to suit your needs. For more information about CloudWatch, see the Amazon CloudWatch User Guide.

**To view metrics in the CloudWatch console**

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Metrics** section in the left navigation, choose **AWS Chatbot**.
3. Choose the metrics to view.

# Accessing Amazon CloudWatch Logs for AWS Chatbot

AWS provides event logging with Amazon CloudWatch Logs. With CloudWatch Logs for AWS Chatbot, you can see all the events handled by AWS Chatbot. You can also see details of any error that may have prevented a notification from appearing in your Amazon Chime or Slack chat room.

Possible errors that you can see with CloudWatch Logs include lack of permissions, unsupported events, and events throttled by the chat client. For more information about these errors, see *Troubleshooting*.

AWS Chatbot also provides an audit log of commands executed by AWS Chatbot in CloudWatch Logs. With CloudWatch Logs' audit log events for AWS Chatbot, you can see an audit log of executed commands and their chat workspace ID, channel ID, and channel user ID attributes. The audit log events in CloudWatch Logs are always enabled and can't be disabled.

AWS Chatbot always logs audit events for command execution to CloudWatch Logs. You can choose to enable logging for all events, or only for errors.

> **Note**
>
> There is an additional charge for using CloudWatch Logs. For more details, see Amazon CloudWatch Pricing.

## Enabling CloudWatch Logs

You can enable CloudWatch Logs during the setup flow of your Amazon Chime, Microsoft Teams, or Slack channel configuration. For existing channels, you can edit the configuration to enable logging.

**To enable CloudWatch Logs for a new configuration**

1. On the **Configure channel** page, during the setup flow, under **Configuration details**, choose **Send logs to CloudWatch.**
2. Choose either **All events** or **Errors only**.
3. Continue the setup flow, then choose **Configure channel**.

**To enable CloudWatch Logs for an existing configuration**

1. In the AWS Chatbot console, under **Configured clients**, navigate to the chat client you want to edit.
2. From the list of existing configurations, choose the configuration you want to edit, then choose **Edit**.
3. On the **Edit** page, choose **Send logs to CloudWatch.**
4. Choose either **All events** or **Errors only**.
5. Choose **Save**.

## Viewing CloudWatch Logs

Your AWS Chatbot logs will be sent to CloudWatch under a designated CloudWatch Logs group for your configuration. The group name is **`/aws/chatbot/configuration-name`**. To learn more about log groups and other CloudWatch concepts such as log events and log streams, see Amazon CloudWatch Logs Concepts in the *Amazon CloudWatch Logs User Guide*.

You can view your logs in the Amazon CloudWatch console. Note that you must specify **US East (N. Virginia)** for the Region. For more information, see View Log Data Sent to CloudWatch Logs in the *Amazon CloudWatch Logs User Guide*.

## Logging AWS Chatbot API calls with AWS CloudTrail

AWS Chatbot integrates events with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Chatbot. CloudTrail captures API calls for AWS Chatbot as events. The calls captured include calls from the AWS Chatbot console and code calls to the AWS Chatbot API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Chatbot. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Chatbot, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Viewing Events with CloudTrail Event History.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

When you create a *trail*, you can enable continuous delivery of AWS Chatbot events to an Amazon S3 bucket that you specify. The trail logs events from all Regions in the AWS partition for that service and delivers the log files to that Amazon S3 bucket. You can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following topics in the *AWS CloudTrail User Guide*:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

## Logging AWS Chatbot API information in CloudTrail

Every event log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or for a federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

## Logging other AWS API information in CloudTrail

When you use commands in AWS Chatbot that call APIs from other AWS services, those APIs are logged in CloudTrail as well.

When you run a command that involves another AWS service, AWS Chatbot assumes an IAM role in your account to invoke AWS APIs on your behalf. These APIs appear in your CloudTrail events, and they are associated with the role that was configured for your AWS Chatbot configuration with a session name that includes **chatbot**, such as **chatbot-session**.

Because AWS Chatbot is a global service, it may process your events in a different AWS Region. An API call is logged in the region where the resource behind that API call exists. For example, if you run a `lambda list-functions` command in AWS Chatbot, CloudTrail will log two APIs: **AssumeRole** and **ListFunctions**. The **AssumeRole** call is logged in the region AWS Chatbot processed it in, and the **ListFunctions** call is logged in the region the function exists in.

## Example: AWS Chatbot log file entries

CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, user identification, and more. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry for the AWS Chatbot `DescribeSlackChannels` action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:jdoe",
        "arn": "arn:aws:sts::111122223333:assumed-role/user/jdoe",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-08-01T17:24:13Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/user",
                "accountId": "111122223333",
                "userName": "jdoe"
            }
        }
    },
    "eventTime": "2019-08-01T23:16:02Z",
    "eventSource": "chatbot.amazonaws.com",
    "eventName": "DescribeSlackChannels",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "10.24.34.3",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.137-0.1.ac.218.74.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "SlackTeamId": "XXXXXXXX",
        "MaxResults": 1000
    },
    "responseElements": null,
    "requestID": "543db7ab-b4b2-11e9-8925-d139e92a1fe8",
    "eventID": "5b2805a5-3e06-4437-a7a2-b5fdb5cbb4e2",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

The following example shows a CloudTrail log entry for a `DescribeSlackWorkspaces` action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:jdoe",
        "arn": "arn:aws:sts::111122223333:assumed-role/user/jdoe",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-08-07T16:11:27Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/user",
                "accountId": "111122223333",
                "userName": "jdoe"
            }
        }
    },
    "eventTime": "2019-08-07T17:46:26Z",
    "eventSource": "chatbot.amazonaws.com",
    "eventName": "DescribeSlackWorkspaces",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "10.24.34.3",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.137-0.1.ac.218.74.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "476570da-b93b-11e9-af41-a744734236af",
    "eventID": "3f061095-b488-43d4-becc-f8652d459ac5",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

# Security in AWS Chatbot

At AWS, cloud security is our highest priority. As an AWS customer, you benefit from a data center and network architecture that we build 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 our security effectiveness as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS Chatbot, 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 AWS Chatbot. The following topics show you how to configure AWS Chatbot to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Chatbot resources.

**Topics**

# Data protection in AWS Chatbot

The AWS shared responsibility model applies to data protection in AWS Chatbot. 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. You are also responsible for 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 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. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 140-3 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) 140-3.

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 AWS Chatbot 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.

We also strongly recommend that you don't provide secrets or confidential data to AWS Chatbot. All commands run using AWS Chatbot come through Slack. As such, we recommend that you carefully consider what information you send AWS Chatbot through Slack. Additionally, all comamnds run using AWS Chatbot can be processed in any commercial AWS Region.

**Note**

AWS Chatbot doesn't modify any event, alarm, or other reporting data when it forwards Amazon Simple Notification Service (Amazon SNS) notifications to chat rooms. It treats all notifications as read only.

# Data privacy in AWS Chatbot

## AWS Chatbot AI powered improvements

By default AWS Chatbot stores and processes user data such as AWS Chatbot configurations, notifications, user inputs, AWS Chatbot generated responses, and interaction data. This data helps AWS Chatbot continuously improve and develop both itself and Artificial Intelligence (AI) technologies. Your data is not shared with any third parties and is protected using sophisticated controls to prevent unauthorized access and misuse. AWS Chatbot doesn't store or use interactions in chat channels with Amazon Q for training AWS Chatbot's AI technologies.

If you would like to opt out of sharing your user data, you can toggle **Share user data** from the **Data privacy** page in your **Account settings**. When you opt out, your data is removed from all AWS Regions. This includes any previously stored data.

# Identity and Access Management for AWS Chatbot

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 AWS Chatbot resources. IAM is an AWS service that you can use with no additional charge.

## Audience

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

**Service user –** If you use the AWS Chatbot service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Chatbot 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 AWS Chatbot, see Troubleshooting AWS Chatbot identity and access.

**Service administrator** – If you're in charge of AWS Chatbot resources at your company, you probably have full access to AWS Chatbot. It's your job to determine which AWS Chatbot 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 AWS Chatbot, see How AWS Chatbot works with IAM.

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Chatbot. To view example AWS Chatbot identity-based policies that you can use in IAM, see Identity-based policies for AWS Chatbot.

# How AWS Chatbot works with IAM

Before you use IAM to manage access to AWS Chatbot, you should understand which IAM features are available to use with AWS Chatbot. The following subsections introduce each IAM capability supported by AWS Chatbot, point you to further information about how to use them, and describe the IAM capabilities that AWS Chatbot doesn't support. To get a high-level view of how AWS Chatbot and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

For an overview of IAM and its features, see Understanding How IAM Works in the *IAM User Guide*.

**Topics**

## Identity-based policies and AWS Chatbot

AWS Chatbot supports the use of IAM identity-based policies for service usage and management.

An AWS Identity and Access Management (IAM) *policy* is a document that defines the permissions that apply to an IAM user, group, or role. The permissions determine what users can do in AWS. A policy typically allows access to specific actions, and can optionally grant that the actions are allowed for specific resources, like Amazon Simple Notification Service (Amazon SNS) notifications. Policies can also explicitly deny access.

*Identity-based policies* are attached to an IAM user, group, or role (identity). These policies let you specify what that AWS identity can do (its permissions). For example, you can attach an identity policy to the IAM user named adesai, to allow that user to perform the AWS Chatbot `DescribeSlackChannels` action.

For information about, and examples of, using identity-based policies with AWS Chatbot, see AWS Chatbot Identity-Based Policies.

For more general information about how IAM identity-based policies work, see Identity vs. Resource in the *IAM User Guide*.

## Resource-level permissions and AWS Chatbot

*Resource-level permissions* are JSON policy statements that specify the AWS resources on which associated IAM entities can perform actions.  You define a resource-level permission in an IAM policy, then attach the policy to a user's AWS account or to any other IAM entity. The users then have permission to access that resource. Resource-level permissions differ from IAM *resource-based policies* because you attach complete resource-based policies directly to an AWS resource.

When you customize IAM policies for users to work with the AWS Chatbot service, one of your primary options for policy editing is to configure resource-based permissions for your policies.

AWS Chatbot supports resource-level permissions, but not resource-based policies.

For more information about how IAM resource-level permissions work with AWS Chatbot, see IAM Resource-Level Permissions for AWS Chatbot.

## Condition keys and AWS Chatbot

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*.

AWS Chatbot doesn't define any service-specific condition keys. It supports global condition keys. To see all actions and resources for which AWS Chatbot can use global condition keys, see Actions, Resources, and Condition Keys for AWS Chatbot in the *IAM User Guide*. For more information about AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

## Using temporary credentials with AWS Chatbot

You can use temporary credentials to sign in with federation, assume an IAM role, or assume a cross-account role. You obtain temporary security credentials by calling AWS Security Token Service (AWS STS) API operations, such as AssumeRole or GetFederationToken.

AWS Chatbot supports using temporary credentials. For more information about defining and using temporary IAM credentials, see Temporary Security Credentials in the *IAM User Guide*.

## Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but can't edit the permissions for service-linked roles.

## Service roles

AWS Chatbot supports service roles.

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might prevent the service from functioning as expected.

### 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.

- **AWS Organizations 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*.
- **IAM account settings** - With IAM, you can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. When you activate STS endpoints for a Region, AWS STS can issue temporary credentials to users and roles in your account that make an AWS STS request. Those credentials can then be used in any Region that is enabled by default or is manually enabled. You must activate the Region in the account where the temporary credentials are generated. It does not matter whether a user is signed into the same account or a different account when they make the request. For more information, see Activating and deactivating AWS STS in an AWS Region in the *IAM User Guide*.

**Note**

AWS Chatbot is a global service that requires access to all AWS Regions. If there is a policy in place that prevents access to services in certain Regions, you must change the policy to allow global AWS Chatbot access.

## IAM policies for AWS Chatbot

This section describes the IAM permissions and policies that AWS Chatbot uses to secure its operations with other AWS services. AWS Chatbot uses these permissions to safely forward

Amazon SNS notifications to chat rooms, support AWS CLI commands sessions in Slack, invoke Lambda functions, and create AWS support tickets directly in the Slack console. You can also define your own custom policies for the same purposes, using these policies as templates.

When you create a new role in the AWS Chatbot console, any of the IAM policies described in this topic might be assigned to that role. You could apply all of them to a single role, or choose only a couple of them based on how the users of that role will use the AWS Chatbot. Some policies contain a superset of permissions of other policies.

**Topics**

- AWS managed IAM policies in AWS Chatbot
- Customer managed IAM policies in AWS Chatbot

## AWS managed IAM policies in AWS Chatbot

AWS Chatbot supports the following AWS managed IAM policies:

- **ReadOnlyAccess**
- **CloudWatchReadOnlyAccess**
- **AWS Support Command Permissions Policy**
- **AmazonQFullAccess**
- **AIOpsOperatorAccess**

AWS managed policies are available to all AWS Chatbot users, but you can't change or edit them. You can copy them and use them as templates for your own policies, knowing that you are using AWS-approved policy language to build your own policies.

AWS Chatbot adheres to standard IAM practices for using admin IAM accounts to activate and use the AWS Chatbot service.

As a convenience, AWS Chatbot also supports the creation of new IAM roles directly in the AWS Chatbot console. However, to configure existing IAM entities to use AWS Chatbot, you need to use the IAM console.

### The IAM ReadOnlyAccess policy

The **ReadOnlyAccess** policy is an AWS managed policy that is automatically assigned to roles in the AWS Chatbot service.

This policy does not appear in the AWS Chatbot console. It defines Get, List, and Describe permissions for the entire suite of AWS services, enabling AWS Chatbot to use this role to access any of those services on your behalf.

You can attach this policy to new roles in IAM, or use it as a template to define your own, more restrictive policy.

> **Note**
>
> AWS Chatbot must use a role that defines all the read-only permissions necessary for its usage. You can define a policy to be more restrictive or specify fewer services than the policy described here, and use that in place of the **ReadOnlyAccess** policy. However, you must ensure that all CloudWatch and Amazon SNS read-only permissions remain in your policy, or some CloudWatch features may not work with AWS Chatbot. The policy also must provide Get, List, and Describe permissions for services supported by AWS Chatbot.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "a4b:Get*",
                "a4b:List*",
                "a4b:Describe*",
                "a4b:Search*",
                "acm:Describe*",
                "acm:Get*",
                "acm:List*",
                "acm-pca:Describe*",
                "acm-pca:Get*",
                "acm-pca:List*",
                "amplify:GetApp",
                "amplify:GetBranch",
                "amplify:GetJob",
                "amplify:GetDomainAssociation",
                "amplify:ListApps",
                "amplify:ListBranches",
                "amplify:ListDomainAssociations",
                "amplify:ListJobs",
                (...)
                "xray:BatchGet*",
                "xray:Get*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

### The CloudWatchReadOnlyAccess policy

You can attach the **CloudWatchReadOnlyAccess** policy to AWS Chatbot roles when you edit them in the IAM console. This policy does not appear in the AWS Chatbot console.

It is an AWS managed policy. You can attach this policy to any role for AWS Chatbot usage. You can define your own policy with greater restrictions, using this policy as a template.

AWS Chatbot users can use this policy to support Amazon CloudWatch events reporting, alarms, CloudWatch logs, and CloudWatch trend charts for most of AWS Chatbot's supported AWS services. It allows read-only operations for CloudWatch Logs and the Amazon Simple Notification Service service, and can be used in place of the customer managed **Notification permissions policy**. However, you must use the IAM console to attach this policy to any IAM role.

The Logs permissions also support the useful **Show Logs** feature for CloudWatch alarms notifications in Slack. AWS Chatbot also supports actions for displaying logs for Lambda and Amazon API Gateway.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:Describe*",
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "sns:Get*",
                "sns:List*"
                ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

### The AWS Support Command Permissions policy

The **AWS Support Command Permissions** policy appears in the AWS Chatbot console when you configure resources. It's provided in the AWS Chatbot console to conveniently set up a role, to allow Slack users to create AWS support tickets through their Slack channels.

In the IAM console, this policy appears as **AWSSupportAccess**.

It is an AWS managed policy. You can also attach this policy in IAM to any role. You can define your own policy with greater restrictions, using this policy as a template, for roles in AWS Chatbot.

The **Support Command Permissions** policy applies only to the AWS Support service.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "support:*"
            ],
            "Resource": "*"
        }
    ]
}
```

### The Amazon Q Permissions policy

The **Amazon Q Permissions** policy appears in the AWS Chatbot console when you configure your **Permissions**. This policy provides full access to enable interactions with Amazon Q, including administrator access. It includes access to log in with IAM Identity Center to access Amazon Q through an Amazon Q Developer Pro subscription. For more information, see AmazonQFullAccess in the *AWS Managed Policy Reference Guide*.

In the IAM console, this policy appears as **AmazonQFullAccess**.

It is an AWS managed policy. You can also attach this policy in IAM to any role. You can define your own policy with greater restrictions, using this policy as a template, for roles in AWS Chatbot.

The policy's JSON code is shown following:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "AllowAmazonQFullAccess",
      "Effect" : "Allow",
      "Action" : [
        "q:StartConversation",
        "q:SendMessage",
        "q:GetConversation",
        "q:ListConversations",
        "q:PassRequest",
        "q:StartTroubleshootingAnalysis",
        "q:GetTroubleshootingResults",
        "q:StartTroubleshootingResolutionExplanation",
        "q:UpdateTroubleshootingCommandResult",
        "q:GetIdentityMetadata",
        "q:CreateAssignment",
        "q:DeleteAssignment",
        "q:GenerateCodeFromCommands",
        "q:CreatePlugin",
        "q:DeletePlugin",
        "q:GetPlugin",
        "q:UsePlugin",
        "q:ListPlugins",
        "q:ListPluginProviders",
        "q:ListTagsForResource",
        "q:UntagResource",
        "q:TagResource"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "AllowCloudControlReadAccess",
      "Effect" : "Allow",
      "Action" : [
        "cloudformation:GetResource",
        "cloudformation:ListResources"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "AllowSetTrustedIdentity",
      "Effect" : "Allow",
      "Action" : [
        "sts:SetContext"
      ],
      "Resource" : "arn:aws:sts::*:self"
    },
    {
      "Sid" : "AllowPassRoleToAmazonQ",
      "Effect" : "Allow",
      "Action" : [
        "iam:PassRole"
      ],
      "Resource" : "arn:aws:iam::*:role/*",
      "Condition" : {
        "StringEquals" : {
          "iam:PassedToService" : [
            "q.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

### The Amazon Q Operations Assistant Permissions policy

The **Amazon Q Operations Assistant Permissions** policy appears in the AWS Chatbot console when you configure your **Permissions**. This policy enables Amazon Q to analyze your AWS resources during investigations of operational events. this policy is scoped based on the resources that Amazon Q Developer supports during investigations, and is updated as more resources are supported. For more information, see AIOpsOperatorAccess in the *Amazon CloudWatch User Guide*.

In the IAM console, this policy appears as **AIOpsOperatorAccess**.

It is an AWS managed policy. You can also attach this policy in IAM to any role. You can define your own policy with greater restrictions, using this policy as a template, for roles in AWS Chatbot.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AIOpsOperatorAccess",
            "Effect": "Allow",
            "Action": [
                "aiops:CreateInvestigation",
                "aiops:CreateInvestigationEvent",
                "aiops:CreateInvestigationResource",
                "aiops:DeleteInvestigation",
                "aiops:Get*",
                "aiops:List*",
                "aiops:UpdateInvestigation",
                "aiops:UpdateInvestigationEvent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SSOManagementAccess",
            "Effect": "Allow",
            "Action": [
                "identitystore:DescribeUser",
                "sso:DescribeInstance",
                "sso-directory:DescribeUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSTSContextSetting",
            "Effect": "Allow",
            "Action": [
                "sts:SetContext"
            ],
            "Resource": "arn:aws:sts::*:self"
        },
        {
            "Sid": "SSMSettingServiceIntegration",
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "arn:aws:ssm:*:*:servicesetting/integrations/*"
        },
        {
            "Sid": "SSMIntegrationTagAccess",
            "Effect": "Allow",
            "Action": [
                "ssm:AddTagsToResource",
                "ssm:CreateOpsItem"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Integration": [
                        "CloudWatch"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "Integration"
                }
            }
        },
        {
            "Sid": "SSMOpsItemIntegration",
            "Effect": "Allow",
            "Action": [
                "ssm:DeleteOpsItem",
                "ssm:UpdateOpsItem"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Integration": [
                        "CloudWatch"
                    ]
                }
            }
        },
        {
            "Sid": "SSMTagOperation",
            "Effect": "Allow",
            "Action": [
                "ssm:AddTagsToResource"
            ],
            "Resource": "arn:aws:ssm:*:*:opsitem/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Integration": [
                        "CloudWatch"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "Integration"
                }
            }
        },
        {
            "Sid": "SSMOpsSummaryIntegration",
            "Effect": "Allow",
            "Action": [
                "ssm:GetOpsSummary"
            ],
            "Resource": "*"
        }
    ]
}
```

## Customer managed IAM policies in AWS Chatbot

AWS Chatbot also supports three service-provided customer managed IAM policies, that you can apply to any AWS Chatbot role. They can also be used as templates for defining custom IAM permissions for your users:

- **ReadOnly Command Permissions policy**
- **Lambda-Invoke Command Permissions policy**
- **Notification permissions policy**

### The AWS Chatbot Read-Only Command Permissions IAM policy

The **Read-Only Command Permissions** policy appears in the AWS Chatbot console when you configure resources. You use this policy to support AWS commands and actions in Slack channels.

It is a customer managed policy. It pairs with the **Lambda-Invoke Command Permissions** policy to provide a convenient AWS Chatbot configuration to enable Slack channel support for sending commands to the AWS CLI.

The **Read-Only Command Permissions** policy denies permission for AWS Chatbot users to get sensitive information from AWS services through the Slack channel, such as Amazon EC2 password information, key pairs and login credentials.

This policy appears in the IAM console as the **AWS-Chatbot-ReadOnly-Commands-Policy**.

You can edit and assign this policy to any role in AWS Chatbot or in IAM. For editing of roles and policies for AWS Chatbot usage, we recommend using the IAM console.

**Note**

If you want to use this policy as a template, we recommend saving a new copy of the policy under a different name and making your changes there.

For your team's command usage in Slack channels, you must use a role that defines the necessary read-only permissions.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iam:*",
                "s3:GetBucketPolicy",
                "ssm:*",
                "sts:*",
                "kms:*",
                "cognito-idp:GetSigningCertificate",
                "ec2:GetPasswordData",
                "ecr:GetAuthorizationToken",
                "gamelift:RequestUploadCredentials",
                "gamelift:GetInstanceAccess",
                "lightsail:DownloadDefaultKeyPair",
                "lightsail:GetInstanceAccessDetails",
                "lightsail:GetKeyPair",
                "lightsail:GetKeyPairs",
                "redshift:GetClusterCredentials",
                "storagegateway:DescribeChapCredentials"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

### The AWS Chatbot Lambda-Invoke Command Permissions policy

The **Lambda-Invoke Command Permissions** policy appears in the AWS Chatbot console when you configure resources. It pairs with the **Read-Only Command Permissions** policy to provide a convenient AWS Chatbot configuration to enable Slack channel access to the AWS CLI, and to features that make sense for CLI use. The policy allows AWS Chatbot users to invoke Lambda functions in their Slack channels.

It is a customer managed policy. In the IAM console, it appears as **AWS-Chatbot-LambdaInvoke-Policy**.

You can edit and assign this policy to any role in AWS Chatbot or in IAM.

By default, the **Lambda-Invoke** policy is very permissive, because you can invoke any function for any action.

We recommend using this policy as a template to define your own, more restrictive policies, such as permissions to invoke functions developed for your DevOps team that only they should be able to invoke, and deny permissions to invoke Lambda functions for any other purpose. To edit roles and policies for AWS Chatbot, use the IAM console.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:invokeAsync",
                "lambda:invokeFunction"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

### The AWS Chatbot Notification Permissions IAM policy

The **Notification Permissions** policy appears in the AWS Chatbot console when you configure resources. It provides the minimum usable IAM policy configuration for using the AWS Chatbot in Slack channels and Amazon Chime webhooks. The **Notification Permissions** policy enables AWS Chatbot admins to forward CloudWatch Events, CloudWatch alarms, and format charting data for viewing in chat room messages. Because many of AWS Chatbot's supported services use CloudWatch as their event and alarm processing layer, AWS Chatbot requires this policy for core functionality. You can use other policies, such as **CloudWatchReadOnlyAccess**, in place of this policy, but you must attach that policy to the role in the IAM console.

It is a customer managed policy. You can edit and assign this policy to any role for AWS Chatbot usage.

> **Note**
>
> If you want to use this policy as a template, we recommend saving a new copy of the policy under a different name and making your changes there.

In the IAM console, it appears as **AWS-Chatbot-NotificationsOnly-Policy**.

The policy's JSON code is shown following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

# Identity-based IAM policies for AWS Chatbot

A policy is an object in AWS that, when you attach it to an identity, defines their permissions. When you create a policy to restrict or allow access to a resource, you can use an identity-based policy.

You can attach IAM identity-based policies to IAM entities such as a user in your AWS account, an IAM group, or an IAM role. You can define allowed or denied actions and resources, and the conditions under which actions are allowed or denied. AWS Chatbot supports specific actions, resources, and condition keys.

> **Note**
>
> To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.
> For information about the specific IAM JSON policy elements that AWS Chatbot supports, see Actions, Resources, and Condition Keys for AWS Chatbot in the *IAM User Guide*.

**Topics**

- Identity-based policies for AWS Chatbot
- Identity-based policy best practices
- Applying AWS Chatbot permissions to an IAM identity
- Allowing users to view their permissions

## Identity-based policies for AWS Chatbot

By default, IAM users, groups, and roles don't have permission to create or modify AWS Chatbot resources. They also can't perform tasks using the AWS Management Console or AWS Command Line Interface (AWS CLI). An IAM administrator can create IAM identity-based policies that grant entities permission to perform specific console and CLI operations on the resources that they need. The administrator attaches those policies to the IAM entities that require those permissions.

> **Note**
>
> In an identity-based policy, you don't specify the principal who gets the permission (the `Principal` element) because the policy gets attached to the entity that needs to use it.

To learn about all of the elements that you use in a policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*. For information about the specific IAM JSON policy elements that AWS Chatbot supports, see Actions, Resources, and Condition Keys for AWS Chatbot in the *IAM User Guide*.

**AWS Chatbot actions for identity-based policies**

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.

Actions in an AWS Chatbot policy use the following prefix before the action.

```
"Action": [

"chatbot:"

]
```

For example, to grant a user permission to view the list of all Slack channels using the `DescribeSlackChannels` operation, you include the `chatbot:DescribeSlackChannels` action in the user's policy. Policy statements must include either an `Action` or `NotAction` element. AWS Chatbot defines its own set of actions that describe tasks that you can perform with this service. To see the list of AWS Chatbot actions, see Actions, Resources, and Condition Keys for AWS Chatbot in the *IAM User Guide*.

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

```
"Action": [

"chatbot:DescribeSlackChannels",

"chatbot:DescribeSlackWorkspaces"

]
```

> ⚠ **Important**
>
> Although you can specify multiple actions of like type in a policy using wildcards (*), we strongly discourage doing so. Follow the practice of granting least privileges and narrowing the permissions necessary for a user to perform their work.

## Identity-based policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Chatbot 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 Validate policies with IAM Access Analyzer 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 Secure API access with MFA in the *IAM User Guide*.

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

## Applying AWS Chatbot permissions to an IAM identity

The following example of an AWS Chatbot identity-based policy controls all aspects of Slack chat room configuration. It grants full read-only permissions to Amazon CloudWatch and Amazon CloudWatch Logs, and Amazon Simple Notification Service (Amazon SNS) topics. It enables Slack chat room configuration through both the AWS Chatbot console and CLI actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllChatbotPermissions",
            "Action": [
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "sns:Get*",
                "sns:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
            {
            "Sid": "AllSlackPermissions",
            "Effect": "Allow",
            "Action": [
                "chatbot:Describe*",
                "chatbot:UpdateSlackChannelConfiguration",
                "chatbot:CreateSlackChannelConfiguration",
                "chatbot:DeleteSlackChannelConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

In this example, `"Resource": "*"` refers to all applicable Slack resources. You attach the policy to an IAM user, group, or role who needs access to all Slack resources.

## Allowing users to view their 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": "*"
        }
    ]
}
```

## IAM resource-level permissions for AWS Chatbot

*Resource-level permissions* define the AWS resources on which you allow assigned entities (users, groups, and roles) to perform actions. You specify the Amazon Resource Name (ARN) of one or more resources as part of an IAM policy, which you can then attach to IAM entities.

> **Note**
>
> AWS Chatbot doesn't support *resource-based policies*, which are directly attached to AWS resources. For more information about the differences between policies and permissions, see Identity-Based Policies and Resource-Based Policies in the *IAM User Guide*.

For more information about the differences between IAM policies and permissions, see Identity-Based Policies and Resource-Based Policies in the *IAM User Guide*. The following sections describe how resource-level permissions work with AWS Chatbot.

**Topics**

### Using the AWS Chatbot resource in a policy

You can set up an IAM policy that defines *who* (users, groups and roles) can perform actions on AWS Chatbot resources. The policy uses *resource-level permissions* to determine *which* AWS

Chatbot resources that users of the IAM policy can work with. The policy also defines *how* they can work with them (through Actions and Conditions).

When creating an IAM policy, you refer to the **chat-configuration** resource by its Amazon Resource Name (ARN). An AWS Chatbot resource ARN consists of three objects:

- A list of one or more Amazon Simple Notification Service (Amazon SNS) topic ARNs for the topics to be associated with the configuration.
- The ARN of the customer's IAM role.

  AWS Chatbot assumes the IAM role in the customer's account and makes API calls to other AWS services to get necessary information. For example, for an Amazon CloudWatch alarm notification, AWS Chatbot requires the metric graphic image displayed with the CloudWatch alarm notification. For that, AWS Chatbot calls a CloudWatch API with the customer's credentials.
- An Amazon Chime webhook URL or Slack channel ID/Slack workspace ID.

  When creating a resource-level permission for a chatbot configuration, in the JSON both Slack channels and Amazon Chime webhooks are considered a *chat-configuration*. The chat-configuration uses a following ARN field to distinguish between a Slack channel and a Amazon Chime webhook.

  The `configuration-name` field is the name for the Slack channel or Amazon Chime webhook that is defined in the AWS Chatbot console.

The AWS Chatbot resource ARN has the following format:

```
arn:${partition}:chatbot::${account-id}:chat-configuration/slack-channel/
${configuration-name}
```

Or:

```
arn:${partition}:chatbot::${account-id}:chat-configuration/chime-webhook/
${configuration-name}
```

For example:

```
arn:aws:chatbot::123456789021:chat-configuration/slack-channel/
devops_channel_01
```

Or:

```
arn:aws:chatbot::123456789021:chat-configuration/chime-webhook/
devops_webhook_IT_team_space
```

> **Note**
>
> When you create the permissions, ensure that any Actions apply to the correct configuration type.

## Example: AWS Chatbot resource-level permission

You can use resource-based permissions to allow or deny access to one or more AWS Chatbot resources in an IAM policy, or to all AWS Chatbot resources.

To add a resource-level permission to a policy, include the channel's ARN in a new `Resource` statement. The following example is based on the identity-based policy in AWS Chatbot Identity-Based Policies. It shows examples for both `slack-channel` and `chime-webhook` resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "sns:Get*",
                "sns:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
            {
            "Sid": "AllSlackPermissions",
            "Effect": "Allow",
            "Action": [
                "chatbot:Describe*",
                "chatbot:UpdateSlackChannelConfiguration",
                "chatbot:CreateSlackChannelConfiguration",
                "chatbot:DeleteSlackChannelConfiguration",
                "chatbot:CreateChimeWebhookConfiguration",
                "chatbot:UpdateChimeWebhookConfiguration"
                ],
            "Resource":"arn:aws:chatbot::123456789021:chat-configuration/slack-channel/
devops_private_channel"
            "Resource":"arn:aws:chatbot::123456789021:chat-configuration/chime-webhook/
devops_aws_chime_webhook1"
            }
        }
    ]
}
```

You attach the policy to the IAM entity that needs it. The associated users can create, edit, view and delete the resource's Slack chat channels, workspaces and associated SNS topics, and create and edit Amazon Chime webhooks.

## Using Service-Linked Roles for AWS Chatbot

A service-linked role is a type of IAM role that links directly to an AWS service. It gives AWS services the permissions to access resources in other services to complete actions on your behalf.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose any **Yes** entry with a link to view the service-linked role documentation for that service.

When you create an AWS Chatbot resource in the AWS Chatbot console, you can also choose to provide a list of one or more SNS topics to associate with the new resource. AWS Chatbot automatically uses the **AWSServiceRoleForAWSChatbot** service-linked role to add or remove subscriptions to the AWS Chatbot global Amazon SNS subscription endpoint.

The service-linked role makes setting up AWS Chatbot easier because you don’t have to manually add the necessary permissions. AWS Chatbot defines the permissions for the service-linked role and only AWS Chatbot can assume that role. The permissions include a trust policy and a permissions policy, which apply only to the AWS Chatbot service.

**Topics**

## AWS Chatbot Service-linked role for performing operations on Amazon SNS topics and CloudWatch Logs

AWS Chatbot uses the service-linked role named **AWSServiceRoleForAWSChatbot**. This is a managed IAM policy with scoped permissions that AWS Chatbot needs to run in customers' accounts.

### Service-Linked Role Permissions for AWS Chatbot

The AWS Chatbot service-linked role gives permissions for the following services and resources:

- Amazon SNS notifications
- CloudWatch Logs

These permissions allow AWS Chatbot to perform operations on Amazon SNS topics and CloudWatch Logs.

Administrators can view, but can't edit, the permissions for the AWS Chatbot service-linked role.

The **AWSServiceRoleForAWSChatbot** service-linked role provides trust permissions to the following service to assume its role:

- `management.chatbot.amazonaws.com`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

When you create an AWS Chatbot configuration, it creates the following policy for the service-linked role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
              "sns:ListSubscriptionsByTopic",
              "sns:ListTopics",
              "sns:Unsubscribe",
              "sns:Subscribe",
              "sns:ListSubscriptions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
              "logs:PutLogEvents",
              "logs:CreateLogStream",
              "logs:DescribeLogStreams",
              "logs:CreateLogGroup",
              "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/chatbot/*"
        }
    ]
}
```

You don't need to take any action to support this role beyond using the AWS Chatbot service.

### Enabling the service-linked role for AWS Chatbot

When you configure AWS Chatbot for the first time, you configure a Slack channel or Amazon Chime webhook to work with Amazon Simple Notification Service (Amazon SNS) topics for forwarding notifications to chat rooms. When you create the first resource, AWS Chatbot automatically creates the IAM service-linked role, which can be seen in the IAM console. You don't need to manually create or configure this role.

### Editing a service-linked role for AWS Chatbot

You can't edit the **AWSServiceRoleForAWSChatbot** service-linked role. You also can't change its name, because other entities might reference it. You can edit the role's description using the IAM console. For more information, see Editing a service-linked role in the *IAM User Guide*.

### Manually deleting the AWSServiceRoleForAWSChatbot service-linked role

Under specific circumstances, you can manually delete the **AWSServiceRoleForAWSChatbot** service-linked role. If you no longer need to use any feature or service that requires a service-linked role, we recommend that you delete that role. Doing so prevents having an unused entity that is not actively maintained in your account.

To delete the AWS Chatbot service-linked role, you must delete all AWS Chatbot resources in your AWS account, including all Slack channels and Amazon Chime webhooks. You can delete all AWS Chatbot resources using the AWS Chatbot console, and then use the IAM console or AWS Command Line Interface (AWS CLI) to delete the service-linked role.

> **Note**
>
> If AWS Chatbot is using the **AWSServiceRoleForAWSChatbot** service-linked role when you try to delete its resources, the deletion might fail. If that happens, wait a few minutes and try deleting it again.

**To delete AWS Chatbot resources**

1. Open the AWS Chatbot console.
2. To remove Amazon Chime webhook configurations, do the following:
    a. Choose **Amazon Chime**.
    b. Choose each webhook that you need to delete and choose **Delete webhook**. You can delete one at a time.
    c. Choose **Delete** to confirm the deletion.
    d. Repeat these steps to delete all webhook configurations.
3. To remove Slack channel configurations, do the following:
    a. Choose **Slack**.
    b. Choose the channel that you need to delete and choose **Delete channel**.
    c. Choose **Delete** to confirm the deletion.
    d. Repeat these steps to delete all Slack channel configurations.

    > **Note**
    >
    > If you delete the AWS Chatbot service-linked role, and then need to use it again, simply open the AWS Chatbot console and create a new Slack channel or Amazon Chime webhook resource to recreate the role in your account. When you create the first new resource in AWS Chatbot, it creates the service-linked role for you again.

4. To delete the **AWSServiceRoleForAWSChatbot** service-linked role, use the IAM console or the AWS Command Line Interface (AWS CLI) . For information, see Deleting a Service-Linked Role in the *IAM User Guide*.

### Supported regions for AWS Chatbot service-linked roles

AWSServiceRoleForAWSChatbot doesn't support using service-linked roles in every AWS Region where the service is available. The following table shows the Regions where you can use the **AWSServiceRoleForAWSChatbot**.

| Region Name | Region Identity | Supported in AWS Chatbot |
|---|---|---|
| US East (N. Virginia) | us-east-1 | Yes |
| US East (Ohio) | us-east-2 | Yes |
| US West (N. California) | us-west-1 | Yes |
| US West (Oregon) | us-west-2 | Yes |
| Asia Pacific (Mumbai) | ap-south-1 | Yes |
| Asia Pacific (Osaka) | ap-northeast-3 | Yes |
| Asia Pacific (Seoul) | ap-northeast-2 | Yes |
| Asia Pacific (Singapore) | ap-southeast-1 | Yes |
| Asia Pacific (Sydney) | ap-southeast-2 | Yes |
| Asia Pacific (Tokyo) | ap-northeast-1 | Yes |
| Canada (Central) | ca-central-1 | Yes |
| Europe (Frankfurt) | eu-central-1 | Yes |
| Europe (Ireland) | eu-west-1 | Yes |
| Europe (London) | eu-west-2 | Yes |

| Region Name | Region Identity | Supported in AWS Chatbot |
|---|---|---|
| Europe (Paris) | eu-west-3 | Yes |
| South America (São Paulo) | sa-east-1 | Yes |
| AWS GovCloud (US) | us-gov-west-1 | No |

## Troubleshooting AWS Chatbot identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Chatbot and IAM.

**Topics**

- I'm not authorized to perform an action in AWS Chatbot
- I'm not authorized to perform iam:PassRole
- I'm an administrator and want to allow others to access AWS Chatbot
- I want to allow people outside of my AWS account to access my AWS Chatbot resources

### I'm not authorized to perform an action in AWS Chatbot

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`widget`* but does not have `chatbot::`*`GetWidget`* permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `chatbot::`*`GetWidget`* action.

## I'm 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 AWS Chatbot.

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 AWS Chatbot. 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'm an administrator and want to allow others to access AWS Chatbot

To allow others to access AWS Chatbot, you must grant permission to the people or applications that need access. If you are using AWS IAM Identity Center to manage people and applications, you assign permission sets to users or groups to define their level of access. Permission sets automatically create and assign IAM policies to IAM roles that are associated with the person or application. For more information, see Permission sets in the *AWS IAM Identity Center User Guide*.

If you are not using IAM Identity Center, you must create IAM entities (users or roles) for the people or applications that need access. You must then attach a policy to the entity that grants them the correct permissions in AWS Chatbot. After the permissions are granted, provide the credentials to the user or application developer. They will use those credentials to access AWS. To learn more about creating IAM users, groups, policies, and permissions, see IAM Identities and Policies and permissions in IAM in the *IAM User Guide*.

### I want to allow people outside of my AWS account to access my AWS Chatbot 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 AWS Chatbot supports these features, see How AWS Chatbot works with IAM.
- 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 Cross account resource access in IAM in the *IAM User Guide*.

## Compliance validation for AWS Chatbot

Third-party auditors assess the security and compliance of AWS Chatbot as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. 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 Chatbot 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 security- and compliance-focused baseline environments on AWS.

- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS 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 that helps you check your compliance with security industry standards and best practices.

## Resilience in AWS Chatbot

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 Availability 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.

## Infrastructure security in AWS Chatbot

As a managed service, AWS Chatbot is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS Chatbot through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. We recommend TLS 1.3. Clients must also support cipher suites with perfect forward secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). 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.

# Troubleshooting AWS Chatbot

AWS Chatbot operates with several AWS services, including Amazon CloudWatch, Amazon GuardDuty, and AWS CloudFormation. If you encounter issues when trying to receive notifications, see the following topic for troubleshooting help.

## Notifications aren't sent to chat rooms.

If you configured your AWS service to send notifications to the Amazon Simple Notification Service (Amazon SNS) topics mapped to AWS Chatbot, but the notifications aren't appearing in the chat rooms or channels, try the steps below.

**Possible causes**

- **There is no connectivity.**

  Test your connectivity and your AWS Chatbot configuration by using the **Send test message button** in the AWS Chatbot console. For more information, see Test notifications from AWS services to Amazon Chime, Test notifications from AWS services to Microsoft Teams, or Test notifications from AWS services to Slack.

- **The bot is not invited to the channel.**

  Ensure that the AWS Chatbot app ("@aws") is added to the chat channel. If it hasn't, in Microsoft Teams or Slack, add the AWS Chatbot app by choosing **Add apps** from the channel's **Details** screen.

- **The notification's originating service is not supported by AWS Chatbot.**

  For a list of supported services, see Using AWS Chatbot with Other AWS Services.

- **The SNS topic doesn't have a subscription to AWS Chatbot.**

  In the Amazon SNS console, go to the **Topics** page, choose the **Subscriptions** tab, and then verify that the topic has a subscription. If the topic doesn't, open the AWS Chatbot console, open your authorized client, and then look at the **Configured channels** or **Configured webhooks** list. Add a new channel or webhook configuration, and then add the SNS topic. Without this configuration, event notifications can't reach the chat rooms.

- **The Amazon SNS topic has server-side encryption turned on.**

If you have server-side encryption enabled for your Amazon SNS topics, you must give permissions to the sending services in your AWS KMS key policy to post events to the encrypted SNS topics. The following policy is an example for EventBridge.

```
{
  "Sid": "Allow CWE to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

In order to successfully test the configuration from the console, your role must also have permission to use the AWS KMS key.

AWS managed service keys don’t allow you to modify access policies, so you will need AWS KMS/CMK for encrypted SNS topics. You can then update the access permissions in the AWS KMS key policy to allow the service that sends messages to publish to your encrypted SNS topics (for example, EventBridge).

- **Your SNS topic subscription to the AWS Chatbot has the Enable raw message delivery setting enabled.**

  Don't enable the **Enable raw message delivery** feature for any SNS topic subscriptions to AWS Chatbot.

- **The event was throttled.**

  AWS Chatbot allows for 10 events per second. If more than 10 events per second are received, any event above 10 is throttled.

## How can I unsubscribe from AWS Chatbot notifications in a channel or chat room?

To unsubscribe a channel or chat room from all AWS Chatbot notifications, remove the respective configuration from the AWS Chatbot console. Otherwise, to identify certain service and notification-types to unsubscribe from, see the section called “I don't want to receive notifications from certain services anymore.”

## I don't want to receive notifications from certain services anymore.

If you want to unsubscribe only some notifications from the channel or chatroom, you can remove specific SNS topics from the AWS Chatbot configuration. Alternatively, you can remove the specific SNS topics as the event and alarm notification targets from the respective service configurations. You should also check if you have Amazon EventBridge rules configured for the service event types and remove the specific SNS topics as the rule triggers targets.

## CloudWatch alarm notifications don't show the graphs from the reporting metrics.

**Possible causes**

- **The IAM role doesn’t have CloudWatchRead permissions.**

  In the AWS Chatbot console, create a new role. This role requires the Notifications permissions policy from the AWS Chatbot console when you configure a new webhook or Slack channel. You can also edit your IAM role to add the CloudWatchRead permissions for AWS Chatbot.

- **AWS Chatbot doesn't have access to all AWS Regions.**

  AWS Chatbot may execute API calls from any nearby AWS Region. If any Region is disabled, you may experience problems with CloudWatch metrics graphs, among other issues. For more information, see the section called “I get AccessDenied or permissions errors.”

## When I set up an SNS topic in the AWS Billing and Cost Management console to forward notifications to the AWS Chatbot, I get a "Please comply with SNS Topic ARN format" error message.

If the AWS Billing and Cost Management console displays an error message for the SNS topic you want to use for notifications, you can edit the SNS topic's permissions policy so it can forward Budget notifications.

Do this if you have already configured an SNS topic that has a subscription to AWS Chatbot or you've configured a new SNS topic. It is not needed if you want to use an Amazon SNS topic that is already configured and working with AWS Billing and Cost Management. You can then set up that topic with a subscription to AWS Chatbot.

## How do I edit my configuration name?

Configuration names can't be edited. Names must be unique across your account.

## I get AccessDenied or permissions errors.

### Possible causes

- **You are missing some IAM permissions or trust relationships.**

  Make sure you have the correct policies set up by following the instructions found in Setting up AWS Chatbot and Identity and Access Management for AWS Chatbot.
- **AWS Chatbot doesn't have access to all AWS Regions.**

  AWS Chatbot is a global service and may execute API calls from any nearby AWS Region. If any Region is disabled, you may experience errors. Make sure the IAM role you set up for AWS Chatbot to assume has access to all Regions.

  Other policy types can limit how IAM roles can be assumed. If you have set up your AWS Chatbot IAM role to have global access but you're still getting errors, one of these policy types may be the culprit:
  - **AWS Organizations service control policies (SCPs)** - SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. A service control policy could be overriding the policies you put in place for AWS Chatbot. See How SCPs Work in the *AWS Organizations User Guide*.

- **IAM account settings**

  With IAM, you can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources. When you activate STS endpoints for a Region, AWS STS can issue temporary credentials to users and roles in your account that make an AWS STS request. Those credentials can then be used in any Region that is enabled by default or is manually enabled. You must activate the Region in the account where the temporary credentials are generated. It does not matter whether a user is signed into the same account or a different account when they make the request. For more information, see Activating and deactivating AWS STS in an AWS Region in the *IAM User Guide*.

If there is a policy in place that prevents access to services in certain Regions, you must change the policy to allow global AWS Chatbot access.

For example, the policy below allows AWS Chatbot in **us-east-2** but denies other services by using a NotAction element.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "NotAction": [
                "chatbot:*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": [
                        "us-east-2"
                    ]
                }
            }
        }
    ]
}
```

## I get an "Event received is not supported" error.

### Possible causes

- **The notification's originating service is not supported by AWS Chatbot.**

  For a list of supported services, see Monitoring AWS services. You can also send customized notifications for these services using custom notifications. For more information, see Custom notifications in AWS Chatbot.

- **The service event is modified.**

  AWS Chatbot only supports default service events. If you need to modify a service event or send custom notifications, send the message using the custom notifications event schema. For more information, see Custom notifications in AWS Chatbot.

## I get an "A valid member name is required" error.

### Possible causes

- **The AWS Chatbot app is not added to the Slack workspace.**

  Add the AWS Chatbot app to the Slack workspace. For more information, see the Getting started guide for AWS Chatbot.

## I get a Slack error saying "You are not authorized to install AWS Chatbot on AWS."

### Possible causes

- **A new scope change requires administrator approval.**

  There may be a new scope added to the AWS Chatbot Slack application that requires approval by an administrator. If AWS Chatbot has released a new scope, administrators need to re-approve the AWS Chatbot Slack application. Note that the approval is only required for Slack workspaces with an app approval policy.

I get an "Event received is not supported" error. 170

Workspace administrators can check their workspace settings to review and approve new scopes for AWS Chatbot. For more information about how to approve an app, see Approve or restrict an app at the org level in the Slack Help Center.

- **Installation of the AWS Chatbot Slack app is restricted for your workspace.**

  This error may appear if the workspace administrator has explicitly restricted the installation of the AWS Chatbot Slack app.

## I can't add AWS Chatbot to a private channel in Microsoft Teams.

Microsoft Teams doesn't currently support AWS Chatbot in private channels. For more information, see Private channel limitations.

# Provide feedback

You can provide feedback about AWS Chatbot directly from your Amazon Chime chat room, chat channels, or from the AWS Chatbot console. To leave feedback from your Amazon Chime chat room or chat channel, type the following command and replace *comments* with your own information.

```
@aws feedback comments
```

To leave feedback from the AWS Chatbot console, navigate to the AWS Chatbot console and choose the **Feedback** link at the bottom of the console. All feedback is sent directly to and reviewed by the AWS Chatbot team.

# Document history

The following table describes important changes to the *AWS Chatbot Admin Guide*. For notifications about documentation updates, you can subscribe to the RSS feed.

| Change | Description | Date |
|---|---|---|
| Monitoring investigations in AWS Chatbot | New page added to explain how to monitor investigations with Amazon Q Developer in AWS Chatbot. | December 3, 2024 |
| Authorize Support cases | New page added to explain how to monitor and manage support cases. | November 18, 2024 |
| Ask Amazon Q Developer update | Page updated to explain enhanced Amazon Q capabilities. | November 18, 2024 |
| Securing your organizations | New page added to explain how to secure your organizations using organization policies and service control policies. | September 26, 2024 |
| Invoking Bedrock Agents | New page added to explain how to invoke Bedrock Agents from chat channels. | September 19, 2024 |
| Tagging support added | New page added to explain how to tag resources. | May 23, 2024 |
| Amazon Q support added | New page added to explain how to use Amazon Q with AWS Chatbot. | November 28, 2023 |

| | | |
|---|---|---|
| Custom notification support added | New page added to explain how to customize notifications. | September 12, 2023 |
| Natural language and AWS Resource Explorer support added | New page added to explain the addition of natural language support and Resource Explorer. | March 30, 2023 |
| Microsoft Teams support added | Microsoft Teams added as a supported chat client. | March 16, 2023 |
| Command alias support added | Content updates to reflect addition of command aliases. | November 22, 2022 |
| Data privacy section updated | Content updates to reflect AI powered improvements and data privacy. | May 11, 2022 |
| Additional CLI command support added | Content updates to reflect expansion of available CLI commands. | November 29, 2021 |
| AWS Chatbot EventBridge release | Content updates to reflect addition of Amazon EventBridge. | April 23, 2021 |
| AWS Chatbot general availability release | Content updates to reflect improvements made to AWS Chatbot during the preview period. | April 22, 2020 |
| Updated CLI commands information. | Updated CLI commands information to add IAM policy information for support tickets, minor updates/edits elsewhere. | December 13, 2019 |

| | | |
|---|---|---|
| Add support for CLI commands in Slack channels. | Addition of documentation for configuring support of commands for AWS services in Slack channels. | November 22, 2019 |
| Announcement of AWS code services development tools support. | Addition of code services notification support information. Miscellaneous edits and fixes. | November 7, 2019 |
| Enhanced troubleshooting information. | New Troubleshooting items. Minor updates and doc linking changes for accuracy. | August 28, 2019 |
| Addition of first set of AWS CloudTrail logging notifications for AWS Chatbot. | AWS CloudTrail provides logging support for several newly integrated AWS Chatbot API actions. | August 7, 2019 |
| AWS Chatbot is now in beta release. | AWS Chatbot is an AWS service that enables DevOps and software development teams to use Amazon Chime or Slack chat rooms to monitor and respond to operational events in their AWS Cloud. | July 24, 2019 |