AWS Chatbot

Administrator Guide

AWS Chatbot: Administrator Guide

Table of Contents

What is AWS Chatbot? ... 1
- Features of AWS Chatbot ... 1
- How AWS Chatbot works ... 2
- Regions and quotas for AWS Chatbot ... 2
- AWS Chatbot requirements ... 2
- Accessing AWS Chatbot ... 2

Setting up AWS Chatbot ... 3
- Prerequisites ... 3
- Setting up IAM permissions for AWS Chatbot ... 3
- Setting up Amazon SNS topics ... 4

Getting started ... 5
- Prerequisites ... 5
- Step 1: Set up chat clients for AWS Chatbot ... 5
  - Setting up AWS Chatbot with Slack ... 5
  - Setting up AWS Chatbot with Amazon Chime ... 7
- Step 2: Subscribe an Amazon SNS topic to AWS Chatbot ... 8
- Step 3: Test notifications with AWS Chatbot ... 9
- Step 4: Remove chat rooms ... 10
  - Removing a Slack client ... 10
  - Removing an Amazon Chime webhook ... 11
- Configuring an IAM role for AWS Chatbot ... 11
- Next steps ... 12

Using AWS Chatbot with other AWS services ... 13
- AWS Billing and Cost Management ... 13
- AWS CloudFormation ... 13
- Notifications for AWS developer tools ... 14
- Amazon CloudWatch Alarms ... 14
- Amazon CloudWatch Events ... 14
  - AWS Config ... 14
  - Amazon GuardDuty ... 15
  - AWS Health ... 15
  - AWS Security Hub ... 15
  - AWS Systems Manager ... 16

Using AWS Chatbot with Slack ... 17
- Running AWS CLI commands from Slack channels ... 17
  - Using commands in Slack ... 17
  - Managing permissions for running commands using AWS Chatbot ... 18
  - Running commands in Slack ... 20
  - Configuring commands support on an existing Slack channel ... 22
  - Enabling multiple accounts to use commands in a Slack channel ... 23
- Using AWS Lambda with AWS Chatbot - Common use cases ... 23
  - Add an IP rule to a security group ... 23
  - Remove an IP rule from a security group ... 24
  - Change the capacity of an Amazon EC2 Auto Scaling group ... 25
  - Approve an AWS CodePipeline action ... 25
- Tutorial: Using AWS Chatbot to run an AWS Lambda function remotely ... 26
  - Prerequisites ... 26
  - Create a Lambda function ... 27
  - Create an SNS topic ... 27
  - Configure a CloudWatch alarm ... 28
  - Configure a Slack client for AWS Chatbot ... 28
  - Invoke a Lambda function from Slack ... 29
  - Test the CloudWatch alarm ... 30
  - Clean up resources ... 30

Monitoring 32
Monitoring with CloudWatch 32
Enabling CloudWatch Metrics 32
Available metrics and dimensions 32
Viewing AWS Chatbot metrics 33
CloudWatch Logs 33
Enabling CloudWatch Logs 34
Viewing CloudWatch Logs 34
Logging AWS Chatbot API calls with AWS CloudTrail 34
Logging AWS Chatbot API information in CloudTrail 35
Logging other AWS API information in CloudTrail 35
Example: AWS Chatbot log file entries 36
Security 38
Data protection in AWS Chatbot 38
Identity and Access Management for AWS Chatbot 39
Audience 39
How AWS Chatbot works with IAM 39
IAM policies for AWS Chatbot 44
Identity-based IAM policies for AWS Chatbot 49
IAM resource-level permissions for AWS Chatbot 52
Using service-linked roles for AWS Chatbot 54
Compliance validation for AWS Chatbot 57
Resilience in AWS Chatbot 57
Infrastructure security 58
Troubleshooting 59
Document history 62
AWS glossary 63

What is AWS Chatbot?

AWS Chatbot is an AWS service that enables DevOps and software development teams to use Amazon Chime and Slack 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 Amazon Chime and Slack chat rooms so teams can analyze and act on them immediately, regardless of location.

You can also run AWS CLI commands in Slack channels, and file AWS Support cases from the Slack screen.

For information about AWS Chatbot compliance, see AWS services in scope by compliance program.

Topics

- Features of AWS Chatbot (p. 1)
- How AWS Chatbot works (p. 2)
- Regions and quotas for AWS Chatbot (p. 2)
- AWS Chatbot requirements (p. 2)
- Accessing AWS Chatbot (p. 2)

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 Slack and Amazon Chime** – You can add AWS Chatbot to your Slack channel or Amazon Chime chat rooms 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.
- **Retrieve diagnostics information through the AWS CLI with Slack** – AWS Chatbot supports read-only commands for most AWS services, making it easy to retrieve diagnostic information about your AWS resources from Slack on desktop and mobile devices. Your teams can analyze and respond to events faster by retrieving diagnostic information in real-time, from a centralized location. You can also initiate workflows by invoking Lambda functions or create AWS Support cases with a simple command from Slack. AWS Chatbot commands use the standard AWS Command Line Interface syntax.

How AWS Chatbot works

AWS Chatbot uses Amazon Simple Notification Service (Amazon SNS) topics to send event and alarm notifications from AWS services to Slack and Amazon Chime chat rooms. Slack and Amazon Chime users map the SNS topics to their Slack channels or Amazon Chime webhooks. For Slack, after the Slack administrator approves AWS Chatbot support for the Slack workspace, anyone in the workspace can add AWS Chatbot to their Slack 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 Amazon Chime and Slack 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 Using AWS Chatbot with Other AWS Services.

Regions and quotas for AWS Chatbot

For 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.

AWS Chatbot requirements

To use AWS Chatbot, you need the following:

- An AWS account to associate with Amazon Chime or Slack chat clients during AWS Chatbot setup.
- Administrative privileges for your Slack workspace or Amazon Chime chat room. 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.

For testing, we recommend using the role that you create with the AWS Chatbot console. To use an existing IAM role, see Configuring an IAM Role for AWS Chatbot.

Accessing AWS Chatbot

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

Setting up AWS Chatbot

To use AWS Chatbot, you authorize an Amazon Chime configuration or a Slack workspace 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.

Topics

- Prerequisites (p. 3)
- Setting up IAM permissions for AWS Chatbot (p. 3)
- Setting up Amazon SNS topics (p. 4)

Prerequisites

With AWS Chatbot, you can use Amazon Chime and Slack 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 signed up for AWS and created an AWS Identity and Access Management (IAM) administrator user. If this is your first time using AWS, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.
- You have started using some AWS services. For more information about AWS services you can use with AWS Chatbot, see Using AWS Chatbot with other AWS services (p. 13).
- You have administrator privileges with a Slack workspace or an Amazon Chime chat room.

Setting up IAM permissions for AWS Chatbot

If you have an existing administrator user, you can access the AWS Chatbot console with no additional permissions.

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

> **Note**
> All users in the Slack channel or Amazon Chime chat room will have the permissions defined by the role.

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**, 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 (p. 41).

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*.

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.

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

Topics

- Prerequisites (p. 5)
- Step 1: Set up chat clients for AWS Chatbot (p. 5)
- Step 2: Subscribe an Amazon SNS topic to AWS Chatbot (p. 8)
- Step 3: Test notifications from AWS services to Amazon Chime or Slack chat rooms (p. 9)
- Step 4: Remove chat rooms (p. 10)
- Configuring an IAM role for AWS Chatbot (p. 11)
- Next steps (p. 12)

Prerequisites

Before you get started, make sure you've completed the tasks in Setting up AWS Chatbot (p. 3).

Step 1: Set up chat clients for AWS Chatbot

You use the AWS Chatbot console to configure Amazon Chime and Slack clients to receive notifications from Amazon Simple Notification Service (Amazon SNS) topics.

> **Note**
> When you configure your clients, don't enable the **Enable raw message delivery** feature for any Amazon SNS topic subscription that you want to use for AWS Chatbot.

AWS Chatbot requires an AWS Identity and Access Management (IAM) role with Amazon CloudWatch read permissions and a trust policy that allows AWS Chatbot to use those permissions on your behalf. When you configure AWS Chatbot, you can create a role with a predefined set of policies to display CloudWatch charts in AWS Chatbot notifications.

You can also use an existing IAM role that you can configure for use with AWS Chatbot. For more information, see Configuring an IAM role for AWS Chatbot. For simplicity, particularly in testing your setup, we recommend using the IAM role with predefined policies that you can configure in AWS Chatbot.

Setting up AWS Chatbot with Slack

To allow AWS Chatbot to send notifications to your Slack channel, you must configure AWS Chatbot with Slack. Owners of Slack workspaces can approve the use of the AWS Chatbot, and any workspace user can configure the workspace to receive notifications or run commands.

To configure a Slack client

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

2. Under **Configure a chat client**, choose **Slack**, then choose **Configure client**.
3. 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.
4. Choose **Allow**.
5. On the **Workspace details** page, you can choose to continue within the console or with an AWS CloudFormation template:

   - To use an AWS CloudFormation template, copy and paste the **Workspace ID** found under **Workspace details**. For more information, see AWS::Chatbot::SlackChannelConfiguration in the *AWS CloudFormation User Guide*.
   - To continue within the console, choose **Configure new channel**.
6. Under **Configuration details**, enter a name for your configuration. The name must be unique across your account and can't be edited later.
7. 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 (p. 33).

   Note
   There is an extra charge for using CloudWatch Logs.
8. For **Slack channel**, choose the channel that you want to use.

   Note
   You can use private Slack channels with AWS Chatbot. To do so, choose **Private channel**. 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`). 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.)
9. Define the **IAM permissions** that the chatbot uses for messaging your Slack chat room:

   a. For **IAM role**, choose **Create an IAM role using a template**. 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.

   b. For **Role name**, enter a name. Valid characters: a-z, A-Z, 0-9, .\w+=,.@-_.

   c. For **Policy templates**, choose **Notification permissions**. This is the IAM policy template for AWS Chatbot. It provides the necessary Read and List permissions for CloudWatch alarms, events and logs, and for Amazon SNS topics.
10. Choose the SNS topics that will send notifications to the Slack channel.

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

    b. 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.

    c. To add an Amazon SNS topic from another AWS Region to the notification subscription, choose **Add another Region**.
11. Choose **Configure**.

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

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

Note
If you configure a private Slack channel, run the `/invite @AWS` command in Slack to invite the AWS Chatbot to the chat room.

The SNS topics you choose also must be configured in the services for which you want to receive notifications. For more information, see Using AWS Chatbot with Other AWS Services.

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 (p. 33).

   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**.

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 Using AWS Chatbot with Other AWS Services.

Note
You can configure a Slack channel to run commands to your AWS account. For more information, see Running AWS CLI Commands from Slack Channels (p. 17).

Step 2: Subscribe 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 Slack channel or Amazon Chime webhook. After doing so, the messages from those topics will appear in the Slack or Amazon Chime chat rooms. 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 Slack or Amazon Chime clients without further configuration.

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Under **Configured clients**, choose Slack or Amazon Chime.
3. Choose any channel in the Slack workspace configuration or webhook in the Amazon Chime webhooks list.
4. Choose **Edit**. The configuration page for the channel or webhook 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 Amazon Chime webhook or Slack 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.

Step 3: Test notifications from AWS services to Amazon Chime or Slack chat rooms

To verify that an Amazon Simple Notification Service (Amazon SNS) topic sends notifications to your Amazon Chime or Slack chat room, 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 Using AWS Chatbot with Other AWS Services.

> **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 Using AWS Chatbot with Other AWS Services 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 Slack channel or Amazon Chime webhook. For more information, see 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.

Step 4: Remove chat rooms

Removing an authorized Slack client from AWS Chatbot

When necessary, you can remove a Slack chat client from the AWS Chatbot configuration. Doing so *deauthorizes* the Slack client, which revokes the permissions that AWS Chatbot uses to operate with Slack.

Before deauthorizing a Slack client, you must delete all Slack channels. Deleting the channels first prevents accidentally deleting the Slack workspace.

To remove a Slack client

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Choose **Configured clients**.
3. On the **Configured clients** page, choose the Slack client.
4. Choose each channel in the Slack workspace configuration and choose **Delete**.

5. After you finish deleting all Slack channels from the workspace, choose **Remove workspace configuration**. AWS deletes the Slack workspace.

Removing an Amazon Chime webhook from AWS Chatbot

You can remove an Amazon Chime webhook from the AWS Chatbot configuration. Doing so *deauthorizes* the Amazon Chime webhook, which revokes the permissions that AWS Chatbot uses to operate with Amazon Chime.

To remove an Amazon Chime webhook

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Choose **Configured clients**.
3. On the **Configured clients** page, choose the Amazon Chime webhook you want to delete.
4. Choose **Delete**.

Configuring 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 Slack 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 need to 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.

> **Note**
> All users in the Slack channel or Amazon Chime chat room will have the permissions defined by the role.

To edit roles

1. Open the AWS Chatbot console at https://console.aws.amazon.com/chatbot/.
2. Choose the Slack channel or Amazon Chime webhook, and choose the IAM role associated with the channel or webhook.

   The IAM console opens, automatically showing role configuration page, with the **Permissions** tab displaying the selected role.
3. Choose **Attach Policies**.

   > **Note**
   > You can attach AWS managed policies and customer managed policies. AWS Chatbot roles support both types of IAM policies.
4. Choose the policy you want by choosing its name. You can use the **Search** box to search for the policy by its 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. Following are the preconfigured customer managed policies available for AWS Chatbot:

- **AWS-Chatbot-NotificationsOnly-Policy**
- **AWS-Chatbot-LambdaInvoke-Policy**
- **AWS-Chatbot-ReadOnly-Commands-Policy**

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

5. You can also 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 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 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 don't overwrite existing customer managed policies unless necessary.

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 Using AWS Chatbot with other AWS services (p. 13).
- Learn about using AWS CLI syntax on your Slack channels in Running AWS CLI commands from Slack channels (p. 17).

Using AWS Chatbot with 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 Slack channel or Amazon Chime webhook in the AWS Chatbot console.

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

Topics

- AWS Billing and Cost Management (p. 13)
- AWS CloudFormation (p. 13)
- Notifications for AWS developer tools (p. 14)
- Amazon CloudWatch Alarms (p. 14)
- Amazon CloudWatch Events (p. 14)

You can set up the following AWS services to forward notifications to Amazon Chime 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 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. The `Alarms` action sends notifications to an Amazon SNS topic that forwards the notification to AWS Chatbot for viewing in chat rooms.

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 CloudWatch Events

AWS Chatbot supports several AWS services through Amazon CloudWatch Events rules. CloudWatch uses CloudWatch Events 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 CloudWatch Events rule, and then map it to an Slack channel or Amazon Chime webhook in the AWS Chatbot console. When a service event matches the rule, the rule's Amazon SNS topic sends a notification to the chat room.

AWS Chatbot supports CloudWatch Events for the following AWS services: AWS Config, Amazon GuardDuty, AWS Health, AWS Security Hub, and AWS Systems Manager.

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 Slack 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 Personal Health Dashboard (PHD).

AWS Health directly supports CloudWatch Events 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*.

Using AWS Chatbot with Slack

AWS Chatbot enables you to use different AWS services through Slack. For example, you can retrieve diagnostic information, invoke Lambda functions, and create support cases for your AWS resources. To do these things, you can run commands using AWS CLI syntax directly in Slack channels.

Topics

- Running AWS CLI commands from Slack channels (p. 17)
- Using AWS Lambda with AWS Chatbot - Common use cases (p. 23)
- Tutorial: Using AWS Chatbot to run an AWS Lambda function remotely (p. 26)

Running AWS CLI commands from Slack channels

You can run commands using AWS CLI syntax directly in Slack channels. AWS Chatbot enables you to retrieve diagnostic information, invoke Lambda functions, and create support cases for your AWS resources.

When you interact with AWS Chatbot in Slack, it parses your input and prompts you for any missing parameters before it runs the command.

Using commands in Slack

After you set up the AWS Chatbot in your Slack workspace, you run commands in Slack with the following prefix:

```
@aws
```

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.

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 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.

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

- AWS Chatbot does not support commands to create, delete, or configure AWS resources (for example, to delete an S3 bucket).
- 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 Slack 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 Slack channel. You must enter, or copy and paste each AWS CLI command in the Slack channel.
- You can create AWS support cases through your Slack channels. You cannot add attachments to these cases from the Slack channel.
- Slack channels do not support standard AWS CLI pagination.

Managing permissions for running commands using 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 an IAM user, role, or group. These policies control what actions an identity can perform. AWS Chatbot provides three IAM policies in the AWS Chatbot console, that you can use to quickly set up AWS CLI commands support for Slack 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 role in the AWS Chatbot console and attach them there, or 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 (p. 11).

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 Slack. When you use the **ReadOnly Command Permissions** policy, you allow or deny the following permissions to users who run commands in Slack:

- 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)
- 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 Slack 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 **Lambda-Invoke Command Permissions** policy is shown following:

```
{
    "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.

Running commands in Slack

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.

Getting help for AWS services

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

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 Slack 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
```

Displaying Amazon CloudWatch Logs information

CloudWatch alarm notifications show buttons in Slack 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 in Slack. 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 in Slack.

Creating an AWS Support case

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 Slack to create AWS support tickets through their Slack channels.

You can quickly create a new AWS support case from Slack 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.

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 Slack channel

If you have existing Slack 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 **Slack** workspace. If you have only one workspace, its contents (the list of Slack channels) appear on the page.

   Note
   In this procedure, we assume use of an existing AWS Chatbot Slack channel configuration. The process is very similar if you need to create a new Slack configuration by choosing **Configure new client**.
3. Choose a Slack channel from the **Configured channels** list, and choose **Edit**. The selected channel can be public or private.
4. In the **Edit Slack channel** page, define the **IAM permissions** that the chatbot uses for messaging your Slack chat room.
5. For **IAM role**, choose **Create an IAM role using a template**.

   Note
   The first time you reconfigure a Slack channel to work with commands, you must create a new role using the AWS Chatbot policies as templates, or choose an existing role that provides the permissions you need.
6. For the **Role name**, enter a new role name using alphanumeric characters. Valid characters: a-z, A-Z, 0-9, .\w+=,.@-_.
7. For **Policy templates**, choose the **Read Only command permissions** and **Lambda-Invoke command permissions** policies for the role.

   If you plan to have users of the role submit AWS Support cases, also attach the **AWS Support command permissions** policy.
8. If you want the role to allow users to view CloudWatch alarms in graphical format, add the **Notification Permissions** policy. It provides the necessary Read and List permissions for CloudWatch alarms, events and logs, and for Amazon SNS topics.

   Note
   You do not need to edit or change the Amazon SNS topics configuration for the Slack channel.
9. 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 Slack channel. To do so, see Configuring an IAM Role for AWS Chatbot (p. 11).

   Important
   If you have a large number of Slack 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 Slack channels without further modification. The IAM policies will be consistent across Slack channels that support commands in your AWS Chatbot service.

Enabling multiple accounts to use commands in a Slack channel

You can configure AWS Chatbot for multiple AWS accounts in the same Slack 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.

Using AWS Lambda with AWS Chatbot - Common use cases

Common use cases for using AWS Chatbot in Slack involve invoking Lambda functions. This topic includes example use cases for using AWS Chatbot in Slack to invoke Lambda functions. In these use case examples, we use the payload parameter. When specified in an AWS CLI command, the payload parameter accepts a JSON object. For a tutorial that walks you through how to use AWS Chatbot with other services, see the Tutorial: Using AWS Chatbot to run an AWS Lambda function remotely (p. 26).

Topics

- Add an IP rule to a security group (p. 23)
- Remove an IP rule from a security group (p. 24)
- Change the capacity of an Amazon EC2 Auto Scaling group (p. 25)
- Approve an AWS CodePipeline action (p. 25)

Add an IP rule to a security group

The example in this section uses the AWS CLI and a Lambda function to add an IP rule to an existing Amazon EC2 Security Group. This allows you to add IP rules from Slack without having to access a console. When using this function, consider removing the added IP address when it no longer needs access to the security group. You should also ensure you are being secure by not adding unverified or inappropriate IP addresses. For more information about security groups, see Security groups for your VPC in the *Amazon Virtual Private Cloud User Guide*.

In the following code example, the allowlist-ip Lambda function accepts the payload specified by the user and adds the IP address by programmatically creating an inbound rule using `authorize_security_group_ingress`. This gives the specified IP address SSH access. The Lambda code uses Python 3.8. For more information about the `authorize_security_group_ingress` method, see the Adding rules to a security group in the *Amazon EC2 User Guide for Linux Instances*.

```
import boto3

def lambda_handler(event, context):
    client = boto3.client('ec2')


    response = client.authorize_security_group_ingress(
        GroupId='sg-0ab290fe68b7139fb',
        IpPermissions=[
            {
                'FromPort': 22,
                'IpProtocol': 'tcp',
```

```
                'IpRanges': [
                    {
                        'CidrIp': event['ip'],
                        'Description': 'SSH access from another office',
                    },
                ],
                'ToPort': 22,
            },
        ],
    )

    print(response)
```

The following AWS Chatbot command invokes the Lambda method using the specified parameters.

```
@aws invoke allowlist-ip --payload {"ip": "192.0.2.0/24"}
```

Remove an IP rule from a security group

The example in this section uses the AWS CLI and a Lambda function to remove an IP rule from an existing Amazon Elastic Compute Cloud Security Group. This is helpful because it conveniently allows you to remove IP rules from Slack without having to access a console. For more information about security groups, see Security groups for your VPC in the *Amazon Virtual Private Cloud User Guide*.

In the following code example, the remove-ip Lambda function accepts the payload specified by the user and removes the IP address using revoke_security_group_ingress. This revokes access from the specified IP address. The Lambda code uses Python 3.8. For more information about the revoke_security_group_ingress method, see the EC2 section of the *Boto3 Docs 1.14.9 documentation*.

```
import boto3

def lambda_handler(event, context):
    client = boto3.client('ec2')


    response = client.revoke_security_group_ingress(
        GroupId='sg-0ab290fe68b7139fb',
        IpPermissions=[
            {
                'FromPort': 22,
                'IpProtocol': 'tcp',
                'IpRanges': [
                    {
                        'CidrIp': event['ip'],
                        'Description': 'SSH access from another office',
                    },
                ],
                'ToPort': 22,
            },
        ],
    )

    print(response)
```

The following AWS Chatbot command invokes the Lambda method using the specified parameters.

```
@aws invoke remove-ip --payload {"ip": "192.0.2.0/24"}
```

Change the capacity of an Amazon EC2 Auto Scaling group

This code example shows how to change the capacity of an Amazon EC2 Auto Scaling group. For more information about Auto Scaling groups, see the Auto Scaling Groups section of the *Amazon EC2 Auto Scaling User Guide*.

In the following code example, the capacity value is passed into the Lambda function and the desired capacity is programmatically set using the set_desired_capacity method. For more information about this method, see the AutoScaling section of the *Boto3 Docs 1.14.1 documentation*. The Lambda code uses Python 3.8.

```
import boto3

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

    response = client.set_desired_capacity(
        AutoScalingGroupName='myGroup',
        DesiredCapacity= event['desired-capacity'],
        HonorCooldown=True,
    )

    print(response)
```

The following AWS Chatbot command invokes the autoscaling-lambda Lambda function with a payload that specifies a capacity of 50.

```
@aws invoke autoscaling-lambda --payload {"desired-capacity": 50}
```

Approve an AWS CodePipeline action

The code example in this section demonstrates how you can use a Lambda function to manually approve a pipeline action. This function enables you to approve or reject a pipeline action from Slack 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. There are steps at the end of this tutorial to delete the resources you created.

You perform the following steps in this tutorial:

1. Create a Lambda function.
2. Configure a CloudWatch alarm on the **Errors** metric of your Lambda function to post a message to your Amazon Simple Notification Service (Amazon SNS) topic when AWS Lambda emits error metrics to CloudWatch.
3. Configure a Slack channel with AWS Chatbot.
4. Subscribe your chatbot to the Amazon SNS topic to receive CloudWatch alarm notifications in Slack.
5. Test your Lambda function and CloudWatch alarm in Slack using the AWS Chatbot.

Topics

- Prerequisites (p. 26)
- Create a Lambda function (p. 27)
- Create an SNS topic (p. 27)
- Configure a CloudWatch alarm (p. 28)
- Configure a Slack client for AWS Chatbot (p. 28)
- Invoke a Lambda function from Slack (p. 29)
- Test the CloudWatch alarm (p. 30)
- Clean up resources (p. 30)

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 Invoke a Lambda function from Slack (p. 29).

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`.

```
exports.handler = async (event) => {
      const response = 'Hello World!'
      return response;
   };
```

7. Choose **Test**.
8. In **Event Name**, enter: **`myHelloWorld`**
9. Choose **Create**.
10. Choose **Test** and then verify that the **Execution results** tab displays Response: "Hello World!"
11. Choose **Save**.

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 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. **Name** - **`myHelloWorldNotifications`**
   b. **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.

Configure a CloudWatch alarm

A CloudWatch alarm can monitor your Lambda function and send 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. **Threshold** - Whenever errors is > **0**
    c. **Send notifications to** - **`myHelloWorldNotifications`**
    d. **Name** - **`myHelloWorld-alarm`**
    e. **Description** - **`Lambda myHelloWorld alarm`**
9. Choose **Create alarm**.

Configure a Slack client for AWS Chatbot

Configuring a Slack client using AWS Chatbot enables you 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**, enter the following details:
    - **Name** - **`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**.

Invoke a Lambda function from Slack

You can invoke Lambda functions from Slack using AWS CLI syntax after configuring a chatbot in AWS Chatbot. To interact with AWS Chatbot in Slack, enter **`@aws`** followed by an AWS CLI command. For more information, see Running AWS CLI commands from Slack channels (p. 17) 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.

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 output is shown following:

```
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 Configure a Slack client for AWS Chatbot (p. 28) procedure and verify that you have the correct permissions assigned to your role:

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

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 **Save**.
5. Return to your Slack channel and then enter the following command:

```
@aws invoke myHelloWorld
```

6. This triggers an error in your output, and it sends a CloudWatch alarm notification in Slack and an email indicating this. 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 check **ALARMS** under the **Alarm** menu on the left of the screen to confirm that your alarm has triggered.

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.

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**.

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. Events may be throttled if the number of events received exceeds 10 per second. Units: Count	
`EventsProcessed`	The number of event notifications received by AWS Chatbot.	

Metric	Description	
	Units: Count	
`UnsupportedEvents`	The number of unsupported events or messages attempted. For a full list of AWS services supported by AWS Chatbot, see *Using AWS Chatbot with other AWS services* (p. 13). Units: Count	
`MessageDeliverySuccess`	The number of messages successfully delivered to the chat client. Units: Count	
`MessageDeliveryFailure`	The number of messages that failed to deliver to the chat client. 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* (p. 59).

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 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 several 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

AWS Chatbot supports logging of the following actions as events in CloudTrail log files:

- DescribeSlackWorkspaces
- DescribeSlackChannels
- RedeemSlackOAuthCode
- GetSlackOAuthParameters

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 or AWS Identity and Access Management (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 (p. 38)
- Identity and Access Management for AWS Chatbot (p. 39)
- Compliance validation for AWS Chatbot (p. 57)
- Resilience in AWS Chatbot (p. 57)
- Infrastructure security in AWS Chatbot (p. 58)

Data protection in AWS Chatbot

AWS Chatbot conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all of the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data.

AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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.

- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon Simple Storage Service (Amazon S3).
- Never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with AWS Chatbot or other AWS services using the console, API, AWS Command Line Interface (AWS CLI), or AWS SDKs. Any data that you enter into AWS Chatbot or other services might get picked up for inclusion in diagnostic logs.
- When you provide a URL to an external server, don't include credentials information for validating your request to that server in the URL.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

> **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.

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 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 (p. 42).

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 employees 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 (p. 39).

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 (p. 49).

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 (p. 40)
- Resource-level permissions and AWS Chatbot (p. 40)
- Condition keys and AWS Chatbot (p. 41)
- Authorization based on AWS Chatbot tags (p. 41)
- Using temporary credentials with AWS Chatbot (p. 41)
- Service-linked roles (p. 41)
- Service roles (p. 41)
- Other policy types (p. 41)
- Troubleshooting AWS Chatbot identity and access (p. 42)

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 (p. 49).

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

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 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*.

Authorization based on AWS Chatbot tags

AWS Chatbot doesn't support tagging resources or controlling access based on tags.

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.

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 (p. 42)
- I'm not authorized to perform iam:PassRole (p. 42)
- I want to view my access keys (p. 43)
- I'm an administrator and want to allow others to access AWS Chatbot (p. 43)
- I want to allow people outside of my AWS account to access my AWS Chatbot resources (p. 43)

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, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies 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 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 asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access AWS Chatbot

To allow others to access AWS Chatbot, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS Chatbot.

To get started right away, see Creating Your First IAM Delegated User and Group 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 (p. 39).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

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 (p. 44)
- Customer managed IAM policies in AWS Chatbot (p. 46)

AWS managed IAM policies in AWS Chatbot

AWS Chatbot supports the following AWS managed IAM policies:

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

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": "*"
        }
    ]
}
```

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 (p. 49)
- Identity-based policy best practices (p. 50)
- Applying AWS Chatbot permissions to an IAM identity (p. 50)
- Allowing users to view their permissions (p. 51)

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

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used 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 are very powerful. They 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 Using AWS Managed Policies** – To start using AWS Chatbot quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition 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 (p. 52)
- Example: AWS Chatbot resource-level permission (p. 53)

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 (p. 49). 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.

Topics

- Service-linked role permissions for AWS Chatbot (p. 54)
- Enabling the service-linked role for AWS Chatbot (p. 55)
- Editing a service-linked role for AWS Chatbot (p. 55)
- Manually deleting the AWSServiceRoleForAWSChatbot service-linked role (p. 55)
- Supported regions for AWS Chatbot service-linked roles (p. 56)

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.

Service-linked role permissions for AWS Chatbot

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.

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.

IAM 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-Local)	ap-northeast-3	Yes

Region Name	Region Identity	Supported in AWS Chatbot
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
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes
AWS GovCloud (US)	us-gov-west-1	No

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) 1.0 or later. We recommend TLS 1.2 or later. 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 use these steps to troubleshoot.

Possible causes

- **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.
- **AWS Chatbot supports CloudWatch Events only for specific AWS services.**

  Services that support CloudWatch Events forwarding by Amazon SNS topics to Slack or Amazon Chime chat rooms include the following:
  - AWS Billing and Cost Management
  - AWS Config Events
  - Amazon GuardDuty Events
  - AWS Health Events
  - AWS Security Hub Events
  - AWS Systems Manager Events including the following:
    - AWS Systems Manager Configuration Compliance Events
    - AWS Systems Manager Maintenance Windows Events
    - AWS Systems Manager Parameter Store Events

  If you map Amazon SNS topics that are associated with any other AWS service, their CloudWatch Events notifications are sent to email or other standard SNS targets. They won't work with AWS Chatbot.

  Note
  Some AWS services support Amazon CloudWatch alarms for reporting and monitoring. You configure CloudWatch alarms using performance metrics from the services that are active in your account. Amazon Elastic Compute Cloud (EC2), with its collection of metrics, is a good example. 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.
- **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.
- **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.

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." (p. 60)

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 (p. 8).

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 (p. 3) and Identity and Access Management for AWS Chatbot (p. 39).
- **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 (p. 41) 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"
                    ]
                }
            }
        }
    ]
}
```

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.

update-history-change	update-history-description	update-history-date
AWS Chatbot general availability release (p. 62)	Content updates to reflect improvements made to AWS Chatbot during the preview period.	April 22, 2020
Updated CLI commands information. (p. 62)	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. (p. 62)	Addition of documentation for configuring support of commands for AWS services in Slack channels.	November 22, 2019
Announcement of AWS CodeSuite development tools support. (p. 62)	Addition of CodeSuite notification support information. Miscellaneous edits and fixes.	November 7, 2019
Enhanced troubleshooting information. (p. 62)	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. (p. 62)	AWS CloudTrail provides logging support for several newly integrated AWS Chatbot API actions.	August 7, 2019
AWS Chatbot is now in beta release. (p. 62)	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

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.