



User Guide

Amazon CodeGuru Reviewer



Amazon CodeGuru Reviewer: User Guide

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is CodeGuru Reviewer?	1
What kind of recommendations does CodeGuru Reviewer provide?	1
What languages and repositories can I use with CodeGuru Reviewer?	1
Accessing CodeGuru Reviewer	2
How CodeGuru Reviewer works	3
Setting up	4
Sign up for AWS	4
Sign up for an AWS account	4
Create a user with administrative access	5
Configure IAM permissions	6
Install or upgrade and then configure the AWS CLI	7
Create a repository	8
Getting started	9
Step 1: Get set up	9
Step 2: Associate a repository	9
Step 3: Get recommendations	10
About full repository analysis and incremental code reviews	10
Step 4: Provide feedback	12
Provide feedback using the CodeGuru Reviewer console	12
Provide feedback using pull request comments	12
Provide feedback using the CLI	13
Tutorial: monitor source code in GitHub	14
Step 1: Fork the repository	14
Step 2: Associate the forked repository	15
Step 3: Push a change to the code	15
Step 4: Create a pull request	16
Step 5: Review recommendations	17
Step 6: Clean up	18
Working with repository associations	20
Create a CodeCommit repository association	21
Create an CodeCommit repository association (console)	22
Create a CodeCommit repository association (CodeCommit console)	23
Create a CodeCommit repository association (AWS CLI)	24
Create a CodeCommit repository association (AWS SDKs)	25

Create a Bitbucket repository association	25
Create an AWS CodeCommit repository association (console)	26
Create a Bitbucket repository association (AWS CLI)	28
Create a Bitbucket repository association (AWS SDKs)	30
Create a GitHub or GitHub Enterprise Cloud repository association	30
Create a GitHub Enterprise Server repository association	33
GitHub Enterprise Server prerequisites	33
Create a GitHub Enterprise Server repository association (console)	34
Create a GitHub Enterprise Server repository association (AWS CLI)	36
Create a GitHub Enterprise Server repository association (AWS SDKs)	37
View all repository associations	38
View all associated repositories (console)	38
View all repository associations (AWS CLI)	38
Disassociate a repository	39
Disassociate a repository (console)	40
Disassociate a repository (AWS CLI)	41
Encrypting a repository association	42
Encrypt an associated repository using an AWS KMS key	43
Update how a repository association is encrypted	44
Tagging a repository association	44
Add a tag to an associated repository	45
View tags for an associated repository	50
Add or update tags for an associated repository	52
Remove tags from an associated repository	54
Working with code reviews	57
About full repository analysis and incremental code reviews	58
Suppress recommendations	60
Structure of the file	60
Steps to suppress recommendations	63
Cost impact of suppressing recommendations	66
Error handling for the analysis configuration file	66
Create code reviews	67
Get recommendations using full repository analysis	68
Get recommendations using incremental code reviews	70
Get recommendations using GitHub Actions	70
View all code reviews	74

Code reviews page	74
Navigate to repositories and pull requests	75
View code review details	75
Information in code review details	76
View code review details by using the AWS CLI	76
View recommendations and provide feedback	77
Create code reviews with GitHub Actions	78
Get recommendations using GitHub Actions	70
Create code reviews with GitHub Actions	70
Disassociate your CI/CD workflow	72
GitHub Actions code review examples	73
Product and service integrations	83
Recommendation types	85
Secrets detection	85
Secrets detection supported file types	86
Types of secrets detected	86
Security	90
Data protection	90
Captured data	92
Data retention	92
Data encryption	92
Traffic privacy	93
Identity and access management	93
Audience	94
Authenticating with identities	94
Managing access using policies	98
Overview of managing access	100
Using identity-based policies	104
Using tags to control access to associated repositories	116
CodeGuru Reviewer permissions reference	119
Troubleshooting	122
Compliance validation	123
VPC endpoints (AWS PrivateLink)	124
Considerations for CodeGuru Reviewer VPC endpoints	125
Creating an interface VPC endpoint for CodeGuru Reviewer	125
Infrastructure security	125

Logging and monitoring	127
Logging CodeGuru Reviewer API calls with AWS CloudTrail	127
CodeGuru Reviewer information in CloudTrail	127
Example: CodeGuru Reviewer log file entries	129
Monitoring CodeGuru Reviewer with CloudWatch	130
Monitoring recommendations with CloudWatch metrics	131
Monitoring CodeGuru Reviewer recommendations with CloudWatch alarms	132
Quotas	134
Repositories	134
Tags	134
CodeGuru Reviewer quotas for creating, deploying, and managing an API	135
Troubleshooting	137
Where can I check the status of a repository association?	137
Where can I check the status of a code review?	138
Where can I check the status of a third-party source provider connection?	138
My repository is in an associated state. Why don't I see recommendations?	138
Why did my association fail?	138
Why did my code review fail?	139
What if I disagree with the recommendation?	139
How do I suppress a recommendation?	140
The repository status has been associating for more than 5 minutes. What should I do?	140
The code review status has been Pending for more than 15 minutes. What should I do?	140
How do you access a repository if its owner is no longer available?	141
Can I use the same AWS CodeStar connection to access repositories in two different accounts?	141
I'm trying to connect to my third-party repositories. What is the difference between an app installation and a connection? Which one can be used to adjust permissions?	141
How do I know if CodeGuru Reviewer used my aws-codeguru-reviewer.yml file in a code review?	141
Why didn't my costs decrease when I used an aws-codeguru-reviewer.yml file?	144
Document history	145
AWS Glossary	150

What is Amazon CodeGuru Reviewer?

Amazon CodeGuru Reviewer is a service that uses program analysis and machine learning to detect potential defects that are difficult for developers to find and offers suggestions for improving your Java and Python code. This service has been released for general availability in several [Regions](#).

By proactively detecting code defects, CodeGuru Reviewer can provide guidelines for addressing them and implementing best practices to improve the overall quality and maintainability of your code base during the code review stage. For more information, see [How Amazon CodeGuru Reviewer works](#).

What kind of recommendations does CodeGuru Reviewer provide?

CodeGuru Reviewer doesn't flag syntactical mistakes, as these are relatively easy to find. Instead, CodeGuru Reviewer identifies more complex problems and suggests improvements related to recommendation types such as resource leak prevention or security analysis. Within each type are several detectors that CodeGuru Reviewer uses to analyze your code. For information about these detectors, see the [Amazon CodeGuru Reviewer Detector Library](#).

CodeGuru Reviewer also integrates with AWS Secrets Manager to use a secrets detector that finds unprotected secrets in your code. For more information, see [Secrets detection](#).

What languages and repositories can I use with CodeGuru Reviewer?

CodeGuru Reviewer is designed to work with Java and Python code repositories in the following source providers:

- [AWS CodeCommit](#)
- Bitbucket
- GitHub
- GitHub Enterprise Cloud
- GitHub Enterprise Server
- Amazon S3

Note

S3 repositories are only supported through GitHub Actions. For more information, see [Create code reviews with GitHub Actions](#).

If you use any of these source providers, you can integrate with CodeGuru Reviewer with just a few steps. After you associate a repository with CodeGuru Reviewer, you can [interact with recommendations in the CodeGuru Reviewer console](#). For incremental code reviews, you can also [see recommendations directly from inside pull requests](#) in your repository.

Accessing CodeGuru Reviewer

You can access CodeGuru Reviewer using any of the following methods:

- **Amazon CodeGuru Reviewer console** – <https://console.aws.amazon.com/codeguru/reviewer/>
- **AWS CLI** – For more information, see [Getting started with the AWS CLI](#) in the *AWS Command Line Interface User Guide*.
- **CodeGuru Reviewer API** – For more information, see the [Amazon CodeGuru Reviewer API Reference](#).
- **AWS SDKs** – For more information, see [Tools to Build on AWS](#).

How Amazon CodeGuru Reviewer works

Amazon CodeGuru Reviewer uses program analysis combined with machine learning models trained on millions of lines of Java and Python code from the Amazon code base and other sources. When you [associate CodeGuru Reviewer with a repository](#), CodeGuru Reviewer can find and flag code defects and suggest recommendations to improve your code. CodeGuru Reviewer provides actionable recommendations with a low rate of false positives and might improve its ability to analyze code over time based on user feedback.

You can associate CodeGuru Reviewer with a repository to allow CodeGuru Reviewer to provide recommendations. After you associate a repository with CodeGuru Reviewer, CodeGuru Reviewer automatically analyzes pull requests that you make, and you can choose to run repository analyses on the code in your branch to analyze all the code at any time.

If you want to suppress recommendations from CodeGuru Reviewer, you can create and add to the root directory of your repository an `aws-codeguru-reviewer.yml` file that lists files and directories to exclude from analysis. For more information, see [Suppress recommendations](#).

You can view recommendations from incremental code reviews and full repository analysis code reviews directly in the CodeGuru Reviewer console. You can also view recommendations from incremental code reviews as pull request comments in your repository. These recommendations address instances in which the code doesn't adhere to AWS SDK best practices, operations on concurrent data structures might not be thread safe, or resource closure might not be handled properly, among other things.

Developers can decide how to incorporate the recommendations from CodeGuru Reviewer and [provide feedback](#) to CodeGuru Reviewer about whether the recommendations were useful. This helps your team ensure code quality and improve their code practices in an organic, interactive way. At the same time, it improves the quality of recommendations CodeGuru Reviewer provides for your code, making CodeGuru Reviewer increasingly effective in future analyses.

Setting up Amazon CodeGuru Reviewer

Complete the tasks in this section to set up Amazon CodeGuru Reviewer for the first time. If you already have an AWS account, know the repository that contains the source code you want reviewed, and prefer to use the defaults for IAM, skip ahead to [Getting started](#).

You should know the following about Amazon Web Services (AWS):

- When you sign up for AWS, your AWS account automatically has access to all services in AWS, including CodeGuru Reviewer. However, you are charged only for the services that you use.
- With CodeGuru Reviewer, you pay for the size of each of your associated repositories measured in lines of code. For more information, see [Amazon CodeGuru pricing](#).

Note

You can suppress recommendations from CodeGuru Reviewer, which reduces the number of lines of code analyzed and, by extension, might reduce costs. For more information, see [Suppress recommendations](#).

Topics

- [Sign up for AWS](#)
- [Configure IAM permissions for Amazon CodeGuru Reviewer](#)
- [Install or upgrade and then configure the AWS CLI](#)
- [Create a repository for your source code](#)

Sign up for AWS

If you have an AWS account and have set up an administrative user already, skip to the next section, [Configure IAM permissions](#).

If you don't have an AWS account, you can use the following procedure to create one. If you are a new Amazon CodeGuru Reviewer customer, you can sign up for a 90-day free trial.

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

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

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

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

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User Guide*.

Configure IAM permissions for Amazon CodeGuru Reviewer

When you create an AWS account, you get a single sign-in identity that has complete access to all of the AWS services and resources in the account. This identity is called the AWS account *root user*. Signing in to the AWS console using the email address and password that you used to create the account gives you complete access to all of the AWS resources in your account.

We strongly recommend that you *not* use the root user for everyday tasks, even the administrative ones. Instead, create an administrative user with the least privileges needed. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. For more information, see [Security best practices in IAM](#) in the *IAM User Guide*.

See the following topics for information about permissions required for CodeGuru Reviewer and how to add them.

- [Authenticating with identities](#)
- [Using identity-based policies for CodeGuru Reviewer](#)
- [Amazon CodeGuru Reviewer permissions reference](#)

Install or upgrade and then configure the AWS CLI

To call Amazon CodeGuru Reviewer commands from the AWS Command Line Interface (AWS CLI) on a local development machine, you must install the AWS CLI.

Note

You cannot create a repository association for a GitHub repository using the AWS CLI. You can use the AWS CLI to create a repository association for all other supported repository types. For more information, see [Working with repository associations in Amazon CodeGuru Reviewer](#).

If you have an older version of the AWS CLI installed, we recommend you upgrade it so the CodeGuru Reviewer commands are available. To check the version, use the `aws --version` command.

To install and configure the AWS CLI

1. Follow the instructions in [Getting started with the AWS CLI](#) to install or upgrade the AWS CLI.
2. To configure the AWS CLI, see [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

Important

When you configure the AWS CLI, you are prompted to specify an AWS Region. Choose one of the supported Regions listed in [Amazon CodeGuru Reviewer endpoints and quotas](#) in the *AWS General Reference*.

3. To verify the installation or upgrade, call the following command from the AWS CLI.

```
aws codeguru-reviewer help
```

If successful, this command displays a list of available CodeGuru Reviewer commands.

Create a repository for your source code

Before you use Amazon CodeGuru Reviewer to create code reviews, the source code that you want to analyze must be set up in a supported repository type. CodeGuru Reviewer supports AWS CodeCommit, Bitbucket, GitHub, and GitHub Enterprise Server. If you use Bitbucket or GitHub Enterprise Server, you must also create a connection to your repository using CodeConnections. For more information, see [What are connections?](#) in the *AWS Developer Tools User Guide*.

If you want to suppress recommendations from CodeGuru Reviewer, you can create and add to the root directory of your repository an `aws-codeguru-reviewer.yml` file that lists files and directories to exclude from analysis. For more information, see [Suppress recommendations](#).

After you know the repository type and the name of your repository, you need to create a repository association. For GitHub repositories, you can create a repository association using only the CodeGuru Reviewer console. For the other supported repository types, you can use the console, AWS CLI, or Amazon CodeGuru Reviewer SDK to create a repository association. For more information, see [Working with repository associations in Amazon CodeGuru Reviewer](#).

Getting started with CodeGuru Reviewer

In this section, you learn what you need to do to get started with Amazon CodeGuru Reviewer so it can start to analyze your source code and provide code reviews. To use CodeGuru Reviewer to create code reviews with example source code in a real repository, see [Tutorial: monitor source code in a GitHub repository](#).

Topics

- [Step 1: Get set up](#)
- [Step 2: Associate a repository](#)
- [Step 3: Get recommendations](#)
- [Step 4: Provide feedback](#)

Step 1: Get set up

Before you get started, you must prepare by running through the steps in [Setting up Amazon CodeGuru Reviewer](#).

Step 2: Associate a repository

You must create a repository association to grant CodeGuru Reviewer access to read your source code and create notifications on your repository. The notifications initiate an analysis on the updated source code every time you create a pull request. When you create your repository association, CodeGuru Reviewer automatically creates a full repository analysis code review. You must initiate future full repository analysis code reviews. For more information, see [Working with repository associations in Amazon CodeGuru Reviewer](#).

Note

The source code reviewed by CodeGuru Reviewer is not stored. For more information, see [the section called "Captured data"](#).

To create a repository association, choose one of the following.

- If your repository type is AWS CodeCommit, see [Create a CodeCommit repository association](#).
- If your repository type is Bitbucket, see [Create a Bitbucket repository association](#).
- If your repository type is GitHub or GitHub Enterprise Cloud, see [Create a GitHub or GitHub Enterprise Cloud repository association](#).
- If your repository type is GitHub Enterprise Server, see [Create a GitHub Enterprise Server repository association](#).

Step 3: Get recommendations

CodeGuru Reviewer uses code reviews to provide [different kinds of recommendations](#) to help improve your code. These recommendations are focused on best practices and resolving potential defects in code that are difficult for developers to find. After CodeGuru Reviewer completes an incremental code review or a full repository analysis code review, you can view recommendations. You can then choose whether to incorporate the recommendations, and you can provide feedback about whether the recommendations were helpful.

Note

We recommend that you use both CodeGuru Reviewer and traditional peer review processes during the code review stage. Using a combination of code review processes helps to identify more issues before they reach production.

For more information, see the following topics:

- [View code review details](#)
- [Get recommendations using full repository analysis](#)
- [Get recommendations using incremental code reviews](#)
- [Get recommendations using GitHub Actions and a CI/CD workflow](#)

About full repository analysis and incremental code reviews

You can receive recommendations in code reviews by creating a full repository analysis or submitting a pull request. After you associate a repository, CodeGuru Reviewer automatically creates a full repository analysis code review, and every pull request in that repository creates an

incremental code review. You can also choose to create additional full repository analysis code reviews.

Type of code review	Is the review automatic after I associate the repository?	Where can I see recommendations?	What code is reviewed?
Full repository analysis	Your first full repository analysis is done automatically when you associate your repository. After that, you must request a full repository analysis in the CodeGuru Reviewer console or by using the AWS CLI or AWS SDK.	In the CodeGuru Reviewer console, or by using the AWS CLI or AWS SDK.	All the code in the branch is reviewed.
Incremental code review	Yes. After associating the repository, every time you do a pull request there is a code review.	In the CodeGuru Reviewer console, in the AWS CLI or AWS SDK, or in pull request comments in the repository source provider.	The code that is changed in the pull request is reviewed.
GitHub Actions code review in a CI/CD workflow	Yes. After enabling CodeGuru Reviewer on your GitHub repository, for every push, pull, or scheduled repository scan there is a code review.	In the GitHub Security tab.	The code that is changed in the push, pull, or scheduled repository scan.

Step 4: Provide feedback

CodeGuru Reviewer is based on program analysis and machine learning models, so it's constantly improving. To assist in the machine learning process and enhance the experience with CodeGuru Reviewer, you can provide feedback on the recommendations in the **Code reviews** page on the CodeGuru Reviewer console. You can also provide feedback directly in the pull requests to indicate whether they were helpful to you.

Your feedback and comments are shared with CodeGuru Reviewer. This can help CodeGuru Reviewer to improve its models and become more helpful to you and others in the future. When you provide feedback, your code is not shared. For more information, see [Captured data in CodeGuru Reviewer](#).

Note

The source code reviewed by CodeGuru Reviewer is not stored. For more information, see [Captured data in CodeGuru Reviewer](#).

Topics

- [Provide feedback using the CodeGuru Reviewer console](#)
- [Provide feedback using pull request comments](#)
- [Provide feedback using the CLI](#)

Provide feedback using the CodeGuru Reviewer console

You can provide feedback for recommendations on incremental code reviews or full repository analysis code reviews [using the Code reviews page](#) of the CodeGuru Reviewer console. Choose the name of a code review to view details and recommendations from that code review. Then choose the thumbs-up or thumbs-down icon under each recommendation to indicate whether the recommendation was helpful.

Provide feedback using pull request comments

You can also provide feedback for incremental code reviews by replying to comments in pull requests, without leaving your repository context. In AWS CodeCommit you can view the recommendations on the **Activity** or **Changes** tab. A thumbs-up or thumbs-down icon is provided

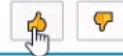
next to comments made by CodeGuru Reviewer. Choose a thumbs-up icon if the recommendation was helpful and a thumbs-down icon if it wasn't. In other repository source providers, you can reply to a comment made by CodeGuru Reviewer, and include either a thumbs-up or thumbs-down emoji in your comment to indicate whether it was helpful or not. For more information, see [Using emoji](#) on the GitHub website and [Use symbols, emojis, and special characters](#) on the Bitbucket website.

The following image shows an example of feedback in CodeCommit.

AmazonCodeGuruBot Bot commented Just now

This code might not produce accurate results if the operation returns paginated results instead of all results. Consider adding another call to check for additional results.

Reply to this recommendation and choose one of the reaction buttons to provide feedback.



This makes sense! 👍

Provide feedback using the CLI

You can also use the AWS CLI or the AWS SDK to provide feedback on recommendations. If you have the code review ARN, you can call [ListRecommendations](#). This returns the list of all recommendations for a completed code review. You can then use [PutRecommendationFeedback](#) to store reactions and send feedback as UTF-8 text code for emojis.

To view the feedback that you have submitted, call [DescribeRecommendationFeedback](#) using the RecommendationId. To view feedback from all users, call [ListRecommendationFeedback](#) by using a filter on RecommendationIds and UserIds. For more information, see the [Amazon CodeGuru Reviewer API Reference](#).

Tutorial: monitor source code in a GitHub repository

In this tutorial, you learn how to configure Amazon CodeGuru Reviewer to monitor source code so that it can create recommendations that improve the code.

You work with an actual suboptimal example application in a GitHub repository as a test case. After you associate the repository with CodeGuru Reviewer, you create a code change and submit a pull request that triggers program analysis.

Because the example application contains intentional inefficiencies, CodeGuru Reviewer creates recommendations about how to make it better. You learn how to review the recommendations and then how to provide feedback about them. Customer feedback from code reviews helps improve CodeGuru Reviewer recommendations over time.

To run this tutorial, you must have a [GitHub](#) account.

Note

- This tutorial creates code reviews that might result in charges to your AWS account. For more information, see [Amazon CodeGuru Pricing](#).
- Don't use the example code in production. It's intentionally problematic and intended for demonstration purposes only.

Step 1: Fork the repository

Fork the example application repository so you can create a pull request on it.

1. Log in to GitHub and navigate to the <https://github.com/aws-samples/amazon-codeguru-reviewer-sample-app> example application repository.
2. Choose **Fork** to fork the example application to your GitHub account.

 [aws-samples](#) / [amazon-codeguru-reviewer-sample-app](#)

 Unwatch ▾ 2

 Star 0

 Fork 0

Step 2: Associate the forked repository

Create a repository association with the example application's repository so that CodeGuru Reviewer listens to it for pull requests.

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. Choose **Associate repository**.
3. Make sure **GitHub or GitHub Enterprise Cloud** is selected, and then choose **Connect to your GitHub account**.
4. To allow CodeGuru Reviewer to access your account, choose **Authorize aws-codesuite**. If prompted, confirm your GitHub password.
5. Select the **amazon-codeguru-reviewer-sample-app** repository, and then choose **Associate**.

CodeGuru Reviewer is now associated with the repository and listening for pull requests.

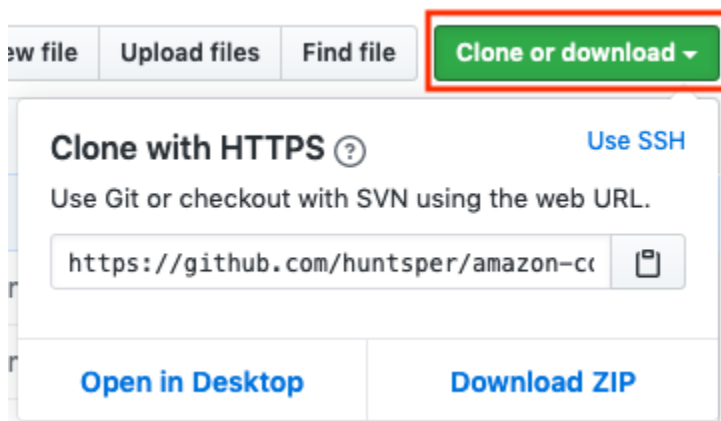
Step 3: Push a change to the code

Push a change to the example application's code. Later in this tutorial, you create a pull request for this change.

1. Run the following Git command to clone the forked repository, replacing USER_ID with your actual GitHub user ID.

```
git clone https://github.com/USER_ID/amazon-codeguru-reviewer-sample-app.git
```

You can get the clone URL by choosing **Clone or download**.



 Note

If you access your GitHub repositories using SSH, use the SSH URL instead of the HTTPS URL shown in this step.

2. Check out a new branch using the following command.

```
cd amazon-codeguru-reviewer-sample-app
git checkout -b dev
```

3. Copy the Java class at `src/main/java/com/shipmentEvents/handlers/EventHandler.java` into `src/main/java/com/shipmentEvents/demo`.

```
cp src/main/java/com/shipmentEvents/handlers/EventHandler.java src/main/java/com/shipmentEvents/demo/
```

GitHub and CodeGuru Reviewer treat `EventHandler.java` as a new file.

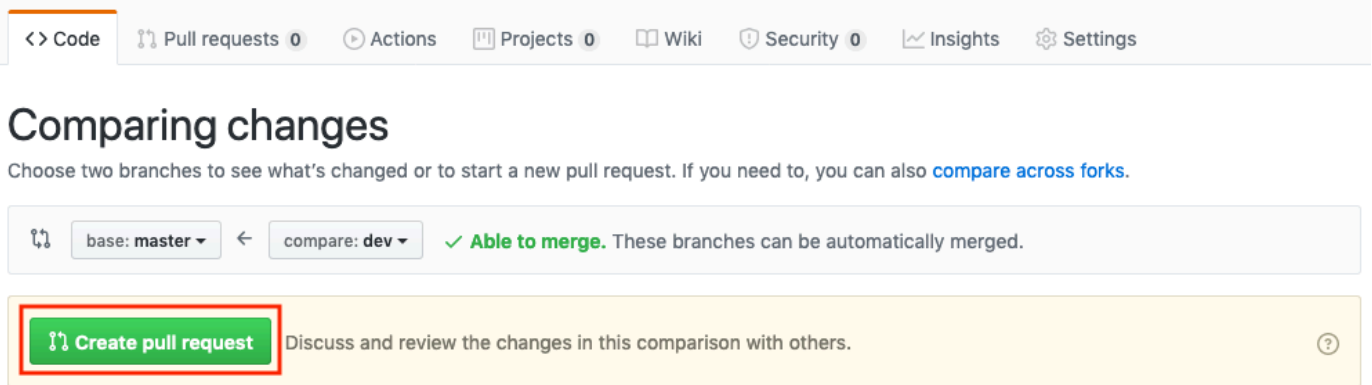
4. Push your changes to the example application's repository.

```
git add --all
git commit -m 'new demo file'
git push --set-upstream origin dev
```

Step 4: Create a pull request

Create a pull request for CodeGuru Reviewer to review.

1. In your forked GitHub repository, choose **New pull request**.
2. On the left side of the comparison (**base**), select **USER_ID/amazon-codeguru-reviewer-sample-app**, where **USER_ID** is your GitHub user ID. Leave the branch at **master**.
3. On the right side of the comparison (**compare**), change the branch to **dev**. The branches should show as **Able to merge**.

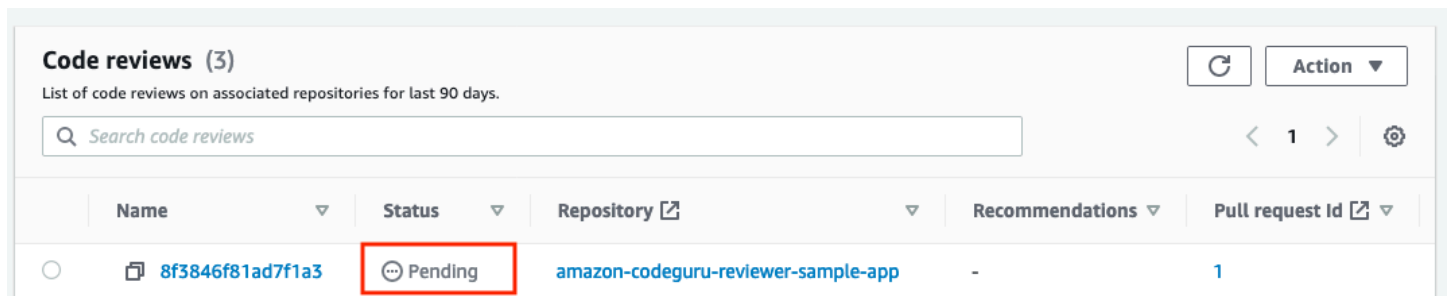


4. Choose **Create pull request**, then choose **Create pull request** again.

Step 5: Review recommendations

After a few minutes, CodeGuru Reviewer issues recommendations on the same GitHub page where the pull request was created. You can check the status of the code review in CodeGuru Reviewer console.

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, expand **Reviewer** and choose **Code reviews**.
3. After a code review is complete, choose it to view its details.



When the code review is complete and the recommendations appear in GitHub, you can provide feedback on the recommendations using the thumbs up or thumbs down icon. Any positive or negative feedback is used to help improve the performance of CodeGuru Reviewer so that recommendations get better over time.

```
src/main/java/com/shipmentEvents/demo/EventHandler.java
```

```
76 +  
77 +      long expirationTime = System.currentTimeMillis() + Duration.of  
78 +      while(System.currentTimeMillis() < expirationTime) {  
79 +          if (s3Client.doesObjectExist(Constants.SUMMARY_BUCKET, sum
```



6 minutes ago Author Owner

Recommendation generated by Amazon CodeGuru Reviewer. You can accept or reject this recommendation by replying to the comment or by reacting with a thumbs up or thumbs down emoji.

This code appears to be waiting for a resource before it runs. Consider using the `waiters` feature to help improve efficiency. Consider using `ObjectExists` or `ObjectNotExists`. For more information, see <https://aws.amazon.com/blogs/developer/waiters-in-the-aws-sdk-for-java/>

Pick your reaction



Reply...

Resolve conversation

Step 6: Clean up

After you're finished with this tutorial, clean up your resources.

1. In your GitHub fork of **amazon-codeguru-reviewer-sample-app**, go to **Settings**, and then choose **Delete this repository**. Follow the instructions to delete the forked repository.
2. Delete your clone of the forked repository, for example, `rm -rf amazon-codeguru-reviewer-sample-app`.
3. In the CodeGuru Reviewer console, select the example repository and choose **Disassociate repository**.

Repositories (1)

Pull requests on associated repositories trigger a review of the updated code.

Manage tags

Disassociate repository

Associate repository

Find resources

All repositories (ex...

1<1>

match

Association ID	Repository Name	Status	Source	Last updated
68656e17-9252-428c-b870-0cfc289cdf52	cgr_test	Associated	GitHub	30 Oct 2020 09:54:44 AM GMT-0700

Working with repository associations in Amazon CodeGuru Reviewer

Amazon CodeGuru Reviewer requires a repository that contains the source code you want it to review. To add a repository with your source code to CodeGuru Reviewer, you create a *repository association*.

Immediately after you create the repository association, its status is **Associating**. A repository association with this status is doing the following.

- Setting up pull request notifications. This is required for pull requests to initiate a CodeGuru Reviewer review. For GitHub, GitHub Enterprise Server, and Bitbucket repositories, the notifications are webhooks created in your repository to initiate CodeGuru Reviewer reviews. If you delete these webhooks, reviews of code in your repository cannot be initiated.
- Setting up source code access. This is required for CodeGuru Reviewer to securely clone the code in your repository.

When the pull request notifications, source code access, and creation of required permissions are complete, the status changes to **Associated**. The association is now complete and CodeGuru Reviewer performs its first full scan of the repository. You can later create incremental code reviews or full repository analysis code reviews to get recommendations. For more information, see [About full repository analysis and incremental code reviews](#).

CodeGuru Reviewer supports associations with repositories from the following source providers:

- AWS CodeCommit
- Bitbucket
- GitHub and GitHub Enterprise Cloud (These are listed together because you work with them identically using CodeGuru Reviewer.)
- GitHub Enterprise Server

Note

The source code reviewed by CodeGuru Reviewer is not stored. For more information, see [the section called “Captured data”](#).

Topics

- [Create an AWS CodeCommit repository association in Amazon CodeGuru Reviewer](#)
- [Create a Bitbucket repository association in Amazon CodeGuru Reviewer](#)
- [Create a GitHub or GitHub Enterprise Cloud repository association in Amazon CodeGuru Reviewer](#)
- [Create a GitHub Enterprise Server repository association in Amazon CodeGuru Reviewer](#)
- [View all repository associations in CodeGuru Reviewer](#)
- [Disassociate a repository in CodeGuru Reviewer](#)
- [Encrypting a repository association in Amazon CodeGuru Reviewer](#)
- [Tagging a repository association in Amazon CodeGuru Reviewer](#)

Create an AWS CodeCommit repository association in Amazon CodeGuru Reviewer

You can create an AWS CodeCommit repository association using the Amazon CodeGuru Reviewer console, the AWS CodeCommit console, the AWS CLI, or the CodeGuru Reviewer SDK. Before you create a CodeCommit repository association, you must have a CodeCommit repository in the same AWS account and Region in which you want your CodeGuru Reviewer code reviews. For more information, see [Create an AWS CodeCommit repository](#) in the *AWS CodeCommit User Guide*.

Topics

- [Create a CodeCommit repository association \(CodeGuru Reviewer console\)](#)
- [Create a CodeCommit repository association \(CodeCommit console\)](#)
- [Create a CodeCommit repository association \(AWS CLI\)](#)
- [Create a CodeCommit repository association \(AWS SDKs\)](#)

Create a CodeCommit repository association (CodeGuru Reviewer console)

To create a CodeCommit repository association

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Choose **Associate repository and run analysis**.
4. Choose **AWS CodeCommit**.
5. From **Repository location**, choose the name of your CodeCommit repository that contains the source code you want CodeGuru Reviewer to analyze.
6. (Optional) Expand **Encryption key - optional** to use your own AWS Key Management Service key (KMS key) to encrypt your associated repository. For more information, see [Encrypting a repository association in Amazon CodeGuru Reviewer](#).
 - a. Select **Customize encryption settings (advanced)**.
 - b. Do one of the following:
 - If you already have a KMS key that you manage, enter its Amazon Resource Name (ARN). For information about finding the ARN of your key using the console, see [Finding the key ID and key ARN](#) in the *AWS Key Management Service Developer Guide*.
 - If you want to create a KMS key, choose **Create an AWS KMS key** and follow the steps in the AWS KMS console. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.
7. In **Run a repository analysis**, specify information for your associated repository's first full scan. This scan generates your repository's initial code review. For more information, see [Get recommendations using full repository analysis](#).
 - a. From **Source branch**, choose the branch to use.
 - b. (Optional) In **Code review name**, type a name for your code review.
 - c. (Optional) Expand **Analysis configuration file - optional** to download a sample `aws-codeguru-reviewer.yml` file to use as a template. Modify the file and upload it to the root directory of your repository. For more information about the analysis configuration file, see [Suppress recommendations](#).

Run a repository analysis [Info](#)

When you associate a repository, CodeGuru Reviewer performs a full code review on the repository. After the repository association is created, code in all pull requests are reviewed and you can create new repository code reviews.

Source branch

Select a source branch for CodeGuru Reviewer to scan.

Code review name - *optional*

Enter your code review name.

▼ Analysis configuration file - *optional*

Suppress recommendations from CodeGuru Reviewer.

Create an `aws-codeguru-reviewer.yml` file that uses glob expressions to exclude files and directories in your repository for code reviews. Then add this file to the root directory of your repository. [Learn more](#) [↗](#)

[Download sample aws-codeguru-reviewer.yml file](#)

8. (Optional) Expand **Tags** to add one or more tags to your repository association. For more information, see [Tagging a repository association in Amazon CodeGuru Reviewer](#).
 - a. Choose **Add new tag**.
 - b. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
 - c. (Optional) To add another tag, choose **Add new tag**.
9. Choose **Associate repository and run analysis**. On the **Repositories** page, the **Status** is **Associating**. When the association is complete, the status changes to **Associated** and a full repository analysis begins. Refresh the page to check for the status change.

Create a CodeCommit repository association (CodeCommit console)

You can [connect to CodeGuru Reviewer directly from the CodeCommit console](#). This allows you to create a CodeCommit repository association with CodeGuru Reviewer without leaving your CodeCommit repository context.

Create a CodeCommit repository association (AWS CLI)

For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section of the AWS CLI Command Reference](#).

To create a CodeCommit repository association

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews and in which your CodeCommit repository exists. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **associate-repository** command specifying the name of the CodeCommit repository you want to associate.

```
aws codeguru-reviewer associate-repository --repository CodeCommit={Name=my-codecommit-repo}
```

3. If successful, this command outputs a [RepositoryAssociation](#) object.

```
{
  "RepositoryAssociation": {
    "AssociationId": "repository-association-uuid",
    "Name": "my-codecommit-repo",
    "LastUpdatedTimeStamp": 1595634764.029,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1595634764.029,
    "Owner": "123456789012",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid",
  }
}
```

4. When the **associate-repository** command succeeds, the status in the returned output is **Associating**. When the association is complete, the status changes to **Associated** and you can

create a pull request or a full repository analysis to get recommendations. You can check your repository association's status using the `describe-repository` command with its Amazon Resource Name (ARN).

```
aws codeguru-reviewer describe-repository-association --association-arn
arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-
association-uuid
```

5. If successful, this command outputs a [RepositoryAssociation](#) object which shows its status.

```
{
  "RepositoryAssociation": {
    "AssociationId": "repository-association-uuid",
    "Name": "my-codecommit-repo",
    "LastUpdatedTimeStamp": 1595634764.029,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1595634764.029,
    "Owner": "123456789012",
    "State": "Associated",
    "StateReason": "Pull Request Notification configuration successful",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid"
  }
}
```

Create a CodeCommit repository association (AWS SDKs)

To create a CodeCommit repository association with the AWS SDKs, use the `AssociateRepository` API. For more information, see [AssociateRepository](#) in the *Amazon CodeGuru Reviewer API Reference*.

Create a Bitbucket repository association in Amazon CodeGuru Reviewer

You can create a Bitbucket repository association using the Amazon CodeGuru Reviewer console, the AWS CLI, or the CodeGuru Reviewer SDK. Before you create a Bitbucket repository association, you must have a Bitbucket repository and you must create a connection to it using the Developer Tools console. For more information, see [Create a connection](#) in the *Developer Tools User Guide*.

For information about creating a Bitbucket repository, see [Create a Git repository](#) on the Bitbucket website.

Topics

- [Create a Bitbucket repository association \(console\)](#)
- [Create a Bitbucket repository association \(AWS CLI\)](#)
- [Create a Bitbucket repository association \(AWS SDKs\)](#)

Create a Bitbucket repository association (console)

To create a Bitbucket repository association

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Choose **Associate repository and run analysis**.
4. Choose **Bitbucket**.
5. From **Connect to Bitbucket (with CodeConnections)**, choose the connection you want to use. If you don't have a connection, choose **Create a Bitbucket connection** to create one in the Developer Tools console. For more information, see [Create a connection](#) in the *Developer Tools User Guide*.
6. From **Repository location**, choose the name of your Bitbucket repository that contains the source code you want CodeGuru Reviewer to analyze.
7. (Optional) Expand **Encryption key - optional** to use your own AWS Key Management Service key (KMS key) to encrypt your associated repository. For more information, see [Encrypting a repository association in Amazon CodeGuru Reviewer](#).
 - a. Select **Customize encryption settings (advanced)**.
 - b. Do one of the following:
 - If you already have a KMS key that you manage, enter its Amazon Resource Name (ARN). For information about finding the ARN of your key using the console, see [Finding the key ID and key ARN](#) in the *AWS Key Management Service Developer Guide*.

- If you want to create a KMS key, choose **Create an AWS KMS key** and follow the steps in the AWS KMS console. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.
8. In **Run a repository analysis**, specify information for your associated repository's first full scan. This scan generates your repository's initial code review. For more information, see [Get recommendations using full repository analysis](#).
- a. From **Source branch**, choose the branch to use.
 - b. (Optional) In **Code review name**, type a name for your code review.
 - c. (Optional) Expand **Analysis configuration file - optional** to download a sample `aws-codeguru-reviewer.yml` file to use as a template. Modify the file and upload it to the root directory of your repository. For more information about the analysis configuration file, see [Suppress recommendations](#).

Run a repository analysis [Info](#)

When you associate a repository, CodeGuru Reviewer performs a full code review on the repository. After the repository association is created, code in all pull requests are reviewed and you can create new repository code reviews.

Source branch
Select a source branch for CodeGuru Reviewer to scan.

Code review name - optional
Enter your code review name.

▼ Analysis configuration file - optional
Suppress recommendations from CodeGuru Reviewer.

Create an `aws-codeguru-reviewer.yml` file that uses glob expressions to exclude files and directories in your repository for code reviews. Then add this file to the root directory of your repository. [Learn more](#) [↗](#)

 **Download sample aws-codeguru-reviewer.yml file**

9. (Optional) Expand **Tags** to add one or more tags to your repository association. For more information, see [Tagging a repository association in Amazon CodeGuru Reviewer](#).

- a. Choose **Add new tag**.
 - b. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
 - c. (Optional) To add another tag, choose **Add new tag**.
10. Choose **Associate repository and run analysis**. On the **Repositories** page, the **Status** is **Associating**. When the association is complete, the status changes to **Associated** and a full repository analysis begins. Refresh the page to check for the status change.

Create a Bitbucket repository association (AWS CLI)

For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section of the AWS CLI Command Reference](#)

To create a Bitbucket repository association

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **associate-repository** command specifying the owner (or user name) of your Bitbucket account, the name of your repository, and the Amazon Resource Name (ARN) of your connection.

```
aws codeguru-reviewer associate-repository --repository
  Bitbucket="{Owner=bitbucket-user-name, Name=repository-name, \
  ConnectionArn=arn:aws:codeconnections:us-west-2:123456789012:connection/
  repository-uuid }"
```

3. If successful, this command outputs a [RepositoryAssociation](#) object.

```
{
  "RepositoryAssociation": {
    "ProviderType": "Bitbucket",
    "Name": "repository-name",
    "LastUpdatedTimeStamp": 1595886585.96,
```

```

    "AssociationId": "repository_association_uuid",
    "CreatedTimeStamp": 1595886585.96,
    "ConnectionArn": "arn:aws:codeconnections:us-
west-2:123456789012:connection/connection_uuid",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid",
    "Owner": "bitbucket-user-name"
  }
}

```

4. When the **associate-repository** command succeeds, the status in the returned output is **Associating**. When the association is complete, the status changes to **Associated** and you can create a pull request or a full repository analysis to get recommendations. You can check your repository association's status using the **describe-repository** command with its Amazon Resource Name (ARN).

```

aws codeguru-reviewer describe-repository-association --association-arn
arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-
association-uuid

```

5. If successful, this command outputs a [RepositoryAssociation](#) object which shows its status.

```

{
  "RepositoryAssociation": {
    "ProviderType": "Bitbucket",
    "Name": "repository-name",
    "LastUpdatedTimeStamp": 1595886585.96,
    "AssociationId": "repository_association_uuid",
    "CreatedTimeStamp": 1595886585.96,
    "ConnectionArn": "arn:aws:codeconnections:us-
west-2:123456789012:connection/connection_uuid",
    "State": "Associated",
    "StateReason": "\"Pull Request Notification configuration successful\"",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid",
    "Owner": "bitbucket-user-name"
  }
}

```

Create a Bitbucket repository association (AWS SDKs)

To create a Bitbucket repository association with the AWS SDKs, use the `AssociateRepository` API. For more information, see [AssociateRepository](#) in the *Amazon CodeGuru Reviewer API Reference*.

Create a GitHub or GitHub Enterprise Cloud repository association in Amazon CodeGuru Reviewer

You can create a GitHub or GitHub Enterprise Cloud repository association using the Amazon CodeGuru Reviewer console. You cannot create a GitHub or GitHub Enterprise Cloud repository association using the AWS CLI or the CodeGuru Reviewer SDK. Before you create a GitHub or GitHub Enterprise Cloud repository association, you must have a GitHub or GitHub Enterprise Cloud repository.

Note

We recommend creating a new GitHub user (for example, *MyCodeGuruUser*) and using that user to provide CodeGuru Reviewer with access to your GitHub repositories. This ensures that CodeGuru Reviewer posts comments on behalf of a unique user. This helps avoid confusion and make the account more transferable, so that it doesn't belong to a single person who might not always be available to maintain it.

To create a GitHub repository association

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Choose **Associate repository and run analysis**.
4. Choose **GitHub or GitHub Enterprise Cloud**.
5. If you are not connected to GitHub, choose **Connect to GitHub or GitHub Enterprise Cloud** and follow the prompts to connect.
6. From **Repository location**, choose the name of your GitHub repository that contains the source code you want CodeGuru Reviewer to analyze.

7. (Optional) Expand **Encryption key - optional** to use your own AWS Key Management Service key (KMS key) to encrypt your associated repository. For more information, see [Encrypting a repository association in Amazon CodeGuru Reviewer](#).
 - a. Select **Customize encryption settings (advanced)**.
 - b. Do one of the following:
 - If you already have a KMS key that you manage, enter its Amazon Resource Name (ARN). For information about finding the ARN of your key using the console, see [Finding the key ID and key ARN](#) in the *AWS Key Management Service Developer Guide*.
 - If you want to create a KMS key, choose **Create an AWS KMS key** and follow the steps in the AWS KMS console. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.
8. In **Run a repository analysis**, specify information for your associated repository's first full scan. This scan generates your repository's initial code review. For more information, see [Get recommendations using full repository analysis](#).
 - a. From **Source branch**, choose the branch to use.
 - b. (Optional) In **Code review name**, type a name for your code review.
 - c. (Optional) Expand **Analysis configuration file - optional** to download a sample `aws-codeguru-reviewer.yml` file to use as a template. Modify the file and upload it to the root directory of your repository. For more information about the analysis configuration file, see [Suppress recommendations](#).

Run a repository analysis [Info](#)

When you associate a repository, CodeGuru Reviewer performs a full code review on the repository. After the repository association is created, code in all pull requests are reviewed and you can create new repository code reviews.

Source branch

Select a source branch for CodeGuru Reviewer to scan.

 *Name of source branch*

Code review name - *optional*

Enter your code review name.

Code review name

▼ Analysis configuration file - *optional*

Suppress recommendations from CodeGuru Reviewer.

Create an `aws-codeguru-reviewer.yml` file that uses glob expressions to exclude files and directories in your repository for code reviews. Then add this file to the root directory of your repository. [Learn more](#) 

 [Download sample aws-codeguru-reviewer.yml file](#)

9. (Optional) Expand **Tags** to add one or more tags to your repository association. For more information, see [Tagging a repository association in Amazon CodeGuru Reviewer](#).
 - a. Choose **Add new tag**.
 - b. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
 - c. (Optional) To add another tag, choose **Add new tag**.
10. Choose **Associate repository and run analysis**. On the **Repositories** page, the **Status** is **Associating**. When the association is complete, the status changes to **Associated** and a full repository analysis begins. Refresh the page to check for the status change.

Create a GitHub Enterprise Server repository association in Amazon CodeGuru Reviewer

You can create a GitHub Enterprise Server repository association using the Amazon CodeGuru Reviewer console, the AWS CLI, or the CodeGuru Reviewer SDK. Before you create a GitHub Enterprise Server repository association, you must have a GitHub Enterprise Server repository.

Note

GitHub Enterprise Cloud repositories have a different procedure and different prerequisites. If you're using GitHub Enterprise Cloud, [follow this procedure instead](#).

Topics

- [GitHub Enterprise Server repository association prerequisites](#)
- [Create a GitHub Enterprise Server repository association \(console\)](#)
- [Create a GitHub Enterprise Server repository association \(AWS CLI\)](#)
- [Create a GitHub Enterprise Server repository association \(AWS SDKs\)](#)

GitHub Enterprise Server repository association prerequisites

To create a GitHub Enterprise Server repository association, you must have a GitHub Enterprise Server connection in AWS CodeConnections. The connection must be in the same AWS account and Region in which you want your code reviews. For more information, see [Create a connection](#) and [Create a connection to GitHub Enterprise Server](#) in the *Developer Tools User Guide*.

Important

CodeConnections does not support GitHub Enterprise Server version 2.22.0 due to a known issue in the release. To create a connection, use version 2.22.1 or later.

Your GitHub Enterprise Server connection requires a *host*. The host represents your GitHub Enterprise Server instance and is to what your GitHub Enterprise Server connection connects. A host can be an on-premises server or a Virtual Private Cloud (VPC). For more information, see [Amazon VPC configuration for your host](#) and [Create a host](#) in the *AWS Developer Tools User Guide*.

Create a GitHub Enterprise Server repository association (console)

To create a GitHub Enterprise Server repository association

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Choose **Associate repository and run analysis**.
4. Choose **GitHub Enterprise Server**.
5. From **Connect to GitHub Enterprise Server (with AWS CodeConnections)**, choose the connection you want to use. If you don't have a connection, choose **Create a GitHub Enterprise Server connection** to create one in the Developer Tools console. For more information, see [Create a connection](#) in the *AWS Developer Tools User Guide*.
6. From **Repository location**, choose the name of your GitHub Enterprise Server repository that contains the source code you want CodeGuru Reviewer to analyze.
7. (Optional) Expand **Encryption key - optional** to use your own AWS Key Management Service key (KMS key) to encrypt your associated repository. For more information, see [Encrypting a repository association in Amazon CodeGuru Reviewer](#).
 - a. Select **Customize encryption settings (advanced)**.
 - b. Do one of the following:
 - If you already have a KMS key that you manage, enter its Amazon Resource Name (ARN). For information about finding the ARN of your key using the console, see [Finding the key ID and key ARN](#) in the *AWS Key Management Service Developer Guide*.
 - If you want to create a KMS key, choose **Create an AWS KMS key** and follow the steps in the AWS KMS console. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.
8. In **Run a repository analysis**, specify information for your associated repository's first full scan. This scan generates your repository's initial code review. For more information, see [Get recommendations using full repository analysis](#).
 - a. From **Source branch**, choose the branch to use.
 - b. (Optional) In **Code review name**, type a name for your code review.
 - c. (Optional) Expand **Analysis configuration file - optional** to download a sample `aws-codeguru-reviewer.yml` file to use as a template. Modify the file and upload it to the

root directory of your repository. For more information about the analysis configuration file, see [Suppress recommendations](#).

Run a repository analysis [Info](#)

When you associate a repository, CodeGuru Reviewer performs a full code review on the repository. After the repository association is created, code in all pull requests are reviewed and you can create new repository code reviews.

Source branch

Select a source branch for CodeGuru Reviewer to scan.

Code review name - optional

Enter your code review name.

▼ Analysis configuration file - optional

Suppress recommendations from CodeGuru Reviewer.

Create an `aws-codeguru-reviewer.yml` file that uses glob expressions to exclude files and directories in your repository for code reviews. Then add this file to the root directory of your repository. [Learn more](#)

 [Download sample aws-codeguru-reviewer.yml file](#)

9. (Optional) Expand **Tags** to add one or more tags to your repository association. For more information, see [Tagging a repository association in Amazon CodeGuru Reviewer](#).
 - a. Choose **Add new tag**.
 - b. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
 - c. (Optional) To add another tag, choose **Add new tag**.
10. Choose **Associate repository and run analysis**. On the **Repositories** page, the **Status** is **Associating**. When the association is complete, the status changes to **Associated** and a full repository analysis begins. Refresh the page to check for the status change.

Create a GitHub Enterprise Server repository association (AWS CLI)

For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section of the AWS CLI Command Reference](#)

To create a GitHub Enterprise Server repository association

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

2. Run the **associate-repository** command specifying the owner (or user name) of your GitHub Enterprise Server account, the name of your repository, and the Amazon Resource Name (ARN) of your connection.

```
aws codeguru-reviewer associate-repository --repository
GitHubEnterpriseServer="{Owner=github-enterprise-server-user-name,
Name=repository-name, \
ConnectionArn=arn:aws:codeconnections:us-west-2:123456789012:connection/
connection-uuid }"
```

3. If successful, this command outputs a [RepositoryAssociation](#) object.

```
{
  "RepositoryAssociation": {
    "ProviderType": "GitHubEnterpriseServer",
    "Name": "repository-name",
    "LastUpdatedTimeStamp": 1595966211.79,
    "AssociationId": "repository-association-uuid",
    "CreatedTimeStamp": 1595966211.79,
    "ConnectionArn": "arn:aws:codeconnections:us-
west-2:123456789012:connection/connection-uuid",
    "State": "Associating",
    "StateReason": "Pending Repository Association",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
east-2:123456789012:association:repository-association-uuid",
    "Owner": "github-enterprise-server-user-name"
  }
}
```

4. When the **associate-repository** command succeeds, the status in the returned output is **Associating**. When the association is complete, the status changes to **Associated** and you can create a pull request or a full repository analysis to get recommendations. You can check your repository association's status using the `describe-repository` command with its Amazon Resource Name (ARN).

```
aws codeguru-reviewer describe-repository-association --association-arn
arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-
association-uuid
```

5. If successful, this command outputs a [RepositoryAssociation](#) object which shows its status.

```
{
  "RepositoryAssociation": {
    "ProviderType": "GitHubEnterpriseServer",
    "Name": "repository-name",
    "LastUpdatedTimeStamp": 1595634764.029,
    "AssociationId": "repository-association-uuid",
    "CreatedTimeStamp": 1595634764.029,
    "ConnectionArn": "arn:aws:codeconnections:us-
west-2:123456789012:connection/connection_uuid"
    "State": "Associated",
    "StateReason": "Pull Request Notification configuration successful",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid",
    "Owner": "github-enterprise-server-user-name"
  }
}
```

Create a GitHub Enterprise Server repository association (AWS SDKs)

To create a GitHub Enterprise Server repository association with the AWS SDKs, use the `AssociateRepository` API. For more information, see [AssociateRepository](#) in the *Amazon CodeGuru Reviewer API Reference*.

View all repository associations in CodeGuru Reviewer

You can view all the associated repositories in your AWS Region and your AWS account using the console or the AWS CLI.

Topics

- [View all associated repositories in CodeGuru Reviewer \(console\)](#)
- [View all repository associations in CodeGuru Reviewer \(AWS CLI\)](#)

View all associated repositories in CodeGuru Reviewer (console)

View all associated repositories

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Associated repositories**.

View all repository associations in CodeGuru Reviewer (AWS CLI)

For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section](#) and [list-repository-associations](#) in the *AWS CLI Command Reference*.

View all repository associations

1. Make sure that you have configured the AWS CLI with the AWS Region that contains the repository associations you want to view. Run the following command at the command line or terminal and review or configure the Region for the AWS CLI.

```
aws configure
```

2. Run **list-repository-associations**.

```
aws codeguru-reviewer list-repository-associations
```

3. If successful, this command outputs one [RepositoryAssociationSummary](#) object for each of your associated repositories.

```
{
```

```

"RepositoryAssociationSummaries": [
  {
    "LastUpdatedTimeStamp": 1595886609.616,
    "Name": "test",
    "AssociationId": "0bdac454-f6af-4adf-a625-de4db4b4bca1",
    "Owner": "123456789012",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:0bdac454-f6af-4adf-a625-de4db4b4bca1",
    "ProviderType": "Bitbucket"
  },
  {
    "LastUpdatedTimeStamp": 1595636969.035,
    "Name": "CodeDeploy-CodePipeline-ECS-Tutorial",
    "AssociationId": "eb2f7513-a132-47ad-81dc-bd718468ee1e",
    "Owner": "123456789012",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:eb2f7513-a132-47ad-81dc-bd718468ee1e",
    "ProviderType": "CodeCommit"
  },
  {
    "LastUpdatedTimeStamp": 1595634785.983,
    "Name": "My-ecs-beta-repo",
    "AssociationId": "d79156d7-6297-4b08-ba5a-f05b274e3518",
    "Owner": "123456789012",
    "State": "Associated",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:d79156d7-6297-4b08-ba5a-f05b274e3518",
    "ProviderType": "CodeCommit"
  }
]
}

```

Disassociate a repository in CodeGuru Reviewer

You can disassociate any associated repository you create. After you disassociate a repository, Amazon CodeGuru Reviewer no longer has permission to read code in the repository's pull requests and doesn't have access to your repository's source code. This means that CodeGuru Reviewer does not have permission to perform operations such as cloning or publishing comments in source code.

A disassociated repository does not send pull request notifications to Amazon CodeGuru Reviewer. You cannot create code reviews of any kind for a disassociated repository.

Immediately after you choose to disassociate a repository, its status changes to **Disassociating**. If you want to review code in your disassociated repository later, you can create a new repository association.

Important

If you are disassociating a Bitbucket or GitHub Enterprise Server repository, you must disassociate the repository before deleting your AWS CodeStar connection.

Note

Charges are not incurred for disassociated repositories.

Topics

- [Disassociate a repository in CodeGuru Reviewer \(console\)](#)
- [Disassociate a repository in CodeGuru Reviewer \(AWS CLI\)](#)

Disassociate a repository in CodeGuru Reviewer (console)

To disassociate a repository

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Do one of the following:
 - Choose the radio button next to the repository you want to disassociate, then choose **Disassociate repository**.
 - Choose the association ID of the repository you want to disassociate. On its **Repository** page, choose **Disassociate repository**. With this option, you can view details about your repository before you disassociate it.

Disassociate a repository in CodeGuru Reviewer (AWS CLI)

For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section of the AWS CLI Command Reference](#)

Disassociate a repository association

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **disassociate-repository** command specifying the Amazon Resource Name (ARN) of your associated repository.

```
aws codeguru-reviewer disassociate-repository --association-arn arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid
```

3. If successful, this command outputs a [RepositoryAssociation](#) object with a state of Disassociating.

```
{
  "RepositoryAssociation": {
    "AssociationId": "repository_association_uuid",
    "Name": "repository-name",
    "LastUpdatedTimeStamp": 1602119553.692,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1590712779.949,
    "Owner": "123456789012",
    "State": "Disassociating",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid"
  }
}
```

4. When the **disassociate-repository** command completes, the repository is not associated with Amazon CodeGuru Reviewer. You can check if your repository association successfully

disassociated using the `describe-repository` command with its Amazon Resource Name (ARN).

```
aws codeguru-reviewer describe-repository-association --association-arn
arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-
association-uuid
```

5. If successful, the repository association is deleted and the command correctly outputs the following:

```
An error occurred (NotFoundException) when calling the
DescribeRepositoryAssociation operation: The requested resource arn:aws:codeguru-
reviewer:us-west-2:123456789012:association:repository-association-uuid is not
found.
```

Encrypting a repository association in Amazon CodeGuru Reviewer

All associated repositories in Amazon CodeGuru Reviewer are encrypted by default using a key that AWS owns and manages for you. You can encrypt an associated repository using an AWS Key Management Service key, known as a *KMS key*, that you manage. If you want to use a KMS key, then you must create one in advance using AWS KMS, or create one when you create your associated repository. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.

You can encrypt an associated repository with a KMS key only when you create it. If you want to update how an existing repository is encrypted, you must disassociate it and then recreate it with the encryption you want. For more information, see [Disassociate a repository in CodeGuru Reviewer](#).

The encryption key (either an AWS owned and managed key, or a KMS key you create) encrypts the associated repository and all of its code reviews. Each code review is a child of the associated repository that contains the reviewed code.

If you encrypt an associated repository with a KMS key, then revoke access to that key by disabling it or removing CodeGuru Reviewer access to AWS KMS using the AWS Identity and Access Management AWS CLI or SDK, the following occurs:

- Recommendations related to the associated repository become unavailable.
- You cannot successfully review code in the associated repository. You can schedule a code review, but the code review fails.

To restore access to an associated repository that is encrypted with a disabled key, you can re-enable it. For more information, see [Enabling and disabling keys](#) in the *AWS Key Management Service Developer Guide*.

 Note

Creation of an AWS KMS key results in charges to your AWS account. For more information, see [AWS Key Management Service pricing](#).

Topics

- [Encrypt an associated repository using an AWS KMS key](#)
- [Update how a repository association is encrypted](#)

Encrypt an associated repository using an AWS KMS key

You can use the Amazon CodeGuru Reviewer console to specify an AWS Key Management Service key (KMS key) to encrypt your associated repository. If you don't do this, your associated repository is encrypted by default using a key that is owned and managed by AWS.

Encrypt an associated repository using a KMS key

1. Follow the steps in one of the following topics to create an association with your repository type:
 - [Create an AWS CodeCommit repository association \(console\)](#)
 - [Create a Bitbucket repository association \(console\)](#)
 - [Create a GitHub or GitHub Enterprise Cloud repository association \(console\)](#)
 - [Create a GitHub Enterprise Server repository association \(console\)](#)
2. Expand **Additional configuration**.
3. Select **Customize encryption settings (advanced)**.
4. Do one of the following:

- If you already have a KMS key that you manage, enter its Amazon Resource Name (ARN). For information about finding the ARN of your key using the console, see [Finding the key ID and key ARN](#) in the *AWS Key Management Service Developer Guide*.
 - If you want to create a KMS key, choose **Create an AWS KMS key** and follow the steps in the AWS KMS console. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.
5. Complete the rest of the steps to create your repository association.

Update how a repository association is encrypted

If you want to update how your associated repository is encrypted, you must disassociate it, then recreate it. When you recreate the associated repository, specify the AWS Key Management Service key (KMS key) you want to use. If you don't specify a KMS key, then your data is encrypted by a key that is managed by AWS.

Change how an associated repository is encrypted

1. Disassociate your associated repository by following the steps in [Disassociate a repository in CodeGuru Reviewer \(console\)](#).
2. Follow the steps in one of the following topics to create an association with your repository type. Specify the KMS key you want to use or don't specify any KMS key if you want to encrypt your data using an AWS owned and managed key.
 - [Create an AWS CodeCommit repository association \(console\)](#)
 - [Create a Bitbucket repository association \(console\)](#)
 - [Create a GitHub or GitHub Enterprise Cloud repository association \(console\)](#)
 - [Create a GitHub Enterprise Server repository association \(console\)](#)

Tagging a repository association in Amazon CodeGuru Reviewer

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each AWS tag has two parts:

- A tag *key* (for example, CostCenter, Environment, Project, or Secret). Tag keys are case sensitive.

- An optional field known as a tag *value* (for example, 111122223333, Production, or a team name). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Together these are known as *key-value* pairs. For limits on the number of tags you can have on an associated repository and restrictions on tag keys and values, see [Tags](#).

Tags help you identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to a CodeGuru Reviewer associated repository that you assign to an AWS CodeBuild build project. For more information about using tags, see the [Tagging Best Practices](#) whitepaper.

In CodeGuru Reviewer, you can use the CodeGuru Reviewer console, the AWS CLI, CodeGuru Reviewer APIs, or AWS SDKs to add, manage, and remove tags for a repository association. In addition to identifying, organizing, and tracking your repository association with tags, you can use tags in IAM policies to help control who can view and interact with your repository association.

A repository association has a parent-child hierarchical relationship with code reviews because a repository association contains all the code reviews inside it. Because of this, you can use tags on repository associations to control access to the code reviews in it. For examples of tag-based access policies, see [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#).

Topics

- [Add a tag to a CodeGuru Reviewer associated repository](#)
- [View tags for a CodeGuru Reviewer associated repository](#)
- [Add or update tags for a CodeGuru Reviewer associated repository](#)
- [Remove tags from a CodeGuru Reviewer associated repository](#)

Add a tag to a CodeGuru Reviewer associated repository

Adding tags to an associated repository can help you identify and organize your AWS resources and manage access to them. First, you add one or more tags (key-value pairs) to an associated repository. Keep in mind that there are limits on the number of tags you can have on an associated repository. There are restrictions on the characters you can use in the key and value fields. For more information, see [Tags](#). After you have tags, you can create IAM policies to manage access to the

associated repository based on these tags. You can use the CodeGuru Reviewer console, AWS CLI, or SDK to add tags to an associated repository.

Important

Adding tags to an associated repository can impact access to that associated repository. Before you add a tag to an associated repository, make sure to review any IAM policies that might use tags to control access to resources such as associated repositories. For examples of tag-based access policies, see [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#).

Topics

- [Add a tag to a CodeGuru Reviewer associated repository \(console\)](#)
- [Add a tag to a CodeGuru Reviewer associated repository \(AWS CLI\)](#)

Add a tag to a CodeGuru Reviewer associated repository (console)

You can use the console to add a tag when you create an associated repository or to one that already exists.

Topics

- [Add a tag when you create a CodeGuru Reviewer associated repository \(console\)](#)
- [Add a tag to an existing CodeGuru Reviewer associated repository \(console\)](#)

Add a tag when you create a CodeGuru Reviewer associated repository (console)

You can use the Amazon CodeGuru Reviewer console to add one or more tags when you create an Amazon CodeGuru Reviewer associated repository.

Add a tag when you create an associated repository

1. Follow the steps in one of the following topics to create an association with your repository type:
 - [Create an AWS CodeCommit repository association \(console\)](#)
 - [Create a Bitbucket repository association \(console\)](#)

- [Create a GitHub or GitHub Enterprise Cloud repository association \(console\)](#)
 - [Create a GitHub Enterprise Server repository association \(console\)](#)
2. Expand **Tags**.
 3. Choose **Add new tag**.
 4. In **Key**, enter a name for the tag. You can add an optional value for the tag in **Value**.
 5. (Optional) To add another tag, choose **Add new tag**.
 6. Complete the rest of the steps to create your repository association.

Add a tag to an existing CodeGuru Reviewer associated repository (console)

You can use the Amazon CodeGuru Reviewer console to add one or more tags to an existing CodeGuru Reviewer associated repository.

Add a tag to an existing associated repository

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Do one of the following:
 - Choose the association ID of the associated repository where you want to view tags, then choose **Manage tags**.
 - Choose the radio button next to the associated repository where you want to view tags, then choose **Manage tags**.
4. In **Manage tags**, for each tag you want to add:
 - a. Choose **Add new tag**.
 - b. In **key**, enter a name for the tag.
 - c. (Optional) In **value**, enter a value for the tag.
5. When you have finished adding tags, choose **Save changes**.

Add a tag to a CodeGuru Reviewer associated repository (AWS CLI)

You can use the AWS CLI to add a tag to an associated repository that already exists or when you create it. For information about using the AWS CLI with CodeGuru Reviewer, see the [CodeGuru Reviewer section of the AWS CLI Command Reference](#).

Topics

- [Add a tag when you create a CodeGuru Reviewer associated repository \(AWS CLI\)](#)
- [Add a tag to an existing CodeGuru Reviewer associated repository \(AWS CLI\)](#)

Add a tag when you create a CodeGuru Reviewer associated repository (AWS CLI)

You can use the AWS CLI to add tags to an associated repository when you create it.

Note

Because you cannot use the AWS CLI to create a GitHub repository, you cannot use the AWS CLI to add tags to a GitHub repository when you create it. You can use the AWS CLI to add tags to an existing GitHub repository using **tag-resource**. You can also add tags when you create a GitHub repository association with the console.

To add a tag when you create a repository association

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **associate-repository** command specifying the tags you want to add with the `--tags` parameter. Specify a tag's key and value using an equal symbol (for example, `my-key=my-value`). For more information about how to use **associate-repository** to create an association with your repository type, see one of the following:

- [Create a CodeCommit repository association \(AWS CLI\)](#)

- [Create a Bitbucket repository association \(AWS CLI\)](#)
- [Create a GitHub Enterprise Server repository association \(AWS CLI\)](#)

The following example adds 3 tags when you create an AWS CodeCommit repository association.

```
aws codeguru-reviewer associate-repository --repository CodeCommit={Name=my-codecommit-repo} /  
--tags value-1=key-1,owner=admin,status=beta
```

3. If successful, this command outputs a [RepositoryAssociation](#) object that includes an array with the 3 tags.

```
{  
  "RepositoryAssociation": {  
    "AssociationId": "repository-association-uuid",  
    "Name": "my-codecommit-repo",  
    "LastUpdatedTimeStamp": 1595634764.029,  
    "ProviderType": "CodeCommit",  
    "CreatedTimeStamp": 1595634764.029,  
    "Owner": "123456789012",  
    "State": "Associating",  
    "StateReason": "Pending Repository Association",  
    "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid",  
  },  
  "Tags": {  
    "owner": "admin",  
    "status": "beta",  
    "value-1": "key-1",  
  }  
}
```

Add a tag to an existing CodeGuru Reviewer associated repository (AWS CLI)

You use the same command, **tag-resource**, to update and to use the AWS CLI to add tags to an associated repository.

To add a tag to an existing repository association

- Follow the steps in [Add or update tags for a CodeGuru Reviewer associated repository \(AWS CLI\)](#).

View tags for a CodeGuru Reviewer associated repository

Tags can help you identify and organize your AWS resources and manage access to them. For more information about tagging strategies, see [Tagging AWS resources](#) in the *Amazon Web Services General Reference*. For examples of tag-based access policies, see [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#).

Topics

- [View tags for an associated repository \(console\)](#)
- [View tags for a CodeGuru Reviewer associated repository \(AWS CLI\)](#)

View tags for an associated repository (console)

You can use the CodeGuru Reviewer console to view the tags associated with a CodeGuru Reviewer associated repository.

To view tags for an associated repository

- Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
- In the navigation pane, choose **Repositories**.
- Do one of the following:
 - Choose the association ID of the associated repository where you want to view tags, then look under **Tags**.
 - Choose the radio button next to the associated repository where you want to view tags, then choose **Manage tags**.

View tags for a CodeGuru Reviewer associated repository (AWS CLI)

Follow these steps to use the AWS CLI to view the AWS tags for an associated repository. If no tags have been added, the returned tags list in the response is empty ("Tags": {}).

To view tags for an associated repository

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **describe-repository-association** command and specify the Amazon Resource Name (ARN) of the associated repository.

```
aws codeguru-reviewer describe-repository-association /  
  --association-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:repository-association-uuid
```

3. If successful, this command outputs a [RepositoryAssociation](#) object that includes an array with its tags.

```
{  
  "RepositoryAssociation": {  
    "AssociationId": "repository-association-uuid",  
    "Name": "my-codecommit-repo",  
    "LastUpdatedTimeStamp": 1595634764.029,  
    "ProviderType": "CodeCommit",  
    "CreatedTimeStamp": 1595634764.029,  
    "Owner": "123456789012",  
    "State": "Associating",  
    "StateReason": "Pending Repository Association",  
    "AssociationArn": "arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:repository-association-uuid",  
  },  
  "Tags": {  
    "owner": "admin",  
    "status": "beta",  
    "value-1": "key-1",  
  }  
}
```

Add or update tags for a CodeGuru Reviewer associated repository

You can change the value for a tag associated with an associated repository or add a new tag. Keep in mind that there are limits on the characters you can use in the key and value fields. For more information, see [Limits](#).

Important

Updating the value of a tag for an associated repository can impact access to that associated repository. Before you update the value of a tag for an associated repository, make sure to review any IAM policies that might use the value to control access to resources such as associated repositories. For examples of tag-based access policies, see [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#).

Topics

- [Add or update tags for a CodeGuru Reviewer associated repository \(console\)](#)
- [Add or update tags for a CodeGuru Reviewer associated repository \(AWS CLI\)](#)

Add or update tags for a CodeGuru Reviewer associated repository (console)

You can use the CodeGuru Reviewer console to update, add, or remove the tags associated with a CodeGuru Reviewer associated repository. Using the console, you can also change the name of the key, which is equivalent to removing the current tag and adding a different one with the new name and the same value.

To add or update tags for an associated repository

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Do one of the following:
 - Choose the association ID of the associated repository where you want to view tags, then choose **Manage tags**.
 - Choose the radio button next to the associated repository where you want to view tags, then choose **Manage tags**.

4. Enter new values in **key** and **value** to edit tags. Choose **Remove** next to a tag to remove it. Choose **Add new tag** to add a new tag.
5. Choose **Save changes** when you are finished.

Add or update tags for a CodeGuru Reviewer associated repository (AWS CLI)

Follow these steps to use the AWS CLI and the **tag-resource** command to add or update the AWS tags for an associated repository. This command adds a new tag or, if you pass in a tag with an existing key, updates the value associated with that key. If you want to use the AWS CLI to update the key of a tag, use **untag-resource** to remove it, then use **tag-resource** to add a new tag with the updated key and its value.

To add or update tags for an associated repository

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **tag-resource** command. Use **--resource-arn** to specify the Amazon Resource Name (ARN) of the associated repository that contains the tags you want to update or add. Use the **--tags** argument to specify the tags you want to update or add. The following command specifies 3 tags. If one of the keys already exists, its value is updated. If not, a new key is added.

```
aws codeguru-reviewer tag-resource /  
  --resource-arn arn:aws:codeguru-reviewer:us-  
west-2:123456789012:association:repository-association-uuid /  
  --tags key1=value1,key2=value2,key3=value3
```

3. If successful, there is no output and no error. If you want to verify the tags were added correctly, use the **describe-repository-association** command and use **--association-arn** to specify the ARN of the associated repository.

```
aws codeguru-reviewer describe-repository-association /
```

```
--association-arn arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid
```

The output is a [RepositoryAssociation](#) object that includes an array with the 3 added or updated tags.

```
{
  "RepositoryAssociation": {
    "AssociationId": "repository-association-uuid",
    "Name": "my-repository-name",
    "LastUpdatedTimeStamp": 1603493340.035,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1603493328.512,
    "Owner": "123456789012",
    "State": "Associated",
    "StateReason": "\"Pull Request Notification configuration successful\"",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:repository-association-uuid"
  },
  "Tags": {
    "key3": "value3",
    "key2": "value2",
    "key1": "value1"
  }
}
```

Remove tags from a CodeGuru Reviewer associated repository

You can use the console or the AWS CLI to remove tags from an associated repository.

Topics

- [Remove tags from a CodeGuru Reviewer associated repository \(console\)](#)
- [Remove tags from a CodeGuru Reviewer associated repository \(AWS CLI\)](#)

Remove tags from a CodeGuru Reviewer associated repository (console)

To remove tags from an associated repository

1. Open the Amazon CodeGuru Reviewer console at <https://console.aws.amazon.com/codeguru/reviewer/>.
2. In the navigation pane, choose **Repositories**.
3. Choose the association ID of the associated repository with the tags you want to edit.
4. In **Tags**, choose **Manage tags**.
5. Choose **Remove** next to each tag you want to remove.
6. Choose **Save changes**.

Remove tags from a CodeGuru Reviewer associated repository (AWS CLI)

You can remove a tag from an associated repository using the console or the AWS CLI.

Important

Removing a tag from an associated repository can impact access to that associated repository. Before you remove a tag from an associated repository, make sure to review any IAM policies that might use its key or value to control access to resources such as associated repositories. For examples of tag-based access policies, see [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#).

To remove tags from an associated repository

1. Make sure that you have configured the AWS CLI with the AWS Region in which you want to create your code reviews. To verify the Region, run the following command at the command line or terminal and review the information for the default name.

```
aws configure
```

The default Region name must match the AWS Region for the repository in CodeCommit.

2. Run the **untag-resource** command. Use `--resource-arn` to specify the Amazon Resource Name (ARN) of the associated repository that contains the tags you want to update or add

to. Use the `--tag-keys` argument to specify the *key* of the tags you want to remove. The following command removes 3 tags.

```
aws codeguru-reviewer untag-resource /
  --resource-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid /
  --tag-keys key1 key2 key3
```

3. If successful, there is no output and no error. If you want to verify the tags were removed correctly, use the **describe-repository-association** command and use `--association-arn` to specify the ARN of the associated repository.

```
aws codeguru-reviewer describe-repository-association /
  --association-arn arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid
```

The output is a [RepositoryAssociation](#) object that includes an array that does not contain the keys you removed. In the following output example, all tags were removed so the tags array is empty.

```
{
  "RepositoryAssociation": {
    "AssociationId": "repository-association-uuid",
    "Name": "my-repository-name",
    "LastUpdatedTimeStamp": 1603493340.035,
    "ProviderType": "CodeCommit",
    "CreatedTimeStamp": 1603493328.512,
    "Owner": "123456789012",
    "State": "Associated",
    "StateReason": "\"Pull Request Notification configuration successful\"",
    "AssociationArn": "arn:aws:codeguru-reviewer:us-
west-2:123456789012:association:repository-association-uuid"
  },
  "Tags": {}
}
```

Working with code reviews

You can create an incremental code review or a full repository analysis code review. For more information about the types of code reviews, see [About full repository analysis and incremental code reviews](#).

When you create a code review, CodeGuru Reviewer reviews your code and provides recommendations about how to improve the code. You can see these recommendations directly in the CodeGuru Reviewer console **Code reviews** page. For incremental code reviews, you can also see these recommendations as comments on a pull request.

If a repository contains Java and Python files, then CodeGuru Reviewer generates recommendations for the language for which there are more files. For example, if there are five Java files and ten Python files in an associated repository, then CodeGuru Reviewer generates recommendations for the Python code and does not generate recommendations for the Java code. If the number of Java and Python files is the same, then only Java recommendations are generated.

If you want to suppress recommendations from CodeGuru Reviewer, you can create and add to the root directory of your repository an `aws-codeguru-reviewer.yml` file that lists files and directories to exclude from analysis. For more information, see [Suppress recommendations](#).

You can do the following on the **Code reviews** page in the CodeGuru console:

- Create full repository analysis code reviews.
- View all code reviews from the past 90 days.
- Navigate directly to incremental code reviews and repositories on which the code reviews were performed.
- View details about code reviews and whether they are completed.
- Check whether CodeGuru Reviewer used your `aws-codeguru-reviewer.yml` file for an analysis and review any error messages about the file.
- View all the recommendations of each code review directly in the console.
- Provide feedback to indicate which recommendations were helpful and which recommendations were not helpful.

Your feedback on recommendations is crucial to helping CodeGuru Reviewer improve its recommendations. You can use the **Code Reviews** page to give feedback on recommendations by choosing the name of the repository and then choosing thumbs-up or thumbs-down icons.

Topics

- [About full repository analysis and incremental code reviews](#)
- [Suppress recommendations from Amazon CodeGuru Reviewer](#)
- [Create code reviews in Amazon CodeGuru Reviewer](#)
- [View all code reviews](#)
- [View code review details](#)
- [View recommendations and provide feedback](#)

About full repository analysis and incremental code reviews

There are three different kinds of code reviews that CodeGuru Reviewer can do to provide recommendations.

- *Incremental code reviews* are created automatically when you create a pull request from your repository context on an associated repository. These code reviews scan the changed code in a pull request.
- *Full repository analysis code reviews* are done when you create a full repository analysis code review in the CodeGuru Reviewer console. These code reviews scan all the code in a specified branch.

Note

Your first full repository analysis is created for you when you associate your repository.

- *Full repository analysis code reviews for CI/CD workflows* scan all the source code in your CI/CD workflow. For more information, see [Create code reviews with GitHub Actions](#).

You can receive recommendations in code reviews by creating a full repository analysis or submitting a pull request. After you associate a repository, CodeGuru Reviewer automatically creates a full repository analysis code review, and every pull request in that repository creates an incremental code review. You can also choose to create additional full repository analysis code reviews.

Type of code review	Is the review automatic after I associate the repository?	Where can I see recommendations?	What code is reviewed?
Full repository analysis	Your first full repository analysis is done automatically when you associate your repository. After that, you must request a full repository analysis in the CodeGuru Reviewer console or by using the AWS CLI or AWS SDK.	In the CodeGuru Reviewer console, or by using the AWS CLI or AWS SDK.	All the code in the branch is reviewed.
Incremental code review	Yes. After associating the repository, every time you do a pull request there is a code review.	In the CodeGuru Reviewer console, in the AWS CLI or AWS SDK, or in pull request comments in the repository source provider.	The code that is changed in the pull request is reviewed.
GitHub Actions code review in a CI/CD workflow	Yes. After enabling CodeGuru Reviewer on your GitHub repository, for every push, pull, or scheduled repository scan there is a code review.	In the GitHub Security tab.	The code that is changed in the push, pull, or scheduled repository scan.

Suppress recommendations from Amazon CodeGuru Reviewer

Whether you initiate a full repository analysis code review or an incremental code review, you can suppress recommendations from CodeGuru Reviewer. You do this by excluding files and directories in an `aws-codeguru-reviewer.yml` file.

By excluding files or directories, your costs associated with CodeGuru Reviewer analyzing your repository might also decrease. For more information, see [Cost impact of suppressing recommendations](#).

The following examples describe scenarios in which you might want to use an `aws-codeguru-reviewer.yml` file to exclude files or directories.

- Your repository contains directories that should not be included in a code review, such as `test`, `generated`, or `module` directories.
- Your repository is a large open-source repository and you don't want files about a feature or product inside the repository to be analyzed.

Topics

- [Structure of the `aws-codeguru-reviewer.yml` file](#)
- [Steps to suppress recommendations](#)
- [Cost impact of suppressing recommendations](#)
- [Error handling for the `aws-codeguru-reviewer.yml` file](#)

Structure of the `aws-codeguru-reviewer.yml` file

The following content lists the criteria that your `aws-codeguru-reviewer.yml` file must meet, includes sample code that you can use as a template to create your file, and illustrates with example code snippets how to exclude files and directories.

Note

Be sure to use relative paths for files and directories that you add to your `aws-codeguru-reviewer.yml` file.

Criteria for the `aws-codeguru-reviewer.yml` file

In addition to being valid YAML and not containing syntax errors, your `aws-codeguru-reviewer.yml` file must meet the following criteria. If your file does not, then CodeGuru Reviewer returns error messages and does not use your file in any analysis. For more information, see [Error handling for the `aws-codeguru-reviewer.yml` file](#).

- File name: `aws-codeguru-reviewer.yml`

Important

You must name the file `aws-codeguru-reviewer` and use the extension `.yml`, not `.yaml`. If you don't, then CodeGuru Reviewer cannot recognize your file, use it in analyses, or return error messages about your file.

- Maximum file size: 100 KB
- Maximum length of each glob expression: 100 characters
- Maximum number of glob expressions: 100
- Version number: 1.0

For more information about glob expressions, see [glob \(programming\)](#) in *Wikipedia*.

Sample `aws-codeguru-reviewer.yml` file

You can copy the following sample YAML file content to use as a template when creating your own `aws-codeguru-reviewer.yml` file. You can also download this sample file from the CodeGuru Reviewer console or from the [sample application repository in GitHub](#).

Replace the items under `excludeFiles:` with your own files and directories. For information about how to exclude files and directories, see [Example code for the `aws-codeguru-reviewer.yml` file](#).

Note

Be sure to use relative paths for files and directories that you add to your `aws-codeguru-reviewer.yml` file.

```
version: 1.0

# Sample YAML file
# This configuration file is an optional file for Amazon CodeGuru Reviewer.
# You must name your file aws-codeguru-reviewer.yml for CodeGuru Reviewer to recognize it.
# Add the file to the root directory of your repository.
# For more information, see the Amazon CodeGuru Reviewer User Guide:
# https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/welcome.html.

# Exclude files and directories
# Use glob pattern syntax to specify the files and directories that you want to exclude
# from analysis under excludeFiles. For more information, see glob (programming):
# https://en.wikipedia.org/wiki/Glob_(programming).
# The following example excludes from analysis all content in the some-package directory, all content
# in the tst directory, and all JSON files in a sub-directory under a directory prefixed with some-.
# Replace with your own list of files and directories that you want CodeGuru Reviewer to exclude.
# We recommend that you use a schema validator to confirm that the YAML syntax is valid.

excludeFiles:
  - 'src/some-package/**'
  - 'tst/**'
  - 'src/some-*/**/*.json'
```

Example code for the aws-codeguru-reviewer.yml file

In your `aws-codeguru-reviewer.yml` file, add files and directories to exclude under `excludeFiles`. Use glob pattern syntax. For more information about glob expressions, see [glob \(programming\)](https://en.wikipedia.org/wiki/Glob_(programming)) in *Wikipedia*.

Example 1: Excluding a particular directory

In the following example, CodeGuru Reviewer excludes from analysis anything in the `resources` directory.

```
version: 1.0
```

```
excludeFiles:
  - 'resources/*'
```

Example 2: Excluding files under any directory with a particular name

In the following example, CodeGuru Reviewer excludes from analysis all files under any directory named configuration.

```
version: 1.0

excludeFiles:
  - '**/configuration/*'
```

Example 3: Excluding all Java files

In the following example, CodeGuru Reviewer excludes from analysis any file with a .java extension.

```
version: 1.0

excludeFiles:
  - '**/*.java'
```

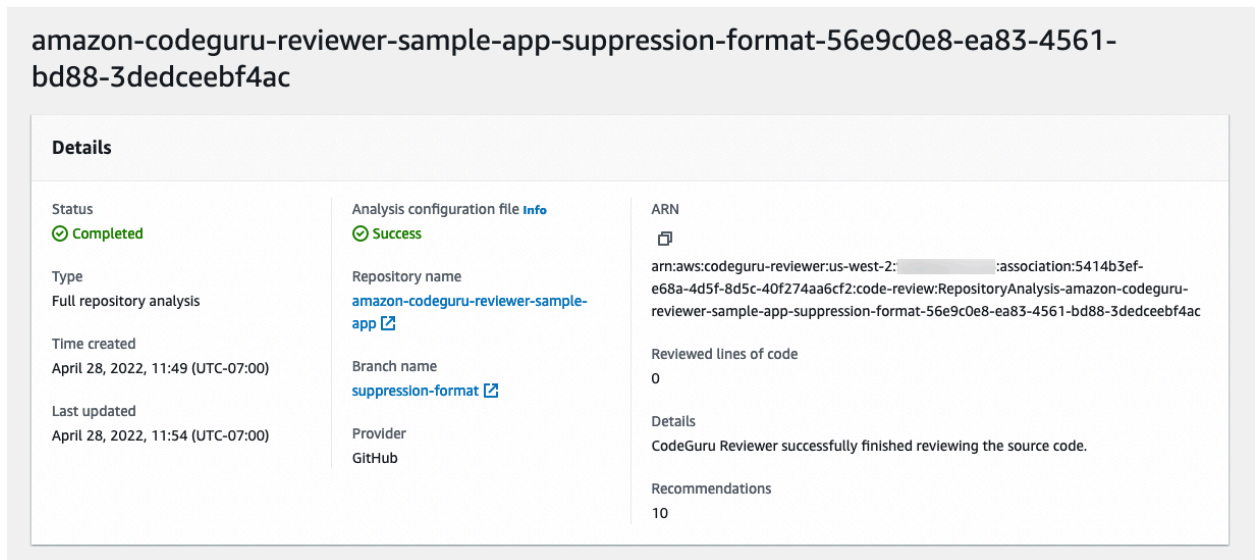
Steps to suppress recommendations

You can add an `aws-codeguru-reviewer.yml` file to a repository either *before* or *after* you associate a repository. For more information about associating a repository, see [Working with repository associations](#).

To suppress recommendations

1. Create an `aws-codeguru-reviewer.yml` file. For more information, see [Structure of the aws-codeguru-reviewer.yml file](#).
2. Add the `aws-codeguru-reviewer.yml` file to the root directory of the repository that you want CodeGuru Reviewer to analyze.
3. For new repositories, associate the repository. After you associate the repository, CodeGuru Reviewer automatically initiates a full repository analysis code review. For more information, see [Working with repository associations](#).

4. For repositories that have already been associated, initiate either a full repository analysis code review or an incremental code review. For more information, see [Create code reviews](#).
5. To confirm that CodeGuru Reviewer used your file for the code review, check the CodeGuru Reviewer console.
 - a. Choose **Code reviews**. This page lists all code reviews performed.
 - b. Choose the code review that CodeGuru Reviewer just performed.
 - If CodeGuru Reviewer used your file in the code review, then **Success** appears under **Analysis configuration file**.



- If CodeGuru Reviewer found errors in your file, then **Error** appears under **Analysis configuration file** and a message indicating the errors appears at the top of the page.

Also, **Failed** appears under **Status**, indicating that CodeGuru Reviewer did not perform a code review.

Fix your `aws-codeguru-reviewer.yml` file based on the error messages and then initiate a new full repository analysis. For more information, see [Error handling for the aws-codeguru-reviewer.yml file](#).

amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a

Details



The CodeGuru Reviewer analysis failed due to errors with your `aws-codeguru-reviewer.yml` file. Fix the errors and then create a new analysis.
Your analysis configuration file uses an unsupported version number. Learn more about the [criteria that your analysis configuration file must meet](#).

Status

Failed

Type

Full repository analysis

Time created

April 28, 2022, 07:19 (UTC-07:00)

Last updated

April 28, 2022, 07:26 (UTC-07:00)

Analysis configuration file [Info](#)

Error

Repository name

[amazon-codeguru-reviewer-sample-app](#)

Branch name

[suppression-format](#)

Provider

GitHub

ARN



arn:aws:codeguru-reviewer:us-west-2: :association:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a

Reviewed lines of code

-

Details

Amazon CodeGuru Reviewer was unable to complete the request because the CodeGuru Reviewer configuration file `aws-codeguru-reviewer.yml` had errors.

Recommendations

-

- If CodeGuru Reviewer did not recognize your file name or find the file at the root directory of your repository, then **No file detected** appears under **Analysis configuration file**. Your file must be named `aws-codeguru-reviewer.yml` and must exist in the root directory of your repository. Otherwise CodeGuru Reviewer cannot recognize that the file exists, use it in code reviews, or return error messages about problems with the file.

Confirm the name and location of your file, make any needed changes, and then initiate a new code review.

amazon-codeguru-reviewer-sample-app-suppression-format-14144a31-a9d4-45f7-b31e-bbdbfb5d486d

Details

Status

Completed

Type

Full repository analysis

Time created

April 28, 2022, 12:03 (UTC-07:00)

Last updated

April 28, 2022, 12:09 (UTC-07:00)

Analysis configuration file [Info](#)

No file detected

Repository name

[amazon-codeguru-reviewer-sample-app](#)

Branch name

[suppression-format](#)

Provider

GitHub

ARN



arn:aws:codeguru-reviewer:us-west-2: :association:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-14144a31-a9d4-45f7-b31e-bbdbfb5d486d

Reviewed lines of code

0

Details

CodeGuru Reviewer successfully finished reviewing the source code.

Recommendations

10

6. Check **Recommendations** to confirm that the recommendations match what you expect based on the settings in your `aws-codeguru-reviewer.yml` file.

Cost impact of suppressing recommendations

You are only charged for the lines of code that CodeGuru Reviewer analyzes. You are not charged for the lines of code in files or directories that you exclude in your `aws-codeguru-reviewer.yml` file. For more information, see [Amazon CodeGuru pricing](#).

If you receive an error message about the `aws-codeguru-reviewer.yml` file, CodeGuru Reviewer did not analyze your repository and you are not charged. For more information about error messages, see [Error handling for the aws-codeguru-reviewer.yml file](#).

Error handling for the aws-codeguru-reviewer.yml file

CodeGuru Reviewer does not validate your `aws-codeguru-reviewer.yml` file, but you could receive error messages under the following situations.

- When you initiate a full repository analysis code review in the console, messages about any errors appear in the CodeGuru Reviewer console.

The following image shows the details section of a successful code review.

amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a

Details



The CodeGuru Reviewer analysis failed due to errors with your `aws-codeguru-reviewer.yml` file. Fix the errors and then create a new analysis.

Your analysis configuration file uses an unsupported version number. Learn more about the [criteria that your analysis configuration file must meet](#).

Status  Failed	Analysis configuration file Info  Error	ARN 
Type Full repository analysis	Repository name amazon-codeguru-reviewer-sample-app	arn:aws:codeguru-reviewer:us-west-2: :association:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a
Time created April 28, 2022, 07:19 (UTC-07:00)	Branch name suppression-format	Reviewed lines of code -
Last updated April 28, 2022, 07:26 (UTC-07:00)	Provider GitHub	Details Amazon CodeGuru Reviewer was unable to complete the request because the CodeGuru Reviewer configuration file <code>aws-codeguru-reviewer.yml</code> had errors.
		Recommendations -

- When you submit a pull request for the analysis configuration file, messages about any errors appear as comments in the `aws-codeguru-reviewer.yml` file.
- When you initiate an incremental code review, messages about any errors appear in comments on the changed lines of code in your pull request.

In these situations, CodeGuru Reviewer does not review any files or directories, and you are not charged for the attempted analysis.

Create code reviews in Amazon CodeGuru Reviewer

Amazon CodeGuru Reviewer uses code reviews to provide [different kinds of recommendations](#) to help improve your code. These recommendations are focused on best practices and resolving potential defects in code that are difficult for developers to find. After a code review is successfully completed on a full repository analysis or incremental code review, you can view recommendations. You can then choose whether to incorporate the recommendations, and you can provide feedback about whether the recommendations were helpful.

If you want to suppress recommendations, you can add files and directories to an `aws-codeguru-reviewer.yml` file for CodeGuru Reviewer to exclude from analysis. You can create this analysis configuration file and add it to the root directory of your repository at any time. CodeGuru Reviewer uses the file for both incremental code reviews and full repository analysis code reviews, or returns errors indicating problems with the file. For more information, see [Suppress recommendations](#).

Note

We recommend that you use both CodeGuru Reviewer and traditional peer review processes during the code review stage. Using a combination of code review processes helps to identify more issues before they reach production.

There are three different kinds of code reviews that CodeGuru Reviewer can do to provide recommendations.

- *Incremental code reviews* are created automatically when you create a pull request from your repository context on an associated repository. These code reviews scan the changed code in a pull request.

- *Full repository analysis code reviews* scan all the code in a specified branch in the CodeGuru Reviewer console.
- *Full repository analysis code reviews for CI/CD workflows* scan all the source code in your CI/CD workflow. For more information, see [Create code reviews with GitHub Actions](#).

For more information on the difference between incremental code reviews and full repository analysis code reviews, see [About full repository analysis and incremental code reviews](#).

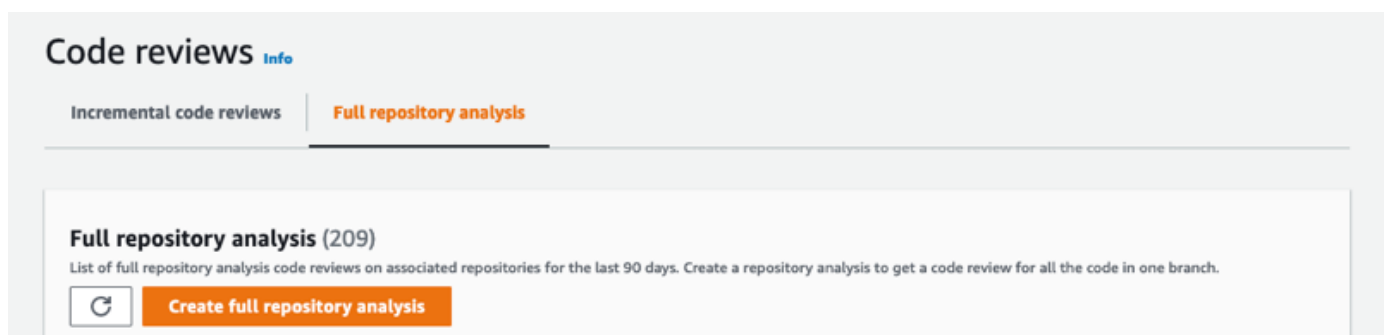
Topics

- [Get recommendations using full repository analysis](#)
- [Get recommendations using incremental code reviews](#)
- [Get recommendations using GitHub Actions](#)

Get recommendations using full repository analysis

To get recommendations on all the code in a branch, associate the repository with CodeGuru Reviewer. CodeGuru Reviewer automatically initiates a full repository analysis. Later, to initiate a full repository analysis on-demand, follow these steps:

1. Navigate to the **Code reviews** pane in the console.
2. On the **Full repository analysis** tab, choose **Create full repository analysis**.



A window opens for you to specify the location of the source code you wish to scan.

3. On the **Create full repository analysis** page, choose the associated repository from the list, then choose the branch you want reviewed.
4. (Optional) If you want to, you can provide a name for your code review. If you don't, CodeGuru Reviewer provides a name for you that you can modify.

5. (Optional) If you want to suppress recommendations, create an `aws-codeguru-reviewer.yml` file and add it to the root directory of your repository. You can download a sample file to use as a template from the **Analysis configuration file** section. For more information, see [Suppress recommendations](#).
6. When you have specified the branch you want reviewed, choose **Create full repository analysis**.

Create full repository analysis [Info](#)

Source code

Specify the branch for your associated repositories to receive code quality recommendations.

Associated repository

Before you create a code review, [associate a repository](#).

Select repository ▼

Source branch

Select a source branch for CodeGuru Reviewer to scan.

Name of source branch

Code review name - optional

Enter your code review name.

Code review name

▼ Analysis configuration file - optional

Suppress recommendations from CodeGuru Reviewer.

Create an `aws-codeguru-reviewer.yml` file that uses glob expressions to exclude files and directories in your repository for code reviews. Then add this file to the root directory of your repository. [Learn more](#)

 **Download sample aws-codeguru-reviewer.yml file**

To view the recommendations, navigate to the **Code reviews** page in the console and choose the name of the code review to view the detailed code review page. If you don't see the code review right away, try refreshing the page. For more information, see [View code review details](#).

If a repository contains Java and Python files, then CodeGuru Reviewer generates recommendations for the language for which there are more files. For example, if there are five

Java files and ten Python files in an associated repository, then CodeGuru Reviewer generates recommendations for the Python code and does not generate recommendations for the Java code. If the number of Java and Python files is the same, then only Java recommendations are generated.

Get recommendations using incremental code reviews

To get recommendations from CodeGuru Reviewer for code changes in an associated repository, referred to as an incremental code review, use the repository source provider to submit a pull request. CodeGuru Reviewer then provides recommendations to improve your code as comments on the pull request in the source provider.

You can also view the recommendations in the CodeGuru Reviewer console.

1. Navigate to the **Code reviews** pane in the console.
2. On the **Incremental code reviews** tab, choose the name of the code review.

CodeGuru Reviewer then displays the detailed code review page that contains the list of recommendations.

Get recommendations using GitHub Actions

This sections shows you how to create recommendations using GitHub Actions and disassociate a workflow, and also provides examples to get your started.

Topics

- [Create code reviews with GitHub Actions](#)
- [Disassociate your CI/CD workflow](#)
- [GitHub Actions code review examples](#)

Create code reviews with GitHub Actions

This section shows you how to create code reviews and get recommendations using GitHub Actions.

To start recommendations using GitHub Actions

1. Create an Amazon S3 bucket with the prefix **codeguru-reviewer-*** to upload your code and artifacts. For information on creating a new Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon S3 User Guide*.
2. Sign into your GitHub account to complete the CI/CD integration process. Your repository must be public or private if it's part of a GitHub organization in order for GitHub actions to work.
3. In order to run the CodeGuru Reviewer Action, you need to provide AWS credentials. We recommend using [aws-actions/configure-aws-credentials](#) to configure your credentials for a job. For self-hosted runners, the `configure-aws-credentials` action assumes the runner's IAM credentials or role to the CodeGuru Reviewer Action. Docker must be installed for self-hosted runners. For information on installing Docker, see [Get started with Docker](#).

For GitHub hosted runners, you can configure the credentials in GitHub Secrets.

The user or IAM role should have the `AmazonCodeGuruReviewerFullAccess` policy enabled and Amazon S3 Permissions (`s3:PutObject`, `s3:ListBucket`, `s3:GetObject`). For more details on AWS credentials, see [Configuration and credential file settings](#) in the *AWS Command Line Interface User Guide*.

4. Add the CodeGuru Reviewer Action. The following code snippet provides an example showing how you can enable your workflow, as supported by CodeGuru Reviewer.

```
- name: Amazon CodeGuru Reviewer Scanner
  if: ${{ always() }}
  uses: aws-actions/codeguru-reviewer@v1.1
  with:
    build_path: target # build artifact(s) directory. This is only required for
    Java repositories
    s3_bucket: codeguru-reviewermyactions-bucket # S3 Bucket with "codeguru-
    reviewer-*" prefix
```

The following is a list of parameters.

Argument	Required	Description	
s3_bucket	Yes	User-owned bucket which starts with the	

Argument	Required	Description	
		prefix codeguru-reviewer- . Must be in the same Region as your application.	
build_path	No	Path to build artifact(s) directory . JAR files in this directory are uploaded for review. The build artifacts are required to get the complete set of security recommendations.	
kms_key_id	No	The key ID uniquely identifies an AWS KMS key within an account and Region.	

- Run your workflow in GitHub to start the code analysis. When the build is complete, review your recommendations in the GitHub **Security** tab.

Disassociate your CI/CD workflow

If your CI workflow association fails, you can disassociate your repository by choosing **Disassociate repository**. If you want to associate your CI workflow later, you can associate your repository again by following set up steps.

If you want to stop CodeGuru Reviewer recommendations for your CI workflow, remove the codeguru action script from your repository's YAML file. Then, choose **Disassociate repository** to remove the repository association. On your next job run, CodeGuru Reviewer associates the repository again unless you remove the codeguru action script from the YAML file.

GitHub Actions code review examples

Run CodeGuru Reviewer Action on a GitHub hosted runner.

```
steps:
  - name: Checkout repository
    uses: actions/checkout@v2
    with:
      fetch-depth: 0    # Required

  - name: Configure AWS Credentials
    uses: aws-actions/configure-aws-credentials@v1
    if: ${{ always() }} # This ensures that your workflow runs successfully
    with:
      aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
      aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      aws-region: us-west-2

  - name: Amazon CodeGuru Reviewer Scanner
    uses: aws-actions/codeguru-reviewer@v1.1
    if: ${{ always() }}
    with:
      build_path: target # build artifact(s) directory
      s3_bucket: codeguru-reviewer-my-bucket # S3 Bucket with "codeguru-reviewer-*"
prefix

  - name: Upload review result
    if: ${{ github.event_name != 'push' }}
    uses: github/codeql-action/upload-sarif@v1
    with:
      sarif_file: codeguru-results.sarif.json
```

Run a CodeGuru Reviewer Action on a self-hosted runner.

```
steps:
  - name: Checkout repository
    uses: actions/checkout@v2
    with:
      fetch-depth: 0

  - name: Configure AWS Credentials
    if: ${{ always() }} # This ensures that your workflow runs successfully
    uses: aws-actions/configure-aws-credentials@v1
```

```
with:
  aws-region: us-west-2
  role-to-assume: my-github-actions-role

# Refer to Step 2 for more details
- name: Amazon CodeGuru Reviewer
  uses: aws-actions/codeguru-reviewer@v1.1
  if: ${{ always() }}
  with:
    build_path: target # build artifact(s) directory
    s3_bucket: codeguru-reviewermy-bucket

- name: Upload review result
  if: ${{ github.event_name != 'push' }}
  uses: github/codeql-action/upload-sarif@v1
  with:
    sarif_file: codeguru-results.sarif.json
```

View all code reviews

You can view all code reviews from the past 90 days and their statuses on the **Code reviews** page in the Amazon CodeGuru Reviewer console. There is an **incremental code review** tab to view code reviews done on incremental code reviews and a **Full repository analysis** tab to view code reviews requested for full repository analyses.

To learn about the types of recommendations, see [Amazon CodeGuru Reviewer Detector Library](#).

Code reviews page

To view this page, in the navigation pane, choose **Reviewer, Code reviews**.

CodeGuru

Dashboard
Reviewer
CI workflows
Repositories
Code reviews
Profiler
Profiling groups

CodeGuru > Code reviews

New pricing
CodeGuru Reviewer has a new lower and more predictable pricing model that offers reductions up to 90%. Easily estimate and scale your automated code reviews across your software development processes.

View pricing

Code reviews

Incremental code reviews
Full repository analysis

Incremental code reviews (14)

List of incremental pull request code reviews on associated repositories for the last 90 days. CodeGuru Reviewer automatically provides a code review when you create a pull request.

< 1 2 > ⚙

Name	Status	Repository	Recommendations	Last update
GITHUB-bugevent-5-902692c811f5ac16b9c701e90b11d9d4	Failed	bugevent	-	June 03, 2021, 13:01 (UTC-07:00)
GITHUB-bugevent-6-4f8787459b6c4a7aff65e55c958e5343	Completed	bugevent	0	June 03, 2021, 13:01 (UTC-07:00)
GITHUB-bugevent-6-dec6edd1cbb59928c9116139a6272e26	Completed	bugevent	0	June 03, 2021, 13:00 (UTC-07:00)
GITHUB-bugevent-5-e278cbdfb229208c2a6f48f80a08513	Completed	bugevent	0	June 03, 2021, 12:56 (UTC-07:00)
GITHUB-bugevent-4-d00bddddda472ca91cd71b335d899eb0	Completed	bugevent	0	June 02, 2021, 19:12 (UTC-07:00)
GITHUB-bugevent-3-68905156bc17b6f4421c9f30d5c0f399	Completed	bugevent	0	June 02, 2021, 18:25 (UTC-07:00)
GITHUB-bugevent-3-2e43e727eccecd9d2b9359a94a1fe6	Completed	bugevent	0	June 02, 2021, 18:14 (UTC-07:00)

Note

After 90 days have passed since a code review was done, you can't view that code review in the Amazon CodeGuru Reviewer console. But you might be able to view the recommendations from incremental code reviews in the repository source provider.

To view code reviews with the AWS CLI or the AWS SDK, call `ListCodeReviews`. You can filter using `ProviderType`, `RepositoryName`, or `State`. For more information, see the [Amazon CodeGuru Reviewer API Reference](#).

Navigate to repositories and pull requests

From the **Code reviews** page, you can navigate to the repository or the pull request that CodeGuru Reviewer scanned. On either the **Incremental code review** or **Full repository analysis** tab, choose a name under the **Repository** column.

View code review details

You can view a summary of code review details or a code review details page in Amazon CodeGuru Reviewer. This allows you to get information about a code review's status and generated recommendations.

Topics

- [Information in code review details](#)
- [View code review details by using the AWS CLI](#)

Information in code review details

You might want to use code review details to get more information about a code review, to provide feedback on recommendations in a code review, or to troubleshoot a **Failed** code review status.

There are three possible code review statuses:

- **Pending** – CodeGuru Reviewer has received the incremental code review notification or the full repository analysis request and a code review is scheduled. Make sure you maintain access permissions to your source branch while CodeGuru Reviewer processes the request. If the code review is for a pull request, keep the pull request open.
- **Completed** – CodeGuru Reviewer successfully finished reviewing the source code.
- **Failed** – The code review has failed to finish reviewing the source code. This could be because of a problem with source code access permissions or a transient exception that occurred:
 - If the problem is due to source code access permissions, the easiest way to fix it is to disassociate the repository and then associate the repository again. If the error persists, contact AWS Support.
 - If the problem is due to a transient exception, the code review request is retried.
 - If the problem is due to an analysis configuration file error, then review the errors with the file, fix the file, and initiate an incremental code review or a full repository analysis. For more information, see [Error handling for the aws-codeguru-reviewer.yml file](#).

When you retry the operation, be sure to keep relevant incremental code reviews open and the source branch available while CodeGuru Reviewer processes the request.

You can view code review details by choosing the name of the code review.

View code review details by using the AWS CLI

You can also use the AWS CLI or the AWS SDK to view the details of a code review.

If you have the code review ARN, you can call [DescribeCodeReview](#). Alternatively, you can call [ListCodeReviews](#) and filter using `ProviderType` and `RepositoryName`.

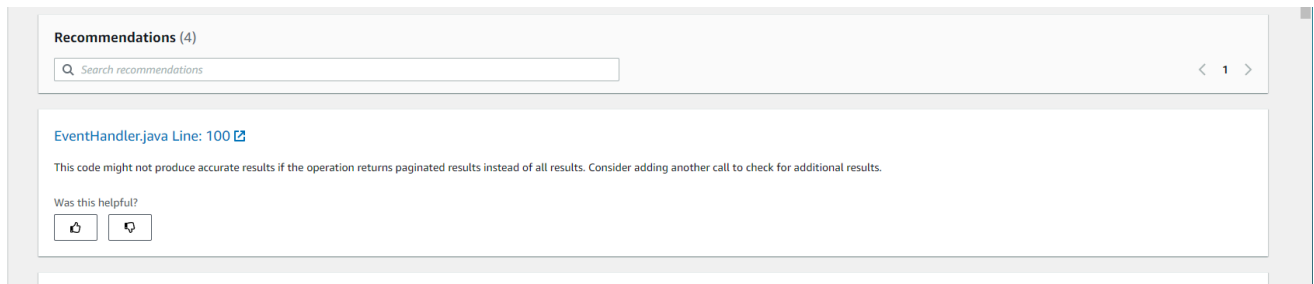
View recommendations and provide feedback

After you access the detailed code review page by choosing the name of the code review from the **Code reviews** page, you can view the recommendations from the code review directly in the console. To leave feedback on a code review in the console, do the following:

1. Navigate to the **Code reviews** page in the CodeGuru Reviewer console.
2. Choose the name of the code review from the **Code reviews** page. A detailed code review page opens.
3. In the **Recommendations** section, choose the thumbs-up or thumbs-down icon for a recommendation to indicate whether it was helpful or not.

Providing feedback can improve the quality of recommendations Amazon CodeGuru Reviewer provides for your code, making CodeGuru Reviewer increasingly effective in later analyses.

You can also view recommendations and provide feedback in incremental code reviews directly in your repository source provider, or by using the CLI. For more information, see [Step 4: Provide feedback](#).



Your feedback is used to improve CodeGuru Reviewer through model-tuning efforts that will help make CodeGuru Reviewer recommendations more useful to you and others.

Create code reviews with GitHub Actions

Amazon CodeGuru Reviewer finds issues in your Java and Python code and recommends how to remediate them. CodeGuru Reviewer detects deviation from best practices for using AWS APIs and SDKs, and also identifies concurrency issues, resource leaks, security vulnerabilities and validates input parameters validation. First, you enable CodeGuru Reviewer on your build workflow. Next, for every push, pull, or scheduled repository scan, the CodeGuru Reviewer GitHub Action copies your code and build artifacts into an S3 bucket in your AWS account. CodeGuru Reviewer APIs are used to analyze the artifacts and provide recommendations.

You can enable security and code quality recommendations with GitHub Actions by making the following changes to your workflow. If your repository has files in both Java and Python, then CodeGuru Reviewer will provide recommendations for the language that has more files. An example workflow file could be `.github/workflows/build.yml`.

Get recommendations using GitHub Actions

This sections shows you how to create recommendations using GitHub Actions and disassociate a workflow, and also provides examples to get your started.

Topics

- [Create code reviews with GitHub Actions](#)
- [Disassociate your CI/CD workflow](#)
- [GitHub Actions code review examples](#)

Create code reviews with GitHub Actions

This section shows you how to create code reviews and get recommendations using GitHub Actions.

To start recommendations using GitHub Actions

1. Create an Amazon S3 bucket with the prefix **codeguru-reviewer-*** to upload your code and artifacts. For information on creating a new Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon S3 User Guide*.

2. Sign into your GitHub account to complete the CI/CD integration process. Your repository must be public or private if it's part of a GitHub organization in order for GitHub actions to work.
3. In order to run the CodeGuru Reviewer Action, you need to provide AWS credentials. We recommend using [aws-actions/configure-aws-credentials](#) to configure your credentials for a job. For self-hosted runners, the `configure-aws-credentials` action assumes the runner's IAM credentials or role to the CodeGuru Reviewer Action. Docker must be installed for self-hosted runners. For information on installing Docker, see [Get started with Docker](#).

For GitHub hosted runners, you can configure the credentials in GitHub Secrets.

The user or IAM role should have the `AmazonCodeGuruReviewerFullAccess` policy enabled and Amazon S3 Permissions (`s3:PutObject`, `s3:ListBucket`, `s3:GetObject`). For more details on AWS credentials, see [Configuration and credential file settings](#) in the *AWS Command Line Interface User Guide*.

4. Add the CodeGuru Reviewer Action. The following code snippet provides an example showing how you can enable your workflow, as supported by CodeGuru Reviewer.

```
- name: Amazon CodeGuru Reviewer Scanner
  if: ${{ always() }}
  uses: aws-actions/codeguru-reviewer@v1.1
  with:
    build_path: target # build artifact(s) directory. This is only required for
    Java repositories
    s3_bucket: codeguru-reviewermyactions-bucket # S3 Bucket with "codeguru-
    reviewer-*" prefix
```

The following is a list of parameters.

Argument	Required	Description	
s3_bucket	Yes	User-owned bucket which starts with the prefix <code>codeguru-reviewer-</code> . Must be in the same	

Argument	Required	Description	
		Region as your application.	
build_path	No	Path to build artifact(s) directory . JAR files in this directory are uploaded for review. The build artifacts are required to get the complete set of security recommendations.	
kms_key_id	No	The key ID uniquely identifies an AWS KMS key within an account and Region.	

- Run your workflow in GitHub to start the code analysis. When the build is complete, review your recommendations in the GitHub **Security** tab.

Disassociate your CI/CD workflow

If your CI workflow association fails, you can disassociate your repository by choosing **Disassociate repository**. If you want to associate your CI workflow later, you can associate your repository again by following set up steps.

If you want to stop CodeGuru Reviewer recommendations for your CI workflow, remove the `codeguru` action script from your repository's YML file. Then, choose **Disassociate repository** to remove the repository association. On your next job run, CodeGuru Reviewer associates the repository again unless you remove the `codeguru` action script from the YML file.

GitHub Actions code review examples

Run CodeGuru Reviewer Action on a GitHub hosted runner.

steps:

- name: Checkout repository
 - uses: actions/checkout@v2
 - with:
 - fetch-depth: 0 # Required
- name: Configure AWS Credentials
 - uses: aws-actions/configure-aws-credentials@v1
 - if: \${{ always() }} # This ensures that your workflow runs successfully
 - with:
 - aws-access-key-id: \${{ secrets.AWS_ACCESS_KEY_ID }}
 - aws-secret-access-key: \${{ secrets.AWS_SECRET_ACCESS_KEY }}
 - aws-region: us-west-2
- name: Amazon CodeGuru Reviewer Scanner
 - uses: aws-actions/codeguru-reviewer@v1.1
 - if: \${{ always() }}
 - with:
 - build_path: target # build artifact(s) directory
 - s3_bucket: codeguru-reviewer-my-bucket # S3 Bucket with "codeguru-reviewer-*" prefix
- name: Upload review result
 - if: \${{ github.event_name != 'push' }}
 - uses: github/codeql-action/upload-sarif@v1
 - with:
 - sarif_file: codeguru-results.sarif.json

Run a CodeGuru Reviewer Action on a self-hosted runner.**steps:**

- name: Checkout repository
 - uses: actions/checkout@v2
 - with:
 - fetch-depth: 0
- name: Configure AWS Credentials
 - if: \${{ always() }} # This ensures that your workflow runs successfully
 - uses: aws-actions/configure-aws-credentials@v1
 - with:
 - aws-region: us-west-2
 - role-to-assume: my-github-actions-role

```
# Refer to Step 2 for more details
- name: Amazon CodeGuru Reviewer
  uses: aws-actions/codeguru-reviewer@v1.1
  if: ${{ always() }}
  with:
    build_path: target # build artifact(s) directory
    s3_bucket: codeguru-reviewermy-bucket

- name: Upload review result
  if: ${{ github.event_name != 'push' }}
  uses: github/codeql-action/upload-sarif@v1
  with:
    sarif_file: codeguru-results.sarif.json
```


Product and service integrations

By default, Amazon CodeGuru Reviewer is integrated with the following products and services. The information provided in the following table can help you configure CodeGuru Reviewer to integrate with the products and services you use.

Products and services that are integrated with Amazon CodeGuru Reviewer

AWS CloudTrail	CloudTrail captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an Amazon S3 bucket that you specify. You can configure CloudTrail to capture API calls from the CodeGuru Reviewer console, CodeGuru Reviewer commands from the AWS Command Line Interface (AWS CLI), and from the CodeGuru Reviewer API.
Amazon CloudWatch	You can use Amazon CloudWatch to monitor the number of recommendations created for your source code in an associated repository over time. For more information, see the section called “Monitoring CodeGuru Reviewer with CloudWatch” .
AWS CodeCommit	You can configure CodeGuru Reviewer to provide analysis and recommendations for repositories in CodeCommit. For more information about CodeCommit, see the AWS CodeCommit User Guide .
CodeConnections	CodeConnections is a service that allows CodeGuru Reviewer to connect to third-party repository source providers such as Bitbucket . You don't need an CodeConnections account to get analysis and recommendations for repositories.

AWS Secrets Manager

AWS Secrets Manager is a service that automatically integrates with CodeGuru Reviewer to find unprotected secrets in your code. For more information, see [Secrets detection](#) and [Create a secret](#) in the *AWS Secrets Manager User Guide*.

Bitbucket

You can configure CodeGuru Reviewer to [provide analysis and recommendations for repositories in Bitbucket](#). To do this, you must have created a Bitbucket account and at least one Bitbucket repository.

GitHub

You can configure CodeGuru Reviewer to [provide analysis and recommendations for repositories in GitHub](#). To do this, you must have created a GitHub account and at least one GitHub repository.

GitHub Enterprise Cloud

You can configure CodeGuru Reviewer to [provide analysis and recommendations for repositories in GitHub Enterprise Cloud](#) in the same way that you would for other GitHub repositories. To do this, you must have created a GitHub Enterprise Cloud organization in your account and at least one repository.

GitHub Enterprise Server

You can configure CodeGuru Reviewer to [provide analysis and recommendations for repositories in GitHub Enterprise Server](#). To do this, you must have created a GitHub Enterprise Server account and at least one repository. You should have already configured your network or virtual private cloud (VPC). You also must already have created your instance and, if you plan to connect with your VPC, launched your instance into your VPC.

Recommendation types in CodeGuru Reviewer

Amazon CodeGuru Reviewer recommends various kinds of fixes in your Java and Python code. These recommendations are based on common code scenarios and might not apply to all cases.

If you don't agree with a recommendation, you can [provide feedback](#) in the CodeGuru Reviewer console or by commenting on the code in the pull requests. Any positive or negative feedback can be used to help improve the performance of CodeGuru Reviewer so that recommendations get better over time.

If you want to suppress recommendations from CodeGuru Reviewer, you can create and add to the root directory of your repository an `aws-codeguru-reviewer.yml` file that lists files and directories to exclude from analysis. For more information, see [Suppress recommendations](#).

The following content describes the secrets detection functionality of CodeGuru Reviewer. For information about the other recommendation types and the detectors that CodeGuru Reviewer uses, see the [Amazon CodeGuru Reviewer Detector Library](#).

Secrets detection

CodeGuru Reviewer integrates with AWS Secrets Manager to use a secrets detector that finds unprotected secrets in your code. Secrets detection is automatic, so you don't need to turn it on.

The secrets detector searches for hardcoded passwords, database connection strings, user names, and more. When an unprotected secret is found during a code review, CodeGuru Reviewer generates a recommendation and displays it with your code reviews. The recommendation tells you about the unprotected secret. To immediately protect that secret, choose **Protect your credential** in the code review. This opens the Secrets Manager console to protect and manage the secret. For more information, see the [AWS Secrets Manager User Guide](#) and [View recommendations and provide feedback](#).

Topics

- [Secrets detection supported file types](#)
- [Types of secrets detected by CodeGuru Reviewer](#)

Secrets detection supported file types

The secrets detector finds unprotected secrets the following file types with a maximum file size of 100kb.

- Config files (*.config, *.cfg, *.conf, *.cnf, *.cf)
- Environment files (*.env)
- HTML files (*.html)
- Initialization files (*.ini)
- Java files (*.java)
- JSON files (*.json)
- Jupyter Notebook files (*.ipynb)
- Key files (*.key)
- Markdown files (*.md)
- Privacy Enhanced Mail files (*.pem)
- Property List files (*.plist)
- Python files (*.py)
- reStructuredText files (*.rst)
- Text files (*.txt, *.text)
- TOML files (*.toml)
- XML files (*.xml)
- YAML files (*.yaml, *.yml)

Types of secrets detected by CodeGuru Reviewer

Amazon CodeGuru Reviewer detects unprotected usernames, passwords, RSA keys, and the following secrets.

Secrets detected by CodeGuru Reviewer

Provider	Secrets detected
Amazon Web Services (AWS)	• Amazon AWS Secret Access Key

Provider	Secrets detected
Atlassian	• Atlassian API Token • Atlassian JSON Web Token • Bitbucket Server Personal Access Token
Databricks	• Databricks Access Token
Datadog	• Datadog API Key • Datadog App Key
GitHub	• GitHub Personal Access Token • GitHub OAuth Access Token • GitHub Refresh Token • GitHub App Installation Access Token • GitHub SSH Private Key
Intercom	• Intercom Access Token
Mailchimp	• Mailchimp API Key
Mailgun	• Mailgun API Key

Provider	Secrets detected
Salesforce	Private Key
SendGrid	SendGrid API Key
Shopify	- Shopify App Shared Secret - Shopify Access Token - Shopify Custom App Access Token - Shopify Private App Password
Slack	- Client ID - Client Secret
Stripe	- Stripe API Key - Stripe Live API Secret Key - Stripe Test API Secret Key - Stripe Live API Restricted Key - Stripe Test API Restricted Key - Stripe Webhook Signing Secret
Tableau	Tableau Personal access token

Provider	Secrets detected
Telegram	Telegram Bot Token
Twilio	- Twilio Account string identifier - Twilio API Key

Security in CodeGuru Reviewer

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

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

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon CodeGuru Reviewer, 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 CodeGuru Reviewer. The following topics show you how to configure CodeGuru Reviewer to meet your security and compliance objectives.

Topics

- [Data protection for CodeGuru Reviewer](#)
- [Identity and access management in CodeGuru Reviewer](#)
- [Compliance validation for CodeGuru Reviewer](#)
- [CodeGuru Reviewer and interface VPC endpoints \(AWS PrivateLink\)](#)
- [Infrastructure security in CodeGuru Reviewer](#)

Data protection for CodeGuru Reviewer

The AWS [shared responsibility model](#) applies to data protection in Amazon CodeGuru Reviewer. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks

for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

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

Topics

- [Captured data in CodeGuru Reviewer](#)
- [Data retention in CodeGuru Reviewer](#)
- [Data encryption in CodeGuru Reviewer](#)
- [Traffic privacy](#)

Captured data in CodeGuru Reviewer

CodeGuru Reviewer stores the following to create code reviews:

- Repository metadata, such as the name and owner of a repository
- Recommendations generated by CodeGuru Reviewer
- Pull request metadata, such as the author and branch of a pull request
- Feedback submitted by customers about code reviews
- The OAuth token created after you connect to your GitHub account. This does not apply to GitHub Enterprise Server or GitHub Enterprise Cloud associated repositories.
- Source code is transiently stored in memory until its code review is complete. This typically lasts a few hours. When the code review is complete, it is flushed from memory, encrypted and stored in an Amazon S3 bucket for up to 10 days, and then deleted.

Data retention in CodeGuru Reviewer

CodeGuru Reviewer stores associated repository metadata (for example, the name and owner of the repository) until the associated repository is disassociated. When you connect to a GitHub account, CodeGuru Reviewer creates an OAuth token and retains the OAuth token forever.

After an associated repository is disassociated, you can no longer see its recommendations. Recommendations and pull request metadata (for example, the branch name) are retained for one year, and then deleted. Feedback provided to help CodeGuru Reviewer improve future recommendations is retained forever.

Data encryption in CodeGuru Reviewer

Encryption is an important part of CodeGuru Reviewer security. Data in transit and at rest is encrypted by default and doesn't require you to do anything.

- **Encryption of data at rest** – Data collected by CodeGuru Reviewer is stored using Amazon Simple Storage Service and Amazon DynamoDB. The data is encrypted using their data-at-rest encryption capabilities.
- **Encryption of data in transit** – All communication between customers and CodeGuru Reviewer and between CodeGuru Reviewer and its downstream dependencies is protected using TLS connections that are signed using the Signature Version 4 signing process. All CodeGuru

Reviewer endpoints use SHA-256 certificates that are managed by AWS Private Certificate Authority. For more information, see [Signature Version 4 signing process](#) in the *Amazon Web Services General Reference* and [What is AWS Private CA?](#) in the *AWS Private Certificate Authority User Guide*.

- **Associated repository and code review encryption** – Associated repositories and code reviews are encrypted by default using a key that AWS owns and manages. If you don't want to use a key managed by AWS, you must create an AWS Key Management Service key. For more information, see [Creating keys](#) and [AWS KMS concepts](#) in the *AWS Key Management Service Developer Guide* and [Encrypting a repository association in Amazon CodeGuru Reviewer](#).

Traffic privacy

You can improve the security of associated repositories and code reviews by configuring CodeGuru Reviewer to use an interface VPC endpoint. To do this, you don't need an internet gateway, NAT device, or virtual private gateway. It also is not required to configure AWS PrivateLink, though it is recommended. For more information, see [CodeGuru Reviewer and interface VPC endpoints \(AWS PrivateLink\)](#). For more information about AWS PrivateLink and VPC endpoints, see [What is AWS PrivateLink?](#) in the *AWS PrivateLink Guide* and [Connect your VPC to services using AWS PrivateLink](#) in the *Amazon VPC User Guide*.

Identity and access management in CodeGuru Reviewer

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

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [Overview of managing access permissions to your CodeGuru Reviewer resources](#)
- [Using identity-based policies for CodeGuru Reviewer](#)
- [Using tags to control access to Amazon CodeGuru Reviewer associated repositories](#)

- [Amazon CodeGuru Reviewer permissions reference](#)
- [Troubleshooting CodeGuru Reviewer identity and access](#)

Audience

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

Service user – If you use the CodeGuru Reviewer service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more CodeGuru Reviewer 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 CodeGuru Reviewer, see [Troubleshooting CodeGuru Reviewer identity and access](#).

Service administrator – If you're in charge of CodeGuru Reviewer resources at your company, you probably have full access to CodeGuru Reviewer. It's your job to determine which CodeGuru Reviewer features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with CodeGuru Reviewer, see [Overview of managing access permissions to your CodeGuru Reviewer resources](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to CodeGuru Reviewer. To view example CodeGuru Reviewer identity-based policies that you can use in IAM, see [Customer managed policy examples](#).

Authenticating with identities

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

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

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

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [AWS Signature Version 4 for API requests](#) in the *IAM User Guide*.

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

AWS account root user

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

Federated identity

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

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

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For

information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

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

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

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

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can [switch from a user to an IAM role \(console\)](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Methods to assume a role](#) in the *IAM User Guide*.

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

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

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

Managing access using policies

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

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

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

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

Identity-based policies

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

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

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific

resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

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

Access control lists (ACLs)

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

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

Other policy types

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

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [Service control policies](#) in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached

to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see [Resource control policies \(RCPs\)](#) in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

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

Overview of managing access permissions to your CodeGuru Reviewer resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

Note

An account administrator (or administrator user) is a user with administrator privileges. For more information, see [Security best practices in IAM](#) in the *IAM User Guide*.

When you grant permissions, you decide who is getting the permissions, the resources they can access, and the actions that can be performed on those resources.

Topics

- [CodeGuru Reviewer resources and operations](#)
- [Understanding resource ownership](#)

- [Managing access to resources](#)
- [Specifying policy elements: actions, effects, and principals](#)

CodeGuru Reviewer resources and operations

In Amazon CodeGuru Reviewer, the primary resources are repository associations and code reviews. In a policy, you use an Amazon Resource Name (ARN) to identify the resource the policy applies to. In the following ARNs, the repository association ID and the code review ID are universally unique identifiers (UUIDs). For more information, see [Amazon Resource Names \(ARNs\)](#) in the *Amazon Web Services General Reference*.

Resource type	ARN format
Repository association	arn:aws:codeguru-reviewer: <i>region-ID</i> : <i>account-ID</i> :association: <i>repository-association-uuid</i>
Code review	arn:aws:codeguru-reviewer: <i>region-ID</i> : <i>account-ID</i> :code-review: <i>code-review-uuid</i>

For example, you can indicate a specific repository association with id *my-repository-association-id* in your statement using its ARN, as follows.

```
"Resource": "arn:aws:codeguru-reviewer:us-east-2:123456789012:association:my-repository-association-id"
```

To specify all resources, or if an API action does not support ARNs, use the wildcard character (*) in the Resource element, as follows.

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

To specify multiple resources in a single statement, separate their ARNs with commas, as follows.

```
"Resource": [
  "arn:aws:codeguru-reviewer:us-east-2:123456789012:association:my-repository-association-id-1",
```

```
"arn:aws:codeguru-reviewer:us-east-2:123456789012:association:my-repository-association-id-2"
]
```

CodeGuru Reviewer provides a set of operations to work with the CodeGuru Reviewer resources. For a list, see [Amazon CodeGuru Reviewer permissions reference](#).

Understanding resource ownership

The AWS account owns the resources that are created in it, regardless of who created the resources. Specifically, the resource owner is the AWS account of the [principal entity](#) (that is, the root account, user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a rule, your AWS account is the owner of the CodeGuru Reviewer resource.
- If you grant permissions to create CodeGuru Reviewer resources to a user, the user can create CodeGuru Reviewer resources. However, your AWS account, to which the user belongs, owns the CodeGuru Reviewer resources.
- If you create an IAM role in your AWS account with permissions to create CodeGuru Reviewer resources, anyone who can assume the role can create CodeGuru Reviewer resources. Your AWS account, to which the role belongs, owns the CodeGuru Reviewer resources.

Managing access to resources

A permissions policy describes who has access to which resources.

Note

This section discusses the use of IAM in CodeGuru Reviewer. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [IAM JSON policy reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based policies* (IAM policies). Policies attached to a resource are referred to as *resource-based policies*. CodeGuru Reviewer supports identity-based (IAM policies) only.

Identity-based policies

You can attach policies to IAM identities. To grant a user permissions to view repository associations and code reviews in the CodeGuru Reviewer console, you can attach a permissions policy to a role that the user has. For IAM best practices, see [Security best practices in IAM](#) in the *IAM User Guide*.

In CodeGuru Reviewer, identity-based policies are used to manage permissions to the resources related to associated repositories and code reviews. For example, you can control access to code reviews.

You can create IAM policies to restrict the calls and resources that users in your account have access to, and then attach those policies to IAM roles that users can use. For more information about how to create IAM roles and to explore example IAM policy statements for CodeGuru Reviewer, see [Customer managed policy examples](#).

Specifying policy elements: actions, effects, and principals

For each CodeGuru Reviewer resource, the service defines a set of API operations. To grant permissions for these API operations, CodeGuru Reviewer defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action to perform the API operation. For more information, see [CodeGuru Reviewer resources and operations](#) and [Amazon CodeGuru Reviewer permissions reference](#).

The following are the basic policy elements:

- **Resource** – You use an ARN to identify the resource that the policy applies to.
- **Action** – You use action keywords to identify resource operations to allow or deny. For example, the `codeguru-reviewer:DisassociateRepository` permission gives the user permissions to perform the [DisassociateRepository](#) operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user cannot access a resource, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions.

To learn more about IAM policy syntax and descriptions, see [IAM JSON policy reference](#) in the *IAM User Guide*.

For a table showing all of the CodeGuru Reviewer API actions and the resources they apply to, see [Amazon CodeGuru Reviewer permissions reference](#).

Using identity-based policies for CodeGuru Reviewer

By default, users and IAM roles don't have permission to create or modify Amazon CodeGuru Reviewer resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the roles or groups that require those permissions. To learn how to attach policies to an IAM role or group, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.

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

Topics

- [Policy best practices](#)
- [Permissions required to use the CodeGuru Reviewer console](#)
- [AWS managed \(predefined\) policies for CodeGuru Reviewer](#)
- [CodeGuru Reviewer updates to AWS managed policies](#)
- [Customer managed policy examples](#)

Policy best practices

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

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.

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

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Permissions required to use the CodeGuru Reviewer console

A user who uses the CodeGuru Reviewer console must have a minimum set of permissions that allows the user to describe other AWS resources for the AWS account. You must have permissions from the following services:

- CodeGuru Reviewer
- AWS CodeCommit (if your source code is in a CodeCommit repository)
- CodeConnections (if your source code is in a repository managed by CodeConnections, such as Bitbucket)
- AWS Identity and Access Management (IAM)

If your source code is in a GitHub repository, you must have an OAuth token to connect to it. Associated GitHub repositories are not managed by CodeConnections. For more information, see [Git automation with OAuth tokens](#) on the GitHub website.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended.

The following shows an example of a permissions policy that allows a user to get information about a repository association only in the us-east-2 Region for account 123456789012 for any repository association with a universally unique identifier (UUID) that starts with 12345.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeguru-reviewer:DescribeRepositoryAssociation",
      "Resource": "arn:aws:codeguru-reviewer:us-east-2:123456789012:association:12345*"
    }
  ]
}
```

AWS managed (predefined) policies for CodeGuru Reviewer

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS managed policies](#) in the *IAM User Guide*.

To create and manage CodeGuru Reviewer service roles, you must also attach the AWS managed policy named `IAMFullAccess`.

You can also create your own custom IAM policies to allow permissions for CodeGuru Reviewer actions and resources. You can attach these custom policies to the roles or groups that require those permissions.

The following AWS managed policies, which you can attach to users in your account, are specific to CodeGuru Reviewer.

Topics

- [AmazonCodeGuruReviewerFullAccess](#)
- [AmazonCodeGuruReviewerReadOnlyAccess](#)
- [AmazonCodeGuruReviewerServiceRolePolicy](#)

AmazonCodeGuruReviewerFullAccess

AmazonCodeGuruReviewerFullAccess – Provides full access to CodeGuru Reviewer, including permissions to tag repository associations and to create, update, and delete code reviews and repository associations. It also grants permission to related resources in other services that integrate with CodeGuru Reviewer, such as Amazon CloudWatch, CodeConnections, and CodeCommit. Apply this only to administrative-level users to whom you want to grant full control over CodeGuru Reviewer repository associations, code reviews, and related resources in your AWS account, including the ability to delete code reviews and repository associations.

The **AmazonCodeGuruReviewerFullAccess** policy contains the following statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRDeletion",
```

```

    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer"
},
{
    "Sid": "codecommitAccess",
    "Effect": "Allow",
    "Action": [
        "codecommit:ListRepositories"
    ],
    "Resource": "*"
},
{
    "Sid": "codecommitTagManagement",
    "Effect": "Allow",
    "Action": [
        "codecommit:TagResource",
        "codecommit:UntagResource"
    ],
    "Resource": "*",
    "Condition": {
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "codeguru-reviewer"
        }
    }
},
{
    "Sid": "CodeConnectTagManagement",
    "Effect": "Allow",
    "Action": [
        "codestar-connections:TagResource",
        "codestar-connections:UntagResource",
        "codestar-connections:ListTagsForResource"
    ],
    "Resource": "*",
    "Condition": {
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "codeguru-reviewer"
        }
    }
}

```

```

    },
    {
      "Sid": "CodeConnectManagedRules",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:UseConnection",
        "codestar-connections:ListConnections",
        "codestar-connections:PassConnection"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "codestar-connections:ProviderAction": [
            "ListRepositories",
            "ListOwners"
          ]
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events>DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
  ]
}

```

AmazonCodeGuruReviewerReadOnlyAccess

AmazonCodeGuruReviewerReadOnlyAccess – Grants read-only access to CodeGuru Reviewer and related resources in other AWS services. Apply this policy to users who you want to grant the ability to view code reviews, but not to create or make any changes to them.

The `AmazonCodeGuruReviewerReadOnlyAccess` policy contains the following statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:List*",
        "codeguru-reviewer:Describe*",
        "codeguru-reviewer:Get*"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonCodeGuruReviewerServiceRolePolicy

`AmazonCodeGuruReviewerServiceRolePolicy` – Grants permission to related resources in CodeCommit, CodeConnections, Amazon S3, and CloudWatch that are required to create repository associations.

For CodeCommit repository associations, the CodeCommit and CloudWatch permissions in this policy are required. For associations with repositories that are managed by an AWS CodeStar connection, such as Bitbucket, the CodeConnections permissions are required. For code reviews with security analysis, the Amazon S3 permissions are required.

When you create your first association with a CodeCommit, Amazon S3, or CodeConnections managed repository, CodeGuru Reviewer adds the `AmazonCodeGuruReviewerServiceRolePolicy` policy to your AWS account. This policy grants CodeGuru Reviewer access to CodeCommit repositories, CodeConnections resources in your account that have an `aws:ResourceTag/codeguru-reviewer` tag. It also grants access to Amazon S3 buckets that have a prefix that begins with `codeguru-reviewer-`. When you associate a CodeCommit repository, CodeGuru Reviewer adds this tag to the repository. When you associate an CodeConnections managed repository, CodeGuru Reviewer adds this tag to the CodeConnections resource, if it doesn't already exist.

The `AmazonCodeGuruReviewerServiceRolePolicy` policy contains the following statement.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessCodeGuruReviewerEnabledRepositories",
      "Effect": "Allow",
      "Action": [
        "codecommit:GetRepository",
        "codecommit:GetBranch",
        "codecommit:DescribePullRequestEvents",
        "codecommit:GetCommentsForPullRequest",
        "codecommit:GetDifferences",
        "codecommit:GetPullRequest",
        "codecommit:ListPullRequests",
        "codecommit:PostCommentForPullRequest",
        "codecommit:GitPull",
        "codecommit:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/codeguru-reviewer": "enabled"
        }
      }
    },
    {
      "Sid": "AccessCodeGuruReviewerEnabledConnections",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:UseConnection"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "codestar-connections:ProviderAction": [
            "ListBranches",
            "GetBranch",
            "ListRepositories",
            "ListOwners",
            "ListPullRequests",
            "GetPullRequest",
            "ListPullRequestComments",
            "ListPullRequestCommits",

```

```

        "ListCommitFiles",
        "ListBranchCommits",
        "CreatePullRequestDiffComment",
        "GitPull"
    ]
},
"Null": {
    "aws:ResourceTag/codeguru-reviewer": "false"
}
},
{
    "Sid": "CloudWatchEventsResourceCleanup",
    "Effect": "Allow",
    "Action": [
        "events:DeleteRule",
        "events:RemoveTargets"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
    }
},
{
    "Sid": "AccessCodeGuruReviewerCreatedS3Bucket",
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:PutBucketPolicy",
        "s3:PutLifecycleConfiguration"
    ],
    "Resource": [
        "arn:aws:s3:::codeguru-reviewer-*",
        "arn:aws:s3:::codeguru-reviewer-*/*"
    ]
}
]
}

```

CodeGuru Reviewer updates to AWS managed policies

View details about updates to AWS managed policies for CodeGuru Reviewer since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CodeGuru Reviewer [Amazon CodeGuru Reviewer User Guide document history](#).

Change	Description	Date
AmazonCodeGuruReviewerServiceRolePolicy – Update to an existing policy	CodeGuru Reviewer added new permissions to allow access to the <code>CreateBucket</code> , <code>ListBucket</code> , <code>PutBucketPolicy</code> , and <code>PutLifecycleConfiguration</code> actions on an Amazon S3 bucket resource.	April 28, 2021
CodeGuru Reviewer started tracking changes	CodeGuru Reviewer started tracking changes for its AWS managed policies.	July 2, 2020

Customer managed policy examples

You can create your own custom IAM policies to allow permissions for CodeGuru Reviewer actions and resources. You can attach these custom policies to the roles or groups that require those permissions. You can also create your own custom IAM policies for integration between CodeGuru Reviewer and other AWS services.

The following example IAM policies grant permissions for various CodeGuru Reviewer actions. Use them to limit CodeGuru Reviewer access for your users and roles. These policies control the ability to perform actions with the CodeGuru Reviewer console, API, AWS SDKs, or the AWS CLI.

Note

All examples use the US East (Ohio) Region (us-east-2) and contain fictitious account IDs.

Examples

- [Example 1: Allow a user to see all recommendations created in an associated repository](#)
- [Example 2: Allow a user to view code reviews in an associated repository in a single Region](#)
- [Example 3: Allow a user to perform CodeGuru Reviewer operations in a single Region](#)
- [Example 4: Allow read-only access to CodeGuru Reviewer operations for a user connecting from a specified IP address range](#)

Example 1: Allow a user to see all recommendations created in an associated repository

The following example policy grants permissions for the AWS user with account ID 123456789012 to see a list of all recommendations in their AWS account and Region in the repository association with ID `association-uuid`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:ListRecommendations"
      ],
      "Resource": "arn:aws:codeguru-reviewer:us-east-2:123456789012:association:association-uuid"
    }
  ]
}
```

Example 2: Allow a user to view code reviews in an associated repository in a single Region

The following shows an example of a permissions policy that allows a user with account ID 123456789012 to get information about code reviews in Region `us-east-2` in an associated repository with ID `association-uuid`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeguru-reviewer:DescribeCodeReview",

```



```

    "Resource": "arn:aws:codeguru-reviewer:us-
east-2:123456789012:association:association-uuid"
  }
]
}

```

Example 3: Allow a user to perform CodeGuru Reviewer operations in a single Region

The following permissions policy uses a wildcard character ("codeguru-reviewer:*") to allow users to perform all CodeGuru Reviewer actions in the us-east-2 Region and not from other AWS Regions.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeguru-reviewer:*",
      "Resource": "arn:aws:codeguru-reviewer:us-east-2:123456789012:*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-2"
        }
      }
    }
  ]
}

```

Example 4: Allow read-only access to CodeGuru Reviewer operations for a user connecting from a specified IP address range

You can create a policy that only allows users CodeGuru Reviewer read-only access if their IP address is within a certain IP address range. The following example grants read-only CodeGuru Reviewer permissions to users whose IP addresses are within the specified IP address block of 203.0.113.0/24.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",

```

```
"Action": [
    "codeguru-reviewer:List*",
    "codeguru-reviewer:Describe*"
],
"Resource": "*",
"Condition": {
    "IpAddress": {
        "aws:SourceIp": "203.0.113.0/24"
    }
}
}
```

Using tags to control access to Amazon CodeGuru Reviewer associated repositories

Conditions in IAM policy statements are part of the syntax that you can use to specify permissions to CodeGuru Reviewer associated repository-based actions. You can create a policy that allows or denies actions on associated repositories based on the tags associated with those associated repositories, and then apply those policies to the IAM groups you configure for managing users. For information about applying tags to an associated repository using the console or AWS CLI, see [Add a tag to a CodeGuru Reviewer associated repository](#). For information about applying tags using the CodeGuru Reviewer SDK, see [AssociateRepository](#) in the *Amazon CodeGuru Reviewer API Reference*. For information about using tags to control access to AWS resources, see [Controlling Access to AWS Resources Using Resource Tags](#) in the *IAM User Guide*.

You can directly use tags on an associated repository to affect permissions on the following CodeGuru Reviewer API operations:

- `AssociateRepository`
- `DescribeRepositoryAssociation`
- `DisassociateRepositoryAssociation`

You can use tags on an associated repository to indirectly affect permissions on a code review that belongs to the associated repository. Use tags on an associated repository to affect permissions on the following CodeGuru Reviewer API operations that are related to code reviews:

- `CreateCodeReview`

- ListRecommendations
- DescribeCodeReview

Example Example 1: Limit CodeGuru Reviewer associated repository actions based on request tags

The following policy denies users permission to the `DisassociateRepositoryAssociation` action if the request contains a tag with the key `ViewAssociatedRepositoryDetails` and the key value `DenyViewRepository`. In addition, the policy prevents these unauthorized users from disassociating repositories by using the `aws:TagKeys` condition key to not allow `DisassociationAllowed` if the request contains a tag with the key `DenyDisassociate`. An administrator must attach this IAM policy in addition to the managed user policy to users who are not authorized to perform these actions. The `aws:RequestTag` condition key is used to control which tags can be passed in an IAM request

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codeguru-reviewer:DescribeRepositoryAssociation"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:RequestTag/ViewAssociatedRepositoryDetails": "DenyViewRepository"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codeguru-reviewer:DisassociateRepository"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["DenyDisassociate"]
        }
      }
    }
  ]
}
```

```

    }
  ]
}
```

Example Example 2: Deny or allow actions on code reviews based on their associated repository's resource tags

You can create a policy that allows or denies actions on CodeGuru Reviewer code reviews by using the CodeGuru Reviewer tags that are added to their associated repositories. An associated repository contains code reviews, and you can use tags on the associated repository to affect permissions on its code reviews. For example, you can create a policy that denies users the ability to view recommendations created by code reviews in an associated repository. The following policy denies a user with AWS account ID 123456789012 in the AWS Region us-west-2 from viewing recommendations created by code reviews in all associated repositories that have a Recommendation tag with a value of Secret.

```

{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "codeguru-reviewer:ListRecommendations"
      ]
      "Resource" : "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:*",
      "Condition" : {
        "StringEquals" : "aws:ResourceTag/Recommendations": "Secret"
      }
    }
  ]
}
```

Example Example 3: Limit all possible CodeGuru Reviewer actions to associated repositories based on resource tags

You can create policies that selectively allow CodeGuru Reviewer actions on all associated repositories that are not tagged with specific tags. For example, the following policy allows you to associate, disassociate, and view the details of associated repositories that are not tagged with the specified tags:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:DisassociateRepositoryAssociation"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/Status": "AssociatedRepositoryAllow",
          "aws:ResourceTag/Team": "Saanvi"
        }
      }
    }
  ]
}
```

Amazon CodeGuru Reviewer permissions reference

You can use AWS condition keys in your CodeGuru Reviewer policies to express conditions. For a list, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

You specify the actions in the policy's Action field. To specify an action, use the `codeguru-reviewer:` prefix followed by the API operation name (for example, `codeguru-reviewer:AssociateRepository` and `codeguru-reviewer:DisassociateRepository`). To specify multiple actions in a single statement, separate them with commas (for example, `"Action": [ "codeguru-reviewer:AssociateRepository", "codeguru-reviewer:DisassociateRepository" ]`).

Using wildcard characters

You specify an Amazon Resource Name (ARN), with or without a wildcard character (*), as the resource value in the policy's Resource field. You can use a wildcard to specify multiple actions or resources. For example, `codeguru-reviewer:*` specifies all CodeGuru Reviewer actions and `codeguru-reviewer:List*` specifies all CodeGuru Reviewer actions that begin with the word `List`. The following example refers to all repository associations with a universally unique identifier (UUID) that begins with `PullRequest-GITHUB`.

```
arn:aws:codeguru-reviewer:us-east-2:123456789012:association:PullRequest-GITHUB*
```

You can use the following table as a reference when you are setting up [Authenticating with identities](#) and writing permissions policies that you can attach to an IAM identity (identity-based policies).

CodeBuild API operations and required permissions for actions

AssociateRepository

Action: codeguru-reviewer:AssociateRepository

Required to associate a repository with CodeGuru Reviewer.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*:code-review:*code-review-uuid*

DescribeCodeReview

Action: codeguru-reviewer:DescribeCodeReview

Required to view information about a code review, including its status.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*:code-review:*code-review-uuid*

DescribeRecommendationFeedback

Action: codeguru-reviewer:DescribeRecommendationFeedback

Required to view customer feedback about a recommendation.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*:code-review:*code-review-uuid*

DescribeRepositoryAssociation

codeguru-reviewer:DescribeRepositoryAssociation Required to get information about a repository association and its status details.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*

DisassociateRepository

Action: codeguru-reviewer:DisassociateRepository

Required to remove the association between CodeGuru Reviewer and a repository.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*

ListCodeReviews

Action: codeguru-reviewer:ListCodeReviews

Required to view the names of all code reviews in the current AWS account that were created in the past 90 days.

Resource: *

ListRecommendationFeedback

Action: codeguru-reviewer:ListRecommendationFeedback

Required to list all users' customer feedback for a code review recommendation.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*:code-review:*code-review-uuid*

ListRecommendations

Action: codeguru-reviewer:ListRecommendations

Required to view a list of all the recommendations for one completed code review.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*:code-review:*code-review-uuid*

ListRepositoryAssociations

Action: codeguru-reviewer:ListRepositoryAssociations

Required to list summary information about repository associations.

Resource: arn:aws:codeguru-reviewer:*region-ID*:*account-ID*:association:*repository-association-uuid*

PutRecommendationFeedback

Action: `codeguru-reviewer:PutRecommendationFeedback`

Required to store feedback for a code review recommendation.

Resource: `arn:aws:codeguru-reviewer:region-ID:account-ID:association:repository-association-uuid:code-review:code-review-uuid`

Troubleshooting CodeGuru Reviewer identity and access

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

Topics

- [I am not authorized to perform an action in CodeGuru Reviewer](#)
- [I am not authorized to perform iam:PassRole](#)

I am not authorized to perform an action in CodeGuru Reviewer

If the AWS Management Console tells you that you're not authorized to perform an action, you must contact your administrator for assistance.

The following example error occurs when the user `mateojackson` tries to use the console to view details about a code review, but does not have `codeguru-reviewer:DescribeCodeReview` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
codeguru-reviewer:DescribeCodeReview on resource: my-example-code-review
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-example-code-review` resource using the `codeguru-reviewer:DescribeCodeReview` action.

I am not authorized to perform iam:PassRole

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

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 CodeGuru Reviewer. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

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

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

Compliance validation for CodeGuru Reviewer

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

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

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

- [Security Compliance & Governance](#) – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- [HIPAA Eligible Services Reference](#) – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map

the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).

- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

CodeGuru Reviewer and interface VPC endpoints (AWS PrivateLink)

You can use VPC endpoints when you call Amazon CodeGuru Reviewer APIs. When you use VPC endpoints, your API calls are more secure because they are contained within your VPC and don't access the internet. For more information, see [Actions](#) in the *Amazon CodeGuru Reviewer API Reference*.

You establish a private connection between your VPC and CodeGuru Reviewer by creating an *interface VPC endpoint*. Interface endpoints are powered by [AWS PrivateLink](#), a technology that enables you to privately access CodeGuru Reviewer APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with CodeGuru Reviewer APIs. Traffic between your VPC and CodeGuru Reviewer does not leave the Amazon network.

Each interface endpoint is represented by one or more [Elastic network interfaces](#) in your subnets.

For more information, see [Access an AWS service using an interface VPC endpoint](#) in the *AWS PrivateLink Guide*.

Note

CodeGuru Reviewer does not support Amazon VPC endpoint policies.

Considerations for CodeGuru Reviewer VPC endpoints

Before you set up an interface VPC endpoint for CodeGuru Reviewer, review [Considerations](#) and [Prerequisites](#) in the *AWS PrivateLink Guide*.

CodeGuru Reviewer supports making calls to all of its API actions from your VPC.

VPC endpoint policies are not supported for CodeGuru Reviewer. By default, full access to CodeGuru Reviewer is allowed through the endpoint.

Creating an interface VPC endpoint for CodeGuru Reviewer

You can create a VPC endpoint for the CodeGuru Reviewer service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI).

Create a VPC endpoint for CodeGuru Reviewer using the following service name:

- com.amazonaws.*region*.codeguru-reviewer

If you enable private DNS for the endpoint, you can make API requests to CodeGuru Reviewer using its default DNS name for the Region, for example, `codeguru-reviewer.us-east-1.amazonaws.com`.

For more information, see [Access an AWS service using an interface VPC endpoint](#) in the *AWS PrivateLink Guide*.

Infrastructure security in CodeGuru Reviewer

As a managed service, Amazon CodeGuru Reviewer is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access CodeGuru Reviewer through the network. Clients must support the following:

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

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

Logging and monitoring in CodeGuru Reviewer

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon CodeGuru Reviewer and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure, if one occurs. AWS provides the following tools for monitoring your CodeGuru Reviewer resources and builds and for responding to potential incidents.

Topics

- [Logging CodeGuru Reviewer API calls with AWS CloudTrail](#)
- [Monitoring CodeGuru Reviewer with Amazon CloudWatch](#)

Logging CodeGuru Reviewer API calls with AWS CloudTrail

Amazon CodeGuru Reviewer is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CodeGuru Reviewer. CloudTrail captures API calls for CodeGuru Reviewer as events. The calls captured include calls from the CodeGuru Reviewer console, the CodeGuru Reviewer AWS CLI, and code calls to the CodeGuru Reviewer API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for CodeGuru Reviewer. 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 CodeGuru Reviewer, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see [What is AWS CloudTrail?](#) in the *AWS CloudTrail User Guide*.

CodeGuru Reviewer information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in CodeGuru Reviewer, that activity is recorded in a CloudTrail event with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing events with CloudTrail Event history](#) in the *AWS CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for CodeGuru Reviewer, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act on the event data collected in CloudTrail logs. For more information, see the following articles in the *AWS CloudTrail User Guide*:

- [Creating a trail for your AWS account](#)
- [AWS service integrations with CloudTrail Logs](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

CodeGuru Reviewer supports logging the following actions as events in CloudTrail log files:

- [AssociateRepository](#)
- [DescribeCodeReview](#)
- [DescribeRecommendationFeedback](#)
- [DescribeRepositoryAssociation](#)
- [DisassociateRepository](#)
- [ListCodeReviews](#)
- [ListRecommendationFeedback](#)
- [ListRecommendations](#)
- [ListRepositoryAssociations](#)
- [PutRecommendationFeedback](#)

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

For more information, see the [CloudTrail userIdentity element](#) in the *AWS CloudTrail User Guide*.

Example: CodeGuru Reviewer log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. 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, and so on. 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 that demonstrates the `AssociateRepository` action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AAAAAAAAAEXAMPLE:TestSession",
    "arn": "arn:aws:sts::123456789012:assumed-role/TestRole/TestSession",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-11-27T02:06:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/TestRole",
        "accountId": "123456789012",
        "userName": "TestRole"
      }
    }
  },
  "eventTime": "2019-11-27T03:46:35Z",
  "eventSource": "codeguru-reviewer.amazonaws.com",
  "eventName": "AssociateRepository",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "52.13.164.128",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.672
Linux/4.14.138-99.102.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201
vendor/Oracle_Corporation exec-env/AWS_Lambda_java8",
```

```

    "requestParameters": {
      "ClientRequestToken": "7485aa2f-ce15-4bc6-a6cc-2a76d702f15f",
      "Repository": {
        "CodeCommit": {
          "Name": "repository-name"
        }
      }
    },
    "responseElements": {
      "RepositoryAssociation": {
        "AssociationArn": "arn:aws:codeguru-reviewer:us-west-2:123456789012:association:6eda8e7a-319a-4750-bca8-7f73a816fadc",
        "AssociationId": "6eda8e7a-319a-4750-bca8-7f73a816fadc",
        "CreatedTimeStamp": 1574826395.662,
        "LastUpdatedTimeStamp": 1574826395.662,
        "Name": "TestRepository",
        "Owner": "123456789012",
        "ProviderType": "CodeCommit",
        "State": "Associating",
        "StateReason": "Pending Repository Association"
      }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }

```

Monitoring CodeGuru Reviewer with Amazon CloudWatch

You can use Amazon CloudWatch to monitor the number of recommendations created for your source code in an associated repository over time.

The recommendations are available for three *dimensions*:

- **ProviderType** – View the number of recommendations for a provider type. You can view the count of recommendations in all repositories in AWS CodeCommit, your Bitbucket account, your GitHub account, or your GitHub Enterprise Server account, over a period of time.

- **CodeReviewType** – View the number of recommendations for a code review type. The one available code review type is **PullRequest**. Use it to view the count of recommendations in one pull request.
- **RepositoryName** – View the count of recommendations for one repository over a period of time.

You can set a CloudWatch alarm that notifies you when the number of recommendations exceeds a threshold you set.

For more information about creating and using CloudWatch alarms and metrics, see [Using Amazon CloudWatch alarms](#).

You can track the following metric for each dimension over a period of time.

Metric	Description
RecommendationsPublishedCount	The number of recommendations over a period of time per ProviderType , CodeReviewType , or RepositoryName for completed code reviews. Units: Count Valid CloudWatch statistic: Count Valid CloudWatch period: 1 hour

Topics

- [Monitoring recommendations with CloudWatch metrics](#)
- [Monitoring CodeGuru Reviewer recommendations with CloudWatch alarms](#)

Monitoring recommendations with CloudWatch metrics

You can view Amazon CodeGuru Reviewer metrics in the Amazon CloudWatch console.

To access recommendation metrics

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **AWS/CodeGuruReviewer**.
4. Choose the dimension you want metrics for: **ProviderType**, **CodeReviewType**, or **RepositoryName**. The graph on the page displays metrics for recommendations for all selected items that are available for the selected dimension.

Monitoring CodeGuru Reviewer recommendations with CloudWatch alarms

You can create an Amazon CloudWatch alarm for your CodeGuru Reviewer recommendations to monitor their count over time.

An alarm watches the number of recommendations for one of three CodeGuru Reviewer CloudWatch dimensions that you specify:

- **ProviderType** – View the number of recommendations for a provider type. You can view the count of recommendations in all repositories in AWS CodeCommit, your Bitbucket account, your GitHub account, or your GitHub Enterprise Server account, over a period of time.
- **CodeReviewType** – View the number of recommendations for a code review type. The one available code review type is **PullRequest**. Use it to view the count of recommendations in one pull request.
- **RepositoryName** – View the count of recommendations for one repository over a period of time.

You set one or more actions that happen when the number of recommendations for a dimension exceeds a count over a number of time periods you choose. For example, you can specify that an Amazon SNS notification is sent when more than 25 recommendations are generated for a branch in a repository within an hour.

A user or role must have CloudWatch **PutMetricAlarm** permissions to create an alarm. For more information, see [Using identity-based policies for CodeGuru Reviewer](#) and [Amazon CloudWatch permissions reference](#) in the *Amazon CloudWatch User Guide*.

To create a CloudWatch alarm for CodeGuru Reviewer recommendations

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Choose **Create alarm**.
4. Choose **Select metric**.
5. Choose **AWS/CodeGuruReviewer**.
6. Choose the dimension to monitor: **ProviderType**, **CodeReviewType**, or **RepositoryName**. Then choose a metric to create an alarm for.
7. Continue through the process to create your alarm.

For more information about setting up CloudWatch alarms in the CloudWatch console, see [Using Amazon CloudWatch alarms](#) in the *Amazon CloudWatch User Guide*.

Quotas for CodeGuru Reviewer

The following table lists the current quota in Amazon CodeGuru Reviewer. This quota is for each supported AWS Region for each AWS account.

Repositories

Resource	Default
Maximum repository size	4 GB
CodeCommit repositories	
Maximum number of analyzed pull requests per month	5,000
Source code files	
Maximum Java source code size	300 MB
Maximum Python source code size	50 MB

Tags

Tag limits apply to tags on CodeGuru Reviewer associated repository resources.

Resource	Default
Maximum number of tags you can associate with a resource	50 (tags are case sensitive).
Resource tag key names	Any combination of Unicode letters, numbers, spaces, and allowed characters in UTF-8 between 1 and 127 characters in length. Allowed characters are + - = . _ : / @.

Resource	Default
	Tag key names must be unique, and each key can only have one value. A tag key name cannot: - begin with <code>aws :</code> - consist only of spaces - end with a space - contain emojis or any of the following characters: <code>? ^ * [\ ~ ! # \$ % & * () > < " ' ` [] { } ;</code>
Resource tag values	Any combination of Unicode letters, numbers, spaces, and allowed characters in UTF-8 between 0 and 255 characters in length. Allowed characters are <code>+ - = . _ : / @</code>. A key can only have one value, but many keys can have the same value. A tag key value cannot contain emojis or any of the following characters: <code>? ^ * [\ ~ ! # \$ % & * () > < " ' ` [] { } ; .</code>

CodeGuru Reviewer quotas for creating, deploying, and managing an API

The following fixed quotas apply to creating, deploying, and managing an API in CodeGuru Reviewer, using the AWS CLI, the API Gateway console, or the API Gateway REST API and its SDKs. These quotas can't be increased.

The default quota for all except three CodeGuru Reviewer APIs is 1 request per second per account. None of these quotas can be increased. For a list of all CodeGuru Reviewer APIs, see [Amazon CodeGuru Reviewer Actions](#).

The three APIs with different default quotas are in the following table.

Action	Default quota	Can be increased
AssociateRepository	1 request every 2 seconds per account	No
CreateCodeReview	1 request every 2 seconds per account	No
PutRecommendationFeedback	2 request per second per account	No

Troubleshooting

This section helps you troubleshoot common problems you might encounter when working with Amazon CodeGuru Reviewer.

Topics

- [Where can I check the status of a repository association?](#)
- [Where can I check the status of a code review?](#)
- [Where can I check the status of a third-party source provider connection?](#)
- [My repository is in an associated state. Why don't I see recommendations?](#)
- [Why did my association fail?](#)
- [Why did my code review fail?](#)
- [What if I disagree with the recommendation?](#)
- [How do I suppress a recommendation?](#)
- [The repository status has been associating for more than 5 minutes. What should I do?](#)
- [The code review status has been Pending for more than 15 minutes. What should I do?](#)
- [How do you access a repository if its owner is no longer available?](#)
- [Can I use the same AWS CodeStar connection to access repositories in two different accounts?](#)
- [I'm trying to connect to my third-party repositories. What is the difference between an app installation and a connection? Which one can be used to adjust permissions?](#)
- [How do I know if CodeGuru Reviewer used my aws-codeguru-reviewer.yml file in a code review?](#)
- [Why didn't my costs decrease when I used an aws-codeguru-reviewer.yml file?](#)

Where can I check the status of a repository association?

You can check the status of a repository in the CodeGuru console. In the navigation pane, choose **Reviewer**, and then choose **Repositories**. The **Repositories** page lists all of the associated repositories and their statuses.

You can also use the AWS CLI or the AWS SDK. First call `ListRepositoryAssociations` to find the association ID, then call `DescribeAssociation`.

Where can I check the status of a code review?

You can check the status of a code review in the CodeGuru console. In the navigation pane, choose **Reviewer**, and then choose **Code reviews**. The **Code reviews** page lists all of the recent code reviews and their statuses.

You can also use the AWS CLI or the AWS SDK. If you have the code review Amazon Resource Name (ARN), you can call `DescribeCodeReview`. Alternatively, you can call `ListCodeReviews` and filter using `ProviderType` and `RepositoryName`.

Where can I check the status of a third-party source provider connection?

If you are using a source provider that uses `CodeConnections`, you can check the status of a connection using the AWS CLI or AWS SDK. To do this, call `ListConnections` and filter by the type of source provider, such as `Bitbucket`.

If you can see your connection displayed there with a status of **Available**, you should be able to return to the CodeGuru console and find your connection. Try refreshing the display in the console if you haven't already. Your connection only displays on the CodeGuru console if it has a status of **Available**. The console does not display connections with a status of **Pending** or **Error**.

My repository is in an associated state. Why don't I see recommendations?

This could happen for the following reasons:

- CodeGuru Reviewer doesn't have any recommendations.
- There has not been a pull request or a full repository analysis request, so CodeGuru Reviewer has not had a chance to review.
- There was an issue running CodeGuru Reviewer on the source code. You should [contact Support](#).

Why did my association fail?

An association usually fails because of missing permissions. You can find more information about why the association failed from the status reason.

You can check the status of a repository association in the CodeGuru console.

1. In the navigation pane, choose **Reviewer**, and then choose **Repositories** to navigate to the **Repositories** page. This page lists all the associated repositories and their statuses.
2. Select the association for which you want to see status details.
3. From the **Action** list, choose **View repository details**. A small window opens with information about the repository and the association status.

You can also use the AWS CLI or the AWS SDK. First, call `ListRepositoryAssociations` to find the association ID, then call `DescribeAssociation`.

When you have fixed the problem, retry associating the repository.

Why did my code review fail?

To check the failure status reason of the code review, call the `DescribeCodeReview` API using the AWS CLI or the AWS SDK. You can also find more information about why the code review failed from the status reason on the console. To view details about a code review status on the console, navigate to the **Code reviews** page and choose the name of the code review that failed.

Code reviews usually fail for the following reasons:

- Source code access permissions are revoked, and CodeGuru Reviewer was not able to clone the source code to review. In CodeCommit, this usually happens when the customer removes the `codeguru-reviewer-enabled` repository tag from the repository. The easiest way to fix this is to disassociate the repository and then associate the repository again.
- The pull request being reviewed has been closed, or the branch being reviewed has been deleted, and CodeGuru Reviewer was not able to clone the source code to review before that occurred. Wait for CodeGuru Reviewer to finish reviewing your code before deleting the source branch or closing the pull request.

What if I disagree with the recommendation?

Recommendations depend on context and a variety of other factors. It's possible that some recommendations are not useful. In these cases, reply to the recommendation in the source provider or the CodeGuru Reviewer console to leave feedback on the recommendation.

In CodeCommit, a thumbs-up or thumbs-down icon is provided next to the comments that you can use to respond to comments made by CodeGuru Reviewer. In other repository source providers, you can reply to a comment made by CodeGuru Reviewer, and include a thumbs-up or thumbs-down emoji in your comment to indicate whether it was helpful. You can also go to the **Code reviews** page on the CodeGuru Reviewer console and select the name of a code review to view details and recommendations from that code review. There are thumbs-up and thumbs-down icons there under each recommendation that you can choose to indicate whether the recommendation was helpful.

How do I suppress a recommendation?

You cannot suppress *individual* recommendations, but you can suppress recommendations from CodeGuru Reviewer for specific files or directories in your repository. To do so, add the files or directories to an `aws-codeguru-reviewer.yml` file. For more information, see [Suppress recommendations](#).

The repository status has been associating for more than 5 minutes. What should I do?

If you have refreshed the page and the status has not changed after five minutes, it's possible that there is a problem with the repository source provider. To check the status of the repository, on the **Repositories** page, choose **Action**, then **View repository details**.

The code review status has been Pending for more than 15 minutes. What should I do?

If you have refreshed the page and the status has not changed after 15 minutes, it's possible that there is a problem with the repository association or an internal failure. To check the status reason of the code review, call the `DescribeCodeReview` API using the AWS CLI or the AWS SDK. You can also find more information about why the code review failed from the status reason on the console. To view details about a code review status on the console, navigate to the **Code reviews** page and choose the name of the code review that failed.

How do you access a repository if its owner is no longer available?

If the owner of a repository is no longer able to maintain it, you should make another person an administrator. The new administrator should then disassociate the repository and reassociate it. Having a group or email list with administrator privileges helps avoid this problem.

Can I use the same AWS CodeStar connection to access repositories in two different accounts?

Each connection is associated with one third-party repository source provider account. To access repositories in multiple accounts, create separate connections and switch between the accounts to access corresponding repositories. You can create separate connections for the different accounts from the **Associate repository** page in the console.

I'm trying to connect to my third-party repositories. What is the difference between an app installation and a connection? Which one can be used to adjust permissions?

An *app installation* is a feature that allows CodeConnections to create connections to a single repository source provider account. A *connection* is a feature that uses an app installation through CodeConnections to connect a CodeGuru Reviewer account to a repository source provider account. Multiple connections can be used for the same app installation if different users need to have different levels of permissions.

How do I know if CodeGuru Reviewer used my aws-codeguru-reviewer.yml file in a code review?

CodeGuru Reviewer recognizes the presence of a valid and correctly named `aws-codeguru-reviewer.yml` file in your repository when it creates a full repository analysis code review or an incremental code review.

To confirm that CodeGuru Reviewer used your `aws-codeguru-reviewer.yml` file in a code review

1. In the CodeGuru Reviewer console, choose **Code reviews**. This page lists all code reviews performed.
2. Choose a code review in the list.
 - If CodeGuru Reviewer used your file in the code review, then **Success** appears under **Analysis configuration file**.

amazon-codeguru-reviewer-sample-app-suppression-format-56e9c0e8-ea83-4561-bd88-3dedceebf4ac

Details		
Status  Completed	Analysis configuration file Info  Success	ARN  arn:aws:codeguru-reviewer:us-west-2: :association:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-56e9c0e8-ea83-4561-bd88-3dedceebf4ac
Type Full repository analysis	Repository name amazon-codeguru-reviewer-sample-app	Reviewed lines of code 0
Time created April 28, 2022, 11:49 (UTC-07:00)	Branch name suppression-format	Details CodeGuru Reviewer successfully finished reviewing the source code.
Last updated April 28, 2022, 11:54 (UTC-07:00)	Provider GitHub	Recommendations 10

- If CodeGuru Reviewer found errors in your file, then **Error** appears under **Analysis configuration file** and a message indicating the errors appears at the top of the page.

Also, **Failed** appears under **Status**, indicating that CodeGuru Reviewer did not perform a code review.

Fix your `aws-codeguru-reviewer.yml` file based on the error messages and then initiate a new full repository analysis. For more information, see [Error handling for the aws-codeguru-reviewer.yml file](#).

amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a

Details



The CodeGuru Reviewer analysis failed due to errors with your `aws-codeguru-reviewer.yml` file. Fix the errors and then create a new analysis.
Your analysis configuration file uses an unsupported version number. Learn more about the [criteria that your analysis configuration file must meet](#).

Status Failed	Analysis configuration file Info Error	ARN <code>arn:aws:codeguru-reviewer:us-west-2:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-44e02cd2-ea45-4215-9e62-c9752a14a24a</code>
Type Full repository analysis	Repository name amazon-codeguru-reviewer-sample-app	Reviewed lines of code -
Time created April 28, 2022, 07:19 (UTC-07:00)	Branch name suppression-format	Details Amazon CodeGuru Reviewer was unable to complete the request because the CodeGuru Reviewer configuration file <code>aws-codeguru-reviewer.yml</code> had errors.
Last updated April 28, 2022, 07:26 (UTC-07:00)	Provider GitHub	Recommendations -

- If CodeGuru Reviewer did not recognize your file name or find the file at the root directory of your repository, then **No file detected** appears under **Analysis configuration file**. Your file must be named `aws-codeguru-reviewer.yml` and must exist in the root directory of your repository. Otherwise CodeGuru Reviewer cannot recognize that the file exists, use it in code reviews, or return error messages about problems with the file.

Confirm the name and location of your file, make any needed changes, and then initiate a new code review.

amazon-codeguru-reviewer-sample-app-suppression-format-14144a31-a9d4-45f7-b31e-bbdbfb5d486d

Details

Status Completed	Analysis configuration file Info No file detected	ARN <code>arn:aws:codeguru-reviewer:us-west-2:5414b3ef-e68a-4d5f-8d5c-40f274aa6cf2:code-review:RepositoryAnalysis-amazon-codeguru-reviewer-sample-app-suppression-format-14144a31-a9d4-45f7-b31e-bbdbfb5d486d</code>
Type Full repository analysis	Repository name amazon-codeguru-reviewer-sample-app	Reviewed lines of code 0
Time created April 28, 2022, 12:03 (UTC-07:00)	Branch name suppression-format	Details CodeGuru Reviewer successfully finished reviewing the source code.
Last updated April 28, 2022, 12:09 (UTC-07:00)	Provider GitHub	Recommendations 10

For more information about using an `aws-codeguru-reviewer.yml` file, see [Suppress recommendations](#).

Why didn't my costs decrease when I used an `aws-codeguru-reviewer.yml` file?

Using an `aws-codeguru-reviewer.yml` file to suppress recommendations from CodeGuru Reviewer might decrease the amount of code that CodeGuru Reviewer analyzes and reduce your costs. If your costs don't decrease as expected when using an `aws-codeguru-reviewer.yml` file, then check the following possible reasons.

- CodeGuru Reviewer didn't recognize your `aws-codeguru-reviewer.yml` and couldn't use it to suppress recommendations during a code review. Confirm that you named your file correctly and used the correct extension.
- Your `aws-codeguru-reviewer.yml` file is not excluding as much content as you initially thought. Check the files and directories listed under `excludeFiles:` in your `aws-codeguru-reviewer.yml` file.
- Your repository increased in size, which offset any reduction from the files and directories listed under `excludeFiles:` in your `aws-codeguru-reviewer.yml` file.
- For incremental code reviews, monthly charges are based on the *maximum* number of lines of code reviewed during the month. For example, if your repository includes 150,000 lines of code and you initiate a full repository analysis code review, but later in the month, you add some files or directories to an `aws-codeguru-reviewer.yml` file in your repository and run a new code review of 50,000 lines of code, your monthly bill reflects the cost for reviewing the *larger* number: 150,000 lines of code.

For more information about using an `aws-codeguru-reviewer.yml` file, see [Suppress recommendations](#).

Amazon CodeGuru Reviewer User Guide document history

The following table describes the major updates and new features for the *Amazon CodeGuru Reviewer User Guide*. We also update the documentation frequently to address the feedback that you send us. For notification about updates to this documentation, you can subscribe to an RSS feed.

Latest documentation update: August 17, 2022

Change	Description	Date
Updated topic	Updated content clarifies the use of S3 repositories. For more information, see What languages and repositories can I use with CodeGuru Reviewer?	August 23, 2022
New topic	CodeGuru Reviewer now allows you to suppress recommendations with an analysis configuration file. For more information, see Suppress recommendations from CodeGuru Reviewer .	May 2, 2022
Updated topic	CodeGuru Reviewer now retains forever the OAuth token created after you connect to your GitHub account. For more information, see Captured data and Data retention .	March 2, 2022
Updated topic	CodeGuru Reviewer now references the new Amazon	January 26, 2022

CodeGuru Reviewer Detector Library. The Detector Library lists the detectors that CodeGuru Reviewer uses to analyze your code and provides detailed information about these detectors. For more information, see [Recommendation types](#) and the [Amazon CodeGuru Reviewer Detector Library](#).

[New topic](#)

CodeGuru Reviewer now supports the detection of unprotected secrets in your code. For more information, see [Secrets detection](#).

November 28, 2021

[New topic](#)

CodeGuru Reviewer now supports code reviews with GitHub Actions. For more information, see [Create code reviews with GitHub Actions](#).

June 24, 2021

[New topic](#)

CodeGuru Reviewer now supports encryption of an associated repository using an *AWS KMS key*. For more information, see [Encrypting a repository association in Amazon CodeGuru Reviewer](#).

April 26, 2021

[Updated topics](#)

CodeGuru Reviewer now supports reviews of Python source code.

December 3, 2020

New topic	CodeGuru Reviewer now supports code reviews that include security analysis for Java applications. For more information, see Create code reviews with GitHub Actions .	December 1, 2020
Updated topic	CodeGuru Reviewer now supports analysis that helps you improve the quality of your code.	November 25, 2020
New topic	CodeGuru Reviewer now supports adding tags to associated repositories. For more information, see Tagging a repository associated with a repository .	November 19, 2020
New topic	CodeGuru Reviewer now supports AWS PrivateLink. Use VPC endpoints when calling CodeGuru Reviewer API operations to increase the security. For more information, see CodeGuru Reviewer and interface VPC endpoints (AWS PrivateLink) .	September 11, 2020

[New topic](#)

This user guide now includes information about repository analysis scans. You can now enable CodeGuru Reviewer to provide recommendations on all the code in a branch at any time with repository analysis code reviews. For more information, see [About full repository analysis and incremental code reviews.](#)

August 3, 2020

[General availability release](#)

You can now enable CodeGuru Reviewer to provide recommendations on repositories in GitHub Enterprise, as well as Bitbucket, GitHub, and AWS CodeCommit. More recommendation types are available as well.

June 29, 2020

[New topic](#)

This user guide now includes a tutorial that shows you how to create a repository association with a GitHub repository that has example code. The example code is intentionally suboptimal, so CodeGuru Reviewer generates a recommendation on a pull request you create. For more information, see [Tutorial: monitor source code in a GitHub repository.](#)

June 19, 2020

[New topic](#)

This user guide now includes a security section. Learn about data retention, IAM policies, monitoring your profiling groups with AWS CloudTrail, and more. For more information, see [Security in CodeGuru Reviewer](#).

June 11, 2020

[Preview release](#)

This is the preview release of the *Amazon CodeGuru Reviewer User Guide*.

December 3, 2019

AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.