aws

User Guide

Amazon CodeGuru Security

Amazon CodeGuru Security: User Guide

Table of Contents

Amazon CodeGuru Security is in preview release and is subject to change.

What is Amazon CodeGuru Security?

Amazon CodeGuru Security is a static application security tool that uses machine learning to detect security policy violations and vulnerabilities. It provides suggestions for addressing security risks and generates metrics so you can track the security posture of your applications. CodeGuru Security's policies, which are informed by years of Amazon.com and AWS security best practices, help you to create and deploy secure, high-quality applications.

CodeGuru Security is currently supported in several AWS Regions.

What kind of suggestions does CodeGuru Security provide?

CodeGuru Security identifies security vulnerabilities in your code and suggests remediations to improve the security of your code base. Examples of security vulnerabilities it detects include resource leaks, hardcoded credentials, and cross-site scripting. CodeGuru Security can also identify code quality issues with some integrations. For more information on the types of analysis performed in code scans, see Types of code scans.

CodeGuru Security scans are powered by Amazon CodeGuru detectors that can identify a range of code security and code quality issues. For information about these detectors, see the Amazon CodeGuru Detector Library.

What languages does CodeGuru Security support?

CodeGuru Security supports the following language versions:

- **Java** - Java 17 and earlier
- **JavaScript** - ECMAScript 2021 and earlier
- **Python** - Python 3.11 and earlier, within the Python 3 series
- **C#** - All versions (.Net 6.0 and later recommended)
- **TypeScript** - All versions
- **Ruby** - Ruby 2.7 and 3.2
- **Go** - Go 1.18
- **C** - C11 and earlier

- **C++** - C++17 and earlier
- **PHP** - PHP 8.2 and earlier
- **Infrastructure as Code (IaC) languages**
  - **AWS CloudFormation** - 2010-09-09
  - **Terraform** - 1.6.2 and earlier
  - **AWS CDK** - TypeScript and Python

CodeGuru Security supports the following languages for automatic code fixes:

- **Java** - Java 17 and earlier
- **JavaScript** - ECMAScript 2021 and earlier
- **Python** - Python 3.11 and earlier, within the Python 3 series
- **C#** - All versions (.Net 6.0 and later recommended)
- **TypeScript** - All versions
- **Infrastructure as Code (IaC) languages**
  - **AWS CloudFormation** - 2010-09-09
  - **Terraform** - 1.6.2 and earlier
  - **AWS CDK** - TypeScript and Python

For a list of the file types supported for secrets detection, see ???.

What IDEs does CodeGuru Security support?

CodeGuru Security can be used in the following interactive development environments (IDEs). For notebook IDEs, CodeGuru Security is available through the Amazon CodeGuru extension for code written in Python. For other IDEs, CodeGuru Security is available through the Amazon CodeWhisperer plugin for code written in all languages CodeGuru Security supports.

- Amazon SageMaker Studio
- JupyterLab
- Visual Studio Code through the AWS Toolkit
- IntelliJ IDEA through the AWS Toolkit

What integrations does CodeGuru Security support?

CodeGuru Security supports integration with the following products and services:

- GitHub
- GitLab
- Bitbucket
- AWS CLI
- AWS CodePipeline

CodeGuru Security also supports the following services:

- AWS Lambda code scanning with Amazon Inspector. For more information, see Scanning AWS Lambda functions with Amazon Inspector.
- Amazon CodeWhisperer security scans. For more information, see Security scans in Amazon CodeWhisperer.

How is CodeGuru Security different from CodeGuru Reviewer?

CodeGuru Security is a rearchitected and redesigned version of CodeGuru Reviewer. CodeGuru Security uses hundreds of new security detectors to scan your code, in addition to the detectors that were developed for CodeGuru Reviewer. CodeGuru Security also includes many additional features such as vulnerability tracking and a metrics dashboard to help you monitor the security posture of your applications. For more information, see CodeGuru Security Features. If you are a CodeGuru Reviewer customer and want to access the most updated code scanning capabilities with new detectors, enable code quality analysis in your scans. For more information, see the section called “Types of code scans”.

How much does CodeGuru Security cost?

Currently, CodeGuru Security is in preview release and is free to use.

How do I get started with CodeGuru Security?

Currently CodeGuru Security is available through the console, the AWS CLI and AWS SDKs, and through several integrations. For more information, see Getting started with CodeGuru Security.

How CodeGuru Security works

Amazon CodeGuru Security scans your applications, detects security vulnerabilities, and provides suggestions for how to remediate them in your code. After updating your code, you rerun the scan to make sure the security vulnerability has been remediated and to close the finding. By revising code and re-running scans, you generate security metrics you can track to continuously improve the security posture of your applications.

Scans

Code scans are powered by detectors in the Amazon CodeGuru Detector Library. Each detector corresponds to a type of code defect and can detect a range of issues related to a defect. For more information, visit the Amazon CodeGuru Detector Library.

You can run code scans directly in the CodeGuru Security console, use the AWS CLI, use AWS SDKs, or integrate with another service to detect security issues where you build your applications. For instructions on how to integrate CodeGuru Security with other services, see Getting started with CodeGuru Security.

Findings

When CodeGuru Security detects security vulnerabilities, it returns findings, which include information about security issues in your code, where they are, and suggestions for how to remediate them. If the suggested remediation includes a change to your code, CodeGuru Security highlights the vulnerable lines of code to remove and suggests inline code fixes as replacements. For more information, see Working with findings.

Metrics

The CodeGuru Security Dashboard provides metrics to track the security posture of your application, including open critical findings, the severity distribution of findings, and an assessment of what vulnerabilities are most common. CodeGuru Security tracks the vulnerabilities and trends across multiple revisions of the same code resources using the scan name you provide when a scan is created. For more information, see Understanding dashboard metrics.

You can also choose individual scans in the console to view security data. This includes the number of open and closed findings and an assessment of what vulnerabilities appear in a particular scan. For more information about scans, see Working with code scans.

Features

This section outlines common features of Amazon CodeGuru Security and how they help mitigate security risk in your applications.

High precision vulnerability detection

CodeGuru Security uses machine learning based on years of AWS and Amazon.com security best practices to detect security vulnerabilities in your code with Amazon CodeGuru detectors. These detectors look for code vulnerabilities like injection flaws, leaking data, weak cryptography, or missing encryption. As we update our security policies and add new detectors, code scans automatically incorporate the new policies. Detected vulnerabilities are returned as findings, which include details about the security risk and how to remediate it.

Automatic code fixes

For certain vulnerabilities, CodeGuru Security uses generative AI to create plug-and-play code blocks that can directly replace your vulnerable lines of code. You can download a code patch from the console to apply to your file, or you can remove the vulnerable code and then paste suggested code updates into your file. With Amazon Inspector Lambda code scanning, you can apply code fixes that update your code in-place.

Vulnerability tracking

CodeGuru Security utilizes a machine learning based vulnerability tracking feature which tracks a vulnerability even if it moves to a different location within a file or to another file. After a vulnerability is initially detected, the vulnerability tracking feature can detect if it is still present across subsequent scans, or if it has been remediated. When vulnerability tracking detects that a vulnerability has been remediated, it automatically changes the status of the finding to Closed. This status update is passed to any integrated notification system. No user action is required.

Secrets detection

CodeGuru Security integrates with AWS Secrets Manager to use a secrets detector that finds unprotected secrets in your code and text files, including hardcoded passwords, database connection strings, user names, and more. Secrets detection is automatically enabled in scans, so you don't need to turn it on. For more information, see Secrets detection.

Integrations

In addition to running code scans directly in the console, you can integrate CodeGuru Security with several other products and services. By integrating with your existing workflow, you can automate vulnerability detection without disrupting your software development process. For a list of IDEs and services you can use with CodeGuru Security, see IDEs supported by CodeGuru Security and Integrations supported by CodeGuru Security. For instructions on how to integrate with a service, see Getting started with CodeGuru Security, or go to the Integrations page in the CodeGuru Security console.

Metrics dashboard

CodeGuru Security analyzes findings across your account and generates metrics that are presented in a high-level dashboard. The dashboard displays data about your findings like the average time to close findings, what types of vulnerabilities are present in your scans, and the severity distribution of your findings. With the vulnerability tracking feature, the Metrics dashboard is able to maintain an up-to-date representation of the security posture of your code resources.

You can use these metrics to track the progress of your application security, identify vulnerabilities during software development, and track the lifecycle of vulnerabilities. You can also communicate the status of application security, and collaborate with other teams to address security issues. For information about how these metrics are calculated and where to find them, see Understanding dashboard metrics.

Security posture over time

Findings metrics in the dashboard let you monitor progress toward remediation of findings and view trends over time to see if SLAs for remediation are being met.

Prioritization of security vulnerabilities

Findings are categorized by severity, which lets you focus on the security vulnerabilities that you want address first or where you want to concentrate your remediation efforts.

Code quality scanning

In addition to detecting security vulnerabilities in your code, you can enable code quality scanning to maintain the quality of your codebase. For more information, see Types of code scans.

Continually scan your environment for vulnerabilities

Enable automatic code scanning in your workflow to scan files as you develop your applications and catch security vulnerabilities early in the development process.

No machine learning expertise needed

CodeGuru Security uses machine learning models that AWS manages to detect security vulnerabilities in your code. These models make sure that your code abides by security policies, follows best practices, and takes advantage of AWS code security expertise.

Management of scans and findings in customizable views

In addition to the Dashboard, you can view the Scans page and Findings page in the console for a list of all scans and findings in your account. The Scans page gives an overview of all scans in an account. You can choose individual scans for information about the findings generated by the scan. For more information, see Working with code scans. The Findings page lists findings based on a chosen severity level. You can view individual findings for information about the security vulnerability and suggested remediation. For more information, see Working with findings.

Terminology and metrics

This section provides an overview of the key terminology and metrics in Amazon CodeGuru Security.

Age

The amount of time a finding is open, starting at initial detection.

Analysis type

The type of analysis performed in a scan. You can create scans that only detect security vulnerabilities, or scan for both security and quality defects in your code. For more information, see Types of code scans.

Average time to close

The average amount of time that a finding is open, from initial detection to being closed, during a particular date range.

Closed findings

Previously detected findings that CodeGuru Security no longer identifies as security vulnerabilities during a subsequent scan because the security vulnerabilities were remediated.

Closure rate

The percentage of findings that were closed during a particular date range. This number is determined by dividing the number of open findings during the date range by the number

of closed findings for the same period. For example, if 8 out of 10 open findings were closed during a date range, then the closure rate is 80%.

Detector

A defined rule that CodeGuru Security uses to check your code for security vulnerabilities based on industry standards and AWS best practices. Detectors identify a type security vulnerability and are used to group findings based on these categorizations of vulnerabilities. To learn more, see the Amazon CodeGuru Detector Library.

Finding

A security vulnerability that CodeGuru Security detects during a scan.

Finding ID

A unique identifier for a finding.

Finding summary

The number of findings of each severity level that are open across all scans in an account.

Finding status

Indicates whether a finding is open or closed.

Open findings

Detected security vulnerabilities that have not been remediated and are still open. This number could include new findings from a current scan or findings that are still open from a previous scan.

Relevant CWE

The Common Weakness Enumeration, or set of software vulnerabilities with identification, mitigation, and prevention descriptions that applies to a particular detector. For more information, see Common Weakness Enumeration.

Rule ID

An identifier for the rule that generated the finding.

Scan

An analysis of a code resource by CodeGuru Security for potential security policy violations and vulnerabilities.

Scan name

The unique name that CodeGuru Security uses to track scans across multiple revisions of the same code resource. When you create a unique scan name and use it to re-run scans on updated resources, CodeGuru Security is able to provide accurate metrics for your findings.

Scan status

Indicates whether a scan is in progress, complete, or failed.

Severity

The gravity of findings that CodeGuru Security identifies, divided into critical, high, medium, low, and informational. For more information, see Severity definitions.

Vulnerability tags

Categorizations of findings by type, programming language, or other classification such as maintainability or consistency.

Vulnerability name

The categorization of a vulnerability based on the detector that generated the finding.

How is CodeGuru Security different than other AWS security services?

Amazon CodeGuru Security, which identifies security vulnerabilities in your application resources, adds to the AWS collection of security services.

- Amazon Inspector is a vulnerability management service that continuously scans your AWS workloads for software vulnerabilities and unintended network exposure. You can run scans on your AWS Lambda functions that are powered by CodeGuru Security.
- Amazon GuardDuty monitors network traffic for threat patterns such as unusual data access in Amazon Simple Storage Service or API calls from known malicious IP addresses.
- Amazon Macie scans data storage locations for unencrypted data such as personally identifiable information (PII) and financial data.
- AWS Security Hub collects security data from across AWS accounts, services, and supported third-party products and helps you analyze your security trends and identify the highest priority security issues.

- Amazon CodeGuru Reviewer scans your code repositories for code defects related to quality, maintainability, and security and provides recommendations for how to address them. CodeGuru Security is a rearchitected and redesigned version of CodeGuru Reviewer. CodeGuru Security uses hundreds of new security detectors to scan your code, in addition to the detectors that were developed for CodeGuru Reviewer.

Setting up Amazon CodeGuru Security

This section explains how to set up Amazon CodeGuru Security for the first time. If you haven't already, you sign up for an AWS account. After making an account, you configure IAM permissions to get access to the service. Optionally, you can install and configure the AWS CLI and AWS SDKs.

Topics

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account automatically has access to all AWS services, including Amazon CodeGuru Security. Note that you're charged only for the services that you use.

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create a user with administrative access

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

Secure your AWS account root user

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

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

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

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

Create a user with administrative access

1. Enable IAM Identity Center.

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

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

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

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

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

Configure IAM permissions

Following security best practices, create an AWS Identity and Access Management (IAM) role with access restricted to Amazon CodeGuru Security operations and with required permissions. You can add other permissions as needed.

The following policies provide permissions to use Amazon CodeGuru Security:

- **AmazonCodeGuruSecurityFullAccess**: Provides full access to resources needed to use Amazon CodeGuru Security.
- **AmazonCodeGuruSecurityScanAccess**: Provides access to API operations needed to create scans, get scan information, and get scan findings.

For more information on these AWS managed policies, see AWS managed policies for Amazon CodeGuru Security.

You can also create custom IAM policies to allow permissions for CodeGuru Security actions and resources. See the following topics for more information on configuring IAM roles to use CodeGuru Security:

- Authenticating with identities
- How Amazon CodeGuru Security works with IAM

- Identity-based policy examples for Amazon CodeGuru Security

Assigning permissions

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Creating a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Install and configure the AWS CLI and AWS SDKs

AWS CLI

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

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

To install and configure the AWS CLI

1. To install or upgrade the AWS CLI, follow the instructions in Getting started with the AWS CLI.
2. To configure the AWS CLI, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide* and Managing access keys for IAM users in the *IAM 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 Security endpoints and quotas in the *AWS General Reference*.

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

```
aws codeguru-security help
```

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

AWS SDKs

Download and install the AWS SDKs that you want to use. For more information about the SDKs and how to install the programming languages you want to use with AWS, see Tools to Build on AWS.

Start scanning

After setting up, you can scan your code resources with CodeGuru Security using the console, the AWS CLI, AWS SDKs, and several integrations.

To start scanning in the console, open the Scans page in the CodeGuru Security console at https://console.aws.amazon.com/codeguru/security/Scans. For more information, see Create code scans in the console.

To start scanning with the CLI and SDKs, see Automate scans with the AWS CLI and Create code scans with the AWS CLI and AWS SDKs.

For all other integrations, choose from the list on Getting started with CodeGuru Security or on the Integrations page in the console at https://console.aws.amazon.com/codeguru/security/Integrations.

Getting started with CodeGuru Security

In this section, you learn how to get started with Amazon CodeGuru Security. You can use the CodeGuru Security console, the AWS CLI, AWS SDKs, or integrate with the following products and services.

Before you begin this section, make sure you've followed the steps in Setting up Amazon CodeGuru Security.

Topics

Get started with the console

You can create code scans, view and address findings, and view security metrics in the Amazon CodeGuru Security console.

Open the CodeGuru Security console at https://console.aws.amazon.com/codeguru/security/Scans to get started. You need at least AmazonCodeGuruSecurityScanAccess permissions to create scans and view findings in the console.

For instructions on how to create a code scan in the console, see Create code scans in the console.

Integrate with GitHub, GitHub Enterprise Cloud, or GitHub Enterprise Server

This section shows how to integrate Amazon CodeGuru Security with your GitHub, GitHub Enterprise Cloud, or GitHub Enterprise Server repositories. After you complete the setup, CodeGuru Security will scan your repository whenever you push to the `main` branch, or you can customize the workflow to your organization's needs.

You can view scan findings on the Findings page in the CodeGuru Security console. If you enable code scanning in GitHub, you will be able to see findings in the **Code scanning** page on GitHub.

You can also complete these steps on the **Integrations** page in the CodeGuru Security console. Choose **Integrate with GitHub or GitHub Enterprise** to get started.

Topics

Integrate with GitHub or GitHub Enterprise Cloud

Complete the following steps to integrate CodeGuru Security with GitHub or GitHub Enterprise Cloud.

If you want to view findings in GitHub after CodeGuru Security scans your repository, enable code scanning in GitHub.

- To enable code scanning in GitHub, see Code scanning in the GitHub Docs.
- To enable code scanning in GitHub Enterprise Cloud, see Code scanning in the GitHub Enterprise Cloud Docs.

Step 1: Create an IAM role

To allow CodeGuru Security to integrate with GitHub, create an IAM role with sufficient permissions. You can create an AWS CloudFormation stack that sets up a role for you, or manually configure a role.

To manually configure an IAM role for GitHub, see Configuring OpenID Connect in Amazon Web Services in the GitHub Docs. You can attach the AWS managed policy

AmazonCodeGuruSecurityScanAccess to configure your role with the minimum necessary permissions to integrate with GitHub.

If you have already configured a role to use CodeGuru Security with the GitHub repository you want to scan, you can skip to step 2.

Create a role with a CloudFormation stack

Complete the following steps to create a CloudFormation stack that sets up an IAM role with the necessary permissions attached to integrate with GitHub.

1. Open the **Integrations** page in the CodeGuru Security console and choose **Integrate with GitHub**.
2. For **Step 1: Create an IAM role**, choose **Use CloudFormation template**. Then choose **Open template in CloudFormation** to be redirected to the Create stack page in the CloudFormation console.
3. For **Stack name**, enter a unique name for your stack.
4. For **Parameters**, enter the name of the repository you want to scan.
5. Check the box to acknowledge that AWS CloudFormation might create IAM resources with custom names. This allows CloudFormation to create a role for you.
6. Choose **Create stack**. CloudFormation creates a role called `CodeGuruSecurityGitHubAccessRole`. Continue to the next step.

Step 2: Create a custom workflow in GitHub

Complete the following steps to create a custom workflow for your repository that includes steps and actions to run CodeGuru Security scans. The following workflow will initiate security scans every time you push code to the `main` branch of the repository you are integrating with. If CodeGuru Security detects a critical finding, the pipeline build will fail.

1. Log in to your GitHub account.
2. Open the repository that you want to scan.
3. Choose the **Actions** tab.
4. Choose **New workflow**.
5. Choose **set up a workflow yourself**.

6. Paste the following code into the `.github/workflow/main.yml` file editor in GitHub. You can modify the events defined in this file based on your use case.

   Replace *accountID* with the AWS account ID of the account that is assuming the role and *region* with the region where you are running scans. If you manually configured a role, replace *CodeGuruSecurityGitHubAccessRole* with the name of the role you created to integrate with GitHub.

   If you want to add code quality findings to your scan, add `analysis_type : All` in the `CodeGuru Security` step below `fail_on_severity : Critical`.

```
name: CodeGuru Security Example
on:
  push:
    branches:
      - 'main'

permissions:
  id-token: write
  # for writing security events.
  security-events: write
  # only required for workflows in private repositories
  actions: read
  contents: read

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Respository
        uses: actions/checkout@v3
        with:
          fetch-depth: 0

      - name: Configure aws credentials
        uses: aws-actions/configure-aws-credentials@v2
        with:
          role-to-assume:
 arn:aws:iam::accountID:role/CodeGuruSecurityGitHubAccessRole
          aws-region: region
          role-session-name: GitHubActionScript
```

```
      - name: CodeGuru Security
        uses: aws-actions/codeguru-security@v1
        with:
          source_path: .
          aws_region: region
          fail_on_severity: Critical
      - name: Print findings
        run: |
          ls -l
          cat codeguru-security-results.sarif.json

      # If you want content in security scanning, you'll need to enable
 codescanning by going into github.
      # https://docs.github.com/en/code-security/code-scanning/automatically-
scanning-your-code-for-vulnerabilities-and-errors/configuring-code-scanning-for-a-
repository
      - name: Upload result
        uses: github/codeql-action/upload-sarif@v2
        with:
          sarif_file: codeguru-security-results.sarif.json
```

7. Commit your changes.

Step 3: Run scans and address findings

After creating the workflow, CodeGuru Security will scan your repository based on the events that you have defined in the workflow file. If you used the code from the previous step or otherwise configured your workflow to initiate scans on code commits, CodeGuru Security will automatically scan your code whenever you push to the specified branch.

If you enabled code scanning in GitHub, you can view findings by going to the **Security** tab of your repository, and then choosing **Code scanning** in the left navigation bar. You can also view scans and findings in the CodeGuru Security console.

To address findings, update your code based on the suggested remediations, and then push your changes to the branch where you created the workflow. CodeGuru Security will scan the updated code based on the events that you have defined in the workflow file, and you can check that the vulnerabilities were remediated.

Integrate with GitHub Enterprise Server

Complete the following steps to integrate CodeGuru Security with GitHub Enterprise Server.

Step 1: Prerequisites

Complete the following prerequisites before continuing with the integration steps.

- Enable GitHub Actions on your enterprise server. For more information, see Enabling GitHub Actions for GitHub Enterprise Server in the GitHub Enterprise Server Docs.
- Configure self-hosted runners to run workflows. For more information, see Getting started with self-hosted runners for your enterprise in the GitHub Enterprise Server Docs.
- Grant your enterprise server access to the following AWS GitHub actions: aws-actions/configure-aws-credentials and aws-actions/codeguru-security. For more information, see Managing access to actions from GitHub.com in the GitHub Enterprise Server Docs.
- (Optional) To view findings in GitHub, enable code scanning for your enterprise instance in order to upload static analysis results. For more information, see Code scanning in the GitHub Enterprise Server Docs.

Step 2: Create an IAM role

To allow CodeGuru Security to integrate with GitHub, create an IAM role with sufficient permissions. You can create an AWS CloudFormation stack that sets up a role for you, or manually configure a role.

To manually configure an IAM role for GitHub, see Configuring OpenID Connect in Amazon Web Services in the GitHub Docs. You can attach the AWS managed policy AmazonCodeGuruSecurityScanAccess to configure your role with the minimum necessary permissions to integrate with GitHub.

If you have already configured a role to use CodeGuru Security with the GitHub repository you want to scan, you can skip to step 2.

Create a role with a CloudFormation stack

Complete the following steps to create a CloudFormation stack that sets up an IAM role with the necessary permissions attached to integrate with GitHub.

1. Open the **Integrations** page in the CodeGuru Security console and choose **Integrate with GitHub**.

2. For **Step 1: Create an IAM role**, choose **Use CloudFormation template**. Then choose **Open template in CloudFormation** to be redirected to the Create stack page in the CloudFormation console.
3. For **Stack name**, enter a unique name for your stack.
4. For **Parameters**, for **Hostname**, enter the hostname of your GitHub Enterprise Server instance. For **Repository**, enter the name of the repository you want to scan.
5. Check the box to acknowledge that AWS CloudFormation might create IAM resources with custom names. This allows CloudFormation to create a role for you.
6. Choose **Create stack**. CloudFormation creates a role called `CodeGuruSecurityGitHubEnterpriseServerAccessRole`. Continue to the next step.

Step 3: Create a custom workflow in GitHub

Complete the following steps to create a custom workflow for your repository that includes steps and actions to run CodeGuru Security scans. The following workflow will initiate security scans every time you push code to the `main` branch of the repository you are integrating with. If CodeGuru Security detects a critical finding, the pipeline build will fail.

1. Log in to your GitHub account.
2. Open the repository that you want to scan.
3. Choose the **Actions** tab.
4. Choose **New workflow**.
5. Choose **set up a workflow yourself**.
6. Paste the following code into the `.github/workflow/main.yml` file editor in GitHub. You can modify the events defined in this file based on your use case.

   Replace *`accountID`* with the AWS account ID of the account that is assuming the role and *`region`* with the region where you are running scans. If you manually configured a role, replace *`CodeGuruSecurityGitHubEnterpriseServerAccessRole`* with the name of the role you created to integrate with GitHub.

   If you want to add code quality findings to your scan, add `analysis_type : All` in the `CodeGuru Security` step below `fail_on_severity : Critical`.

```
name: CodeGuru Security Example
on:
```

```
  push:
    branches:
      - 'main'

permissions:
  id-token: write
  # for writing security events
  security-events: write
  # only required for workflows in private repositories
  actions: read
  contents: read

jobs:
  build:
    runs-on: self-hosted
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v3
        with:
          fetch-depth: 0

      - name: Configure aws credentials
        uses: aws-actions/configure-aws-credentials@v2
        with:
          role-to-assume:
 arn:aws:iam::accountID:role/CodeGuruSecurityGitHubEnterpriseServerAccessRole
          aws-region: region
          role-session-name: GitHubActionScript

      - name: CodeGuru Security
        uses: aws-actions/codeguru-security@v1
        with:
          source_path: .
          aws_region: region
          fail_on_severity: Critical
      - name: Print Findings
        run: |
          ls -l
          cat codeguru-security-results.sarif.json

      # Note: Code scanning feature needs to be enabled for this repository.
      # https://docs.github.com/en/enterprise-server@3.10/code-security/code-
scanning/automatically-scanning-your-code-for-vulnerabilities-and-errors/
configuring-default-setup-for-code-scanning
```

```
      - name: Upload result
        uses: github/codeql-action/upload-sarif@v2
        with:
          sarif_file: codeguru-security-results.sarif.json
```

7. Commit your changes.

Step 4: Run scans and address findings

After creating the workflow, CodeGuru Security will scan your repository based on the events that you have defined in the workflow file. If you used the code from the previous step or otherwise configured your workflow to initiate scans on code commits, CodeGuru Security will automatically scan your code whenever you push to the specified branch.

If you enabled code scanning in GitHub, you can view findings by going to the **Security** tab of your repository, and then choosing **Code scanning** in the left navigation bar. You can also view scans and findings in the CodeGuru Security console.

To address findings, update your code based on the suggested remediations, and then push your changes to the branch where you created the workflow. CodeGuru Security will scan the updated code based on the events that you have defined in the workflow file, and you can check that the vulnerabilities were remediated.

Integrate with Bitbucket

The following steps show how to integrate Amazon CodeGuru Security into your Bitbucket pipeline. After you complete the setup, CodeGuru Security will scan your repository whenever you push to the `main` branch, or you can customize the workflow to your organization's needs. After a scan completes, you will be able to see findings on the Findings page in the CodeGuru Security console.

You can also complete these steps on the **Integrations** page in the CodeGuru Security console. Choose **Integrate with Bitbucket** to get started.

Step 1: Create an IAM role

To allow CodeGuru Security to integrate with Bitbucket, create an IAM role with sufficient permissions. You can create an AWS CloudFormation stack that sets up a role for you, or manually configure a role.

To manually configure an IAM role for Bitbucket, see Deploy on AWS using Bitbucket Pipelines OpenID Connect in the Bitbucket Support documentation. You can attach the AWS managed policy AmazonCodeGuruSecurityScanAccess to configure your role with the minimum necessary permissions to integrate with Bitbucket.

If you have already configured a role to use CodeGuru Security with the Bitbucket repository you want to scan, you can skip to step 2.

Create a role with a CloudFormation stack

Complete the following steps to create a CloudFormation stack that sets up an IAM role with the necessary permissions attached to integrate with Bitbucket.

1. Open the **Integrations** page in the CodeGuru Security console and choose **Integrate with Bitbucket**.
2. For **Step 1: Create an IAM role**, choose **Use CloudFormation template**. Then choose **Open template in CloudFormation** to be redirected to the Create stack page in the CloudFormation console.
3. For **Stack name**, enter a unique name for your stack.
4. For **Parameters**, for **Audience**, enter the Audience of the repository you want to scan. For **ProviderUrl**, enter the Identity provider URL of the repository you want to scan.

   These values can be found in your Bitbucket account under **Repository settings**. Go to **Pipelines: OpenID Connect** and then **Identity provider**.
5. Check the box to acknowledge that AWS CloudFormation might create IAM resources with custom names. This allows CloudFormation to create a role for you.
6. Choose **Create stack**. CloudFormation creates a role called `CodeGuruSecurityBitbucketAccessRole`. Continue to the next step.

Step 2: Configure Bitbucket pipelines

Complete the following steps to update your Bitbucket pipeline to include steps and actions to run CodeGuru Security scans. The following pipeline will initiate security scans every time you push code to the `main` branch of the repository you are integrating with. If CodeGuru Security detects a critical finding, the pipeline build will fail.

1. Log in to your Bitbucket account.

2. Open the repository that you want to scan.
3. Choose the **Source** tab.
4. If you don't have a pipeline YAML file yet, choose **Add file** and name it `bitbucket-pipelines.yml`.

   If you have already set up a pipeline YAML file, choose **Edit**.
5. Paste the following code into the `bitbucket-pipelines.yml` file editor in Bitbucket. You can modify the events defined in this file based on your use case.

   Replace *accountID* with the AWS account ID of the account that is assuming the role and *region* with the region where you are running scans. If you manually configured a role, replace *CodeGuruSecurityBitbucketAccessRole* with the name of the role you created to integrate with Bitbucket.

   If you want to add code quality findings to your scan, add `--analysis_type All` to the python script line after `--fail_on_severity Critical`.

```
pipelines:
  branches:
    master:
    - step:
        image: public.ecr.aws/l6c8c5q3/codegurusecurity-actions-public:latest
        oidc: true
        script:
          - export
 AWS_ROLE_ARN=arn:aws:iam::accountID:role/CodeGuruSecurityBitbucketAccessRole
          - export AWS_WEB_IDENTITY_TOKEN_FILE=$(pwd)/web-identity-token
          - echo $BITBUCKET_STEP_OIDC_TOKEN > $(pwd)/web-identity-token
          - python /usr/app/codeguru/command.py --source_path . --aws_region region
 --scan_name CGS-Bitbucket-$BITBUCKET_REPO_SLUG --fail_on_severity Critical
          - cat codeguru-security-results.sarif.json
```

6. Choose **Commit** to commit your changes.

Step 3: Run scans and address findings

After updating the pipeline, CodeGuru Security will scan your code based on the events that you have defined in the YAML file. If you configured your pipeline to initiate scans on code commits, CodeGuru Security will automatically scan your code whenever you push to the specified branch.

You can view your findings in the CodeGuru Security console. To address findings, update your code based on the suggested remediations, and then push your changes. CodeGuru Security will scan the updated code based on the events that you have defined in the YAML file, and you can check that the vulnerabilities were remediated.

Integrate with GitLab

The following steps show how to integrate Amazon CodeGuru Security into your GitLab CI/CD workflow. After you complete the setup, CodeGuru Security will scan your repository whenever you push to the `main` branch, or you can customize the workflow to your organization's needs. After a scan completes, you will be able to see findings in the **Vulnerability report** page on GitLab and on the Findings page in the CodeGuru Security console.

You can also complete these steps on the **Integrations** page in the CodeGuru Security console. Choose **Integrate with GitLab** to get started.

Step 1: Create an IAM role

To allow CodeGuru Security to integrate with GitLab, create an IAM role with sufficient permissions. You can create an AWS CloudFormation stack that sets up a role for you, or manually configure a role.

To manually configure an IAM role for GitLab, see Configure OpenID Connect in AWS to retrieve temporary credentials in the GitLab Docs. You can attach the AWS managed policy AmazonCodeGuruSecurityScanAccess to configure your role with the minimum necessary permissions to integrate with GitLab.

If you have already configured a role to use CodeGuru Security with the GitLab repository you want to scan, you can skip to step 2.

Create a role with a CloudFormation stack

Complete the following steps to create a CloudFormation stack that sets up an IAM role with the necessary permissions attached to integrate with GitLab.

1. Open the **Integrations** page in the CodeGuru Security console and choose **Integrate with GitLab**.
2. For **Step 1: Create an IAM role**, choose **Use CloudFormation template**. Then choose **Open template in CloudFormation** to be redirected to the Create stack page in the CloudFormation console.

3. For **Stack name**, enter a unique name for your stack.
4. For **Parameters**, for **Group**, enter the name of the group that contains your project. For **Project**, enter the name of the project you want to scan.
5. Check the box to acknowledge that AWS CloudFormation might create IAM resources with custom names. This allows CloudFormation to create a role for you.
6. Choose **Create stack**. CloudFormation creates a role called `CodeGuruSecurityGitLabAccessRole`. Continue to the next step.

Step 2: Configure your CI/CD workflow

Complete the following steps to configure your GitLab CI/CD workflow and to define the jobs that make up your pipeline to run CodeGuru Security scans. This pipeline will initiate security scans every time you push code to the `main` branch of the repository you are integrating with. If CodeGuru Security detects a critical finding, the pipeline build will fail.

1. Log in to your GitLab account.
2. Open the project that you want to scan.
3. Choose the **Set up CI/CD**.
4. Choose **Configure pipeline**.
5. Paste the following code into the `.gitlab-ci.yml` file editor in GitLab. You can modify the events defined in this file based on your use case.

   Replace *`accountID`* with the AWS account ID of the account that is assuming the role and *`region`* with the region where you are running scans. Replace *`CodeGuruSecurityGitLabAccessRole`* with the name of the role you created to integrate with GitLab.

   If you want to add code quality findings to your scan, add `--analysis_type All` to the python script line after `--fail_on_severity Critical`.

   ```
   codeguru_security_example:
     image:
       name: public.ecr.aws/l6c8c5q3/codegurusecurity-actions-public:latest
       entrypoint: [""]
     variables:
       ROLE_ARN: arn:aws:iam::accountID:role/CodeGuruSecurityGitLabAccessRole
   ```

```
    AWS_PROFILE: oidc # used to get the credential. More detail: https://
gitlab.com/guided-explorations/aws/configure-openid-connect-in-aws/-/tree/main
  id_tokens:
    MY_OIDC_TOKEN:
      aud: https://gitlab.com
  before_script:
    - mkdir -p ~/.aws
    - echo "${MY_OIDC_TOKEN}" > /tmp/web_identity_token
    - echo -e "[profile oidc]\nrole_arn=${ROLE_ARN}\nweb_identity_token_file=/tmp/
web_identity_token" > ~/.aws/config
  script:
    - REPO_NAME="`basename -s .git $(echo $CI_REPOSITORY_URL | grep -oE "[^/]+$")`"
    - python /usr/app/codeguru/command.py --source_path "." --aws_region "region"
 --scan_name CGS-GitLab-$REPO_NAME --fail_on_severity Critical --output_file_format
 "sast"
    - cat codeguru-security-results.sast.json
  rules:
    - if: $CI_PIPELINE_SOURCE == "push" && $CI_COMMIT_BRANCH == "main"
      when: always
  artifacts:
    reports:
      sast: codeguru-security-results.sast.json
```

6. Choose **Commit changes** to commit your changes.

Step 3: Run scans and address findings

After configuring the CI/CD workflow, CodeGuru Security scan your code based on the events that you have defined in the file. If you configured your pipeline to initiate scans on code commits, CodeGuru Security will automatically scan your code whenever you push to the specified branch.

To view your findings, choose **Secure** in the left navigation bar of your project, and then choose **Vulnerability report**. You can also view code scans and findings in the CodeGuru Security console.

To address findings, update your code based on the suggested remediations, and then push your changes to the branch where you configured the workflow. CodeGuru Security will scan the updated code based on the events that you have defined in the workflow file, and you can check that the vulnerabilities were remediated.

Integrate with AWS CodePipeline

The following steps show how to set up AWS CodePipeline with Amazon CodeGuru Security. After you set up, code scans are automated and you can view findings on the Findings page in the CodeGuru Security console.

You can also complete these steps on the **Integrations** page in the CodeGuru Security console. Choose **Integrate with AWS CodePipeline** to get started.

Step 1: Create CodeBuild project

Complete the following steps to create an AWS CloudFormation stack that sets up a CodeGuru Security CodeBuild project. This authorizes CodeGuru Security to discover your repositories and run security scans whenever you create a pull request.

1. Open the **Integrations** page in the CodeGuru Security console and choose **Integrate with AWS CodePipeline**.
2. For **Step 1: Create an IAM role**, choose **Open template in CloudFormation** to be redirected to the Create stack page in the CloudFormation console.
3. For **Stack name**, enter a unique name for your stack.
4. Check the box to acknowledge that AWS CloudFormation might create IAM resources with custom names. This allows CloudFormation to create a CodeGuru Security CodeBuild project.
5. Choose **Create stack**. Continue to the next step.

Step 2: Add step to CodePipeline

Complete the following steps to add CodeGuru Security as a step in your CodePipeline.

1. Open the AWS CodePipeline console.
2. Choose the pipeline you want to scan.
3. Choose **Edit**.
4. Choose **Add stage** and enter a stage name.
5. For the stage you just created, choose **Add action group**.
6. For Action provider, choose **CodeBuild**.
7. For Input artifacts, choose **SourceArtifact**.

8. For Project name, choose **CodeGuruSecurity**.
9. Choose **Done**.
10. Choose **Save**.

Step 3: Run scans and address findings

After you add CodeGuru Security to your CodePipeline pipeline, CodeGuru Security will run scans on every pipeline deployment. You can view scans and findings in the CodeGuru Security console.

To address findings, update your code based on the suggested remediation, and then push your changes to the pipeline where you added CodeGuru Security as a step. CodeGuru Security will automatically scan the updated code and you can check that the vulnerabilities were remediated.

Automate scans with the AWS CLI

The following steps show you how to automate code scanning in the AWS CLI with Amazon CodeGuru Security. The bash script you download from the console uploads your code resources, creates a scan, and outputs findings to a file with a single command. For information on manually creating and configuring code scans with the AWS CLI, see Create code scans with the AWS CLI and AWS SDKs.

Integrate with the AWS CLI

1. Go to the **Integrations** page in the CodeGuru Security console.
2. On the AWS CLI panel, choose **Integrate with the AWS CLI**.
3. Follow the instructions on the page. If you haven't already, install the AWS CLI and `jq` in order to run the script. See Get started with the AWS CLI and Download jq for instructions.
4. Download the `run_codeguru_security.sh` file from the console.
5. To automatically upload a code resource and scan it, open a command prompt window and run the following command. Replace *`scanName`* with the name of the scan, *`uploadFolder`* with the name of the folder where your code resource is stored, and *`region`* with the AWS Region you want to run scans in.

   ```
   ./run_codeguru_security.sh scanName uploadFolder region
   ```

6. After you've scanned your resource, your findings are written to an output file. You can also view findings with the `GetFindings` API or on the Findings page in the console.

To address findings, update your code based on the suggested remediation and re-run the command from Step 5 with the same scan name and the name of the folder that contains your updated code.

Integrate with IDE plugins

You can scan code in your IDE with Amazon Q security scans. To set up Amazon Q in your IDE, see Install Amazon Q in the *Amazon Q Developer User Guide*.

To learn how to scan your code with Amazon Q, see Security scans in the *Amazon Q Developer User Guide*. After scanning your code, you can view findings in the **Problems** tab in VS Code or the **Amazon Q Security Issues** tab in JetBrains.

To view information about the finding and how to remediate it, hold your cursor over the underlined code. To address findings, update your code based on the suggested remediation and then run another scan to check that the vulnerabilities were remediated. For some vulnerabilities, you can apply code fixes that update your code in-place.

Integrate with Amazon Inspector

Amazon CodeGuru Security is available through Amazon Inspector Lambda code scanning. For more information, see Scanning AWS Lambda functions with Amazon Inspector.

> **Note**
>
> CodeGuru Security only reports critical and high severity vulnerabilities in Lambda code scans with Amazon Inspector. Medium, low, and informational code quality findings are not returned.

The following steps show how to activate AWS Lambda code scanning with Amazon Inspector. After you activate code scanning, code scans are automated and you can view findings in the **All findings** section in the Amazon Inspector console.

You can also complete these steps on the **Integrations** page in the CodeGuru Security console. Choose **Integrate with Amazon Inspector** to get started.

Activate code scanning for Lambda

1. Open the Amazon Inspector console.
2. In the navigation bar, choose **Account management**.
3. Select the accounts that you want to activate Lambda code scanning in.
4. Choose **Activate**, then choose **AWS Lambda code scanning**.

Tutorial: Run scans with SageMaker Studio and JupyterLab

The Amazon CodeGuru Security extension scans your Python and notebook files and provides security recommendations and quality improvements to your code.

After running a scan, detected vulnerabilities or quality issues in your code are underlined. Each underlined section corresponds to a finding that details the issue and suggested remediation. You can view all findings in the diagnostic panel. Once you update your code, you can re-run a scan to see if the finding has been remediated.

The following instructions show you how to install and use the CodeGuru Security extension in JupyterLab and Amazon SageMaker Studio. Before you begin installation, make sure you've followed the steps in Setting up Amazon CodeGuru Security.

Step 1: Install the CodeGuru Security extension

You can install the CodeGuru Security extension in one of two ways, via the command line or in the extension manager.

You can find more information on installing JupyterLab extensions in the JupyterLab Extensions documentation.

> **Note**
>
> If you're using SageMaker Studio, make sure to run `conda activate studio` and `conda deactivate` before and after running the following commands.
> If you're using JupyterLab, make sure to run the commands in the same environment where JupyterLab is installed.
> If you installed JupyterLab with the `conda` environment, activate the environment where JupyterLab is installed before running the following commands.

Install with the command line (recommended)

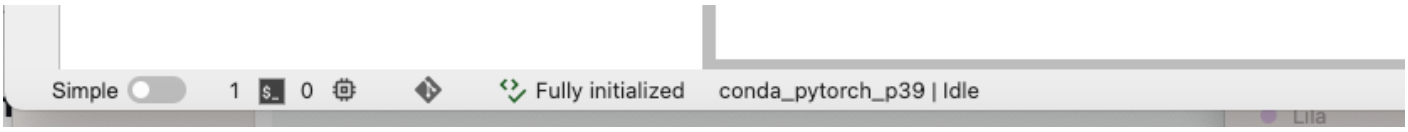
JupyterLab

1. Open a command prompt window and run the following command to install the extension.

```
pip install amazon-codeguru-jupyterlab-extension
```

2. Restart your JupyterLab server.
3. In your browser, refresh the page to view the extension in JupyterLab.

   You can verify that the extension is installed if the LSP server displays **Fully initialized** on the bottom left corner. The following image shows the LSP server with the **Fully initialized** status.



SageMaker Studio

1. Open a command prompt window.
2. Run the following commands to install the extension in the conda environment:

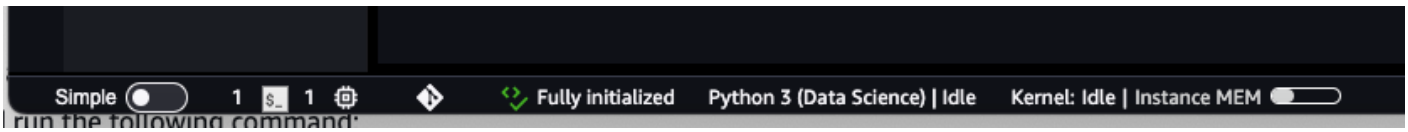
```
conda activate studio
  pip install amazon-codeguru-jupyterlab-extension
  conda deactivate
```

3. Restart your SageMaker Studio server by running the following command:

```
restart-jupyter-server
```

4. In your browser, refresh the page to view the extension in SageMaker Studio.

   You can verify that the extension is installed if the LSP server displays **Fully initialized** on the bottom left corner. The following image shows the LSP server with the **Fully initialized** status.



If you still don't see the extension, try creating a new notebook instance with your application code, and then install the extension.

Install with the extension manager

1. Open SageMaker Studio or JupyterLab.
2. In the left navigation bar, choose the Extension Manager icon.
3. Search @aws/amazon-codeguru-extension.
4. Locate the extension called **@aws/amazon-codeguru-extension** and choose **Install**.
5. A pop-up appears with the title **Server Companion**. Choose **OK**.
6. After a few moments, the following message appears in the Extension Manager:

   "A build is needed to include the latest changes."

   Choose **Rebuild**.
7. After the rebuild is complete, a pop-up appears. Choose **Save and Reload**.
8. Open a command prompt window and run the following command:

   ```
   pip install amazon-codeguru-jupyterlab-extension
   ```
9. Restart your JupyterLab or SageMaker Studio server.
10. Refresh your browser to view the extension.

    You can verify that the extension is installed if the LSP server displays **Fully initialized** on the bottom left corner.

Step 2: Update IAM permissions

To use the extension, a role or user must have the necessary permissions. Follow these steps to update permissions policies with IAM. If you're using the extension in JupyterLab, you must also refresh your AWS account credentials.

1. Update the permissions policy for each role or user who is using the extension. We recommended that you use the AWS managed policy AmazonCodeGuruSecurityScanAccess. For more information on creating policies, see Managed policies and inline policies.

Go to the AWS IAM Console and attach the managed policy to your roles or users.

If you're using SageMaker Studio, attach the policy to the `AmazonSageMaker-ExecutionRole`.

Alternatively, create a new policy with the following permissions.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AmazonCodeGuruSecurityScanAccess",
   "Effect": "Allow",
   "Action": [
    "codeguru-security:CreateScan",
    "codeguru-security:CreateUploadUrl",
    "codeguru-security:GetScan",
    "codeguru-security:GetFindings"
   ],
   "Resource": "arn:aws:codeguru-security:*:*:scans/*"
  }
 ]
}
```

2. If you're using SageMaker Studio, you can skip this step. If you're using JupyterLab, refresh your AWS account credentials via the command line by running the following command:

```
aws configure
```

Step 3: Run a scan

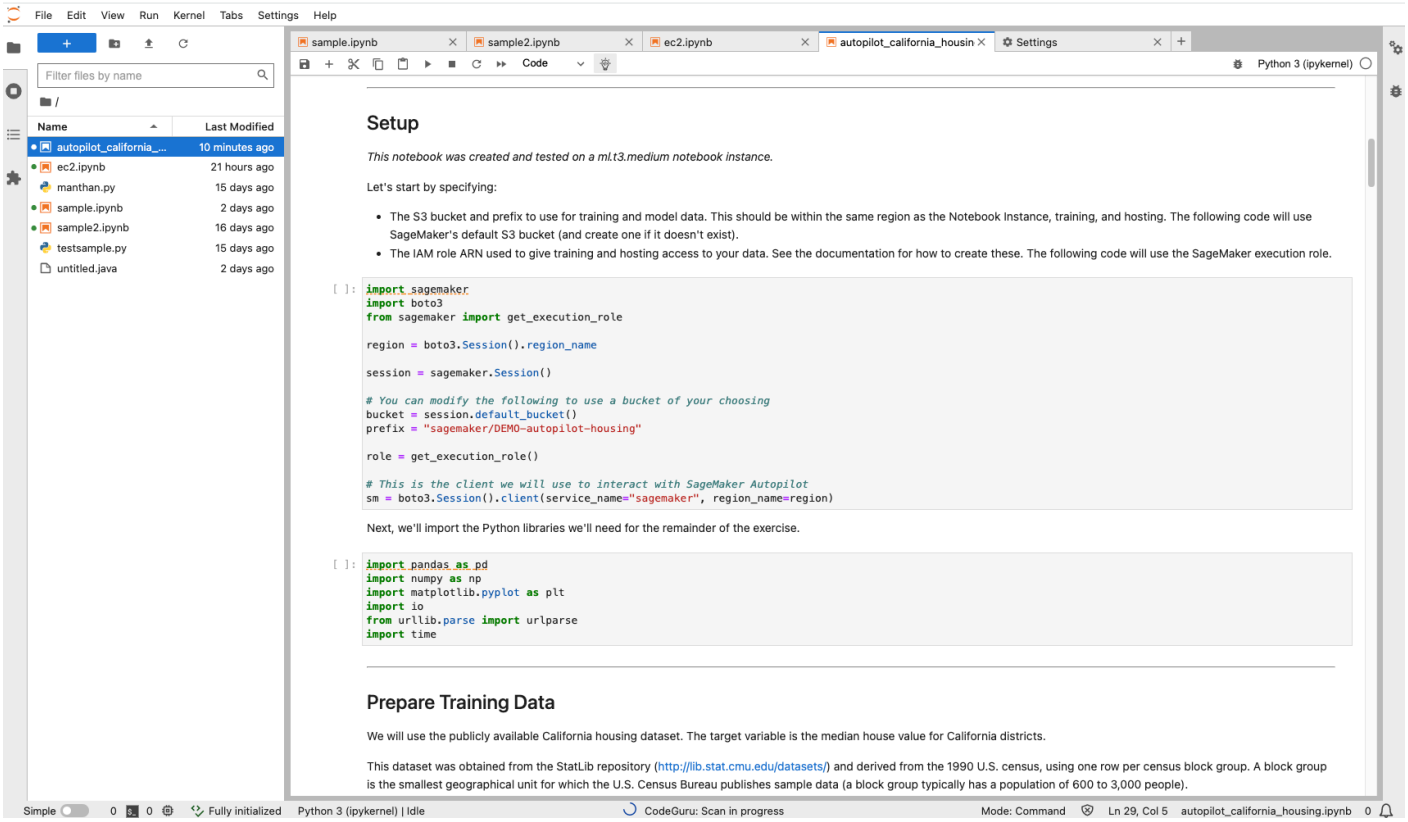
Once you've installed the extension and updated the permissions policy, you are ready to run a scan in JupyterLab or SageMaker Studio.

1. Open the file you want to run a CodeGuru Security scan on in your JupyterLab or SageMaker Studio notebook instance.
2. If the LSP server displays **Fully initialized** on the bottom left corner, the extension is installed, and you are ready to run a scan.

> **Note**
>
> If you see **Server extension missing**, restart your SageMaker Studio or JupyterLab server.

3. You can initiate a scan in one of the following ways:
   - Choose any code cell in your file, and then choose the light bulb icon in the top task bar.
   - Open the context (right-click) menu on any code cell in your file, and then choose **Run code scan**.
4. Once a scan is running, **CodeGuru: Scan in progress** will appear on the bottom panel of the page. The scan might take several seconds to complete. Once complete, the bottom panel displays **CodeGuru: Scan completed** and the findings are underlined in your code.

   The following image shows an in-progress scan.

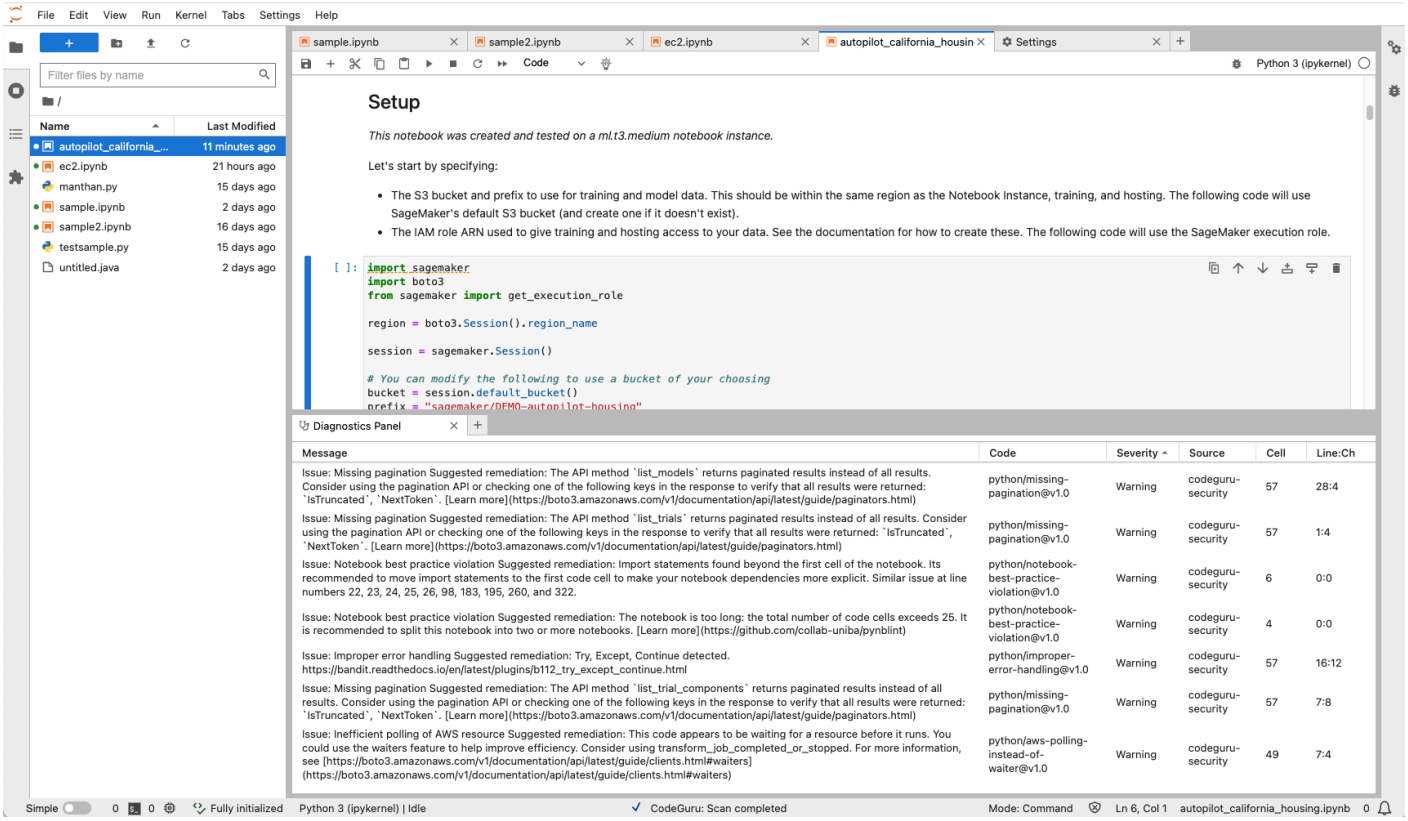


Step 4: View and address findings

Once a scan is complete, you see findings underlined in your code.

1. To view more information on the findings, open the context (right-click) menu for any cell and choose **Show diagnostics panel**. A panel with information about the findings and recommendations appears at the bottom of the notebook file.

   The following image shows a completed scan with the diagnostics panel open to view findings.



   To view a popover with a summary of the finding, hold your cursor over the underlined code.
2. In the diagnostics panel, choose a finding to redirect your cursor to the corresponding lines of code.
3. After you update your code based on the recommendations, you can re-run the scan to see if the issue has been addressed.

   Once you change your code, the scan findings disappear. You must re-run the scan to see them again.

Step 5: Updating scan settings

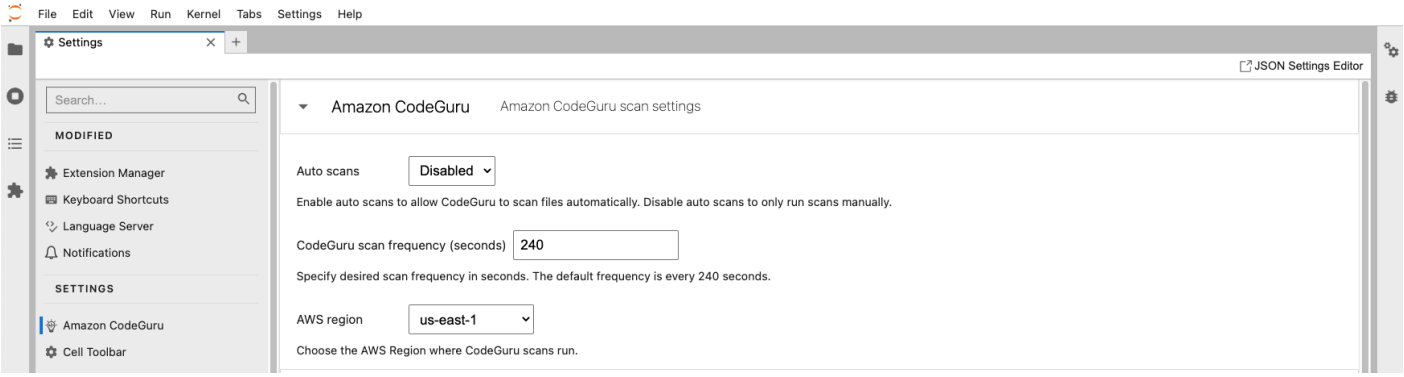
You can specify the frequency of scan runs and the Region where you run scans.

1. Choose **Settings** in the top navigation bar.

2. Choose **Advanced Settings Editor**.
3. In the left navigation bar, choose **CodeGuru Security**.
4. Automatic code scans are disabled by default. If you want scans to run automatically, choose **Enabled** in the dropdown menu next to **Auto scans**.

   If enabled, automatic scans run every 240 seconds by default. If you want to change the frequency of automatic scans, specify a value for **CodeGuru scan frequency**.

   The following image shows the CodeGuru Security scan settings tab with Auto scans disabled.



5. To specify what AWS Region your scans are run in, choose a Region in the dropdown menu next to **Region**.

   You can change the AWS Region where you run scans to keep data in a specific Region while scanning, or to be billed in a specific Region.

Step 6: Disable or uninstall the extension

Disabling the extension prevents you from running scans until it is re-enabled. If you uninstall the extension, you must repeat the installation process to reinstall it.

> **Note**
>
> If you're using SageMaker Studio, be sure to run `conda activate studio` and `conda deactivate` before and after running the following commands.
> If you're using JupyterLab, make sure to run the commands in the same environment where JupyterLab is installed.
> If you installed JupyterLab with the `conda` environment, activate the environment where JupyterLab is installed before running the following commands.

Disable the extension

Open a command prompt window and run the following command.

```
jupyter labextension disable @aws/amazon-codeguru-extension
```

Uninstall the extension

Uninstall with the command line

1. Open a command prompt window and run the following command.

   ```
   pip uninstall amazon-codeguru-jupyterlab-extension
   ```

2. You might also want to remove dependent packages by running the following commands:

   ```
   pip uninstall jupyterlab-lsp
   pip uninstall python-lsp-server
   ```

Uninstall with the extension manager

1. Open a command prompt window and run the following command.

   ```
   pip uninstall amazon-codeguru-jupyterlab-extension
   ```

2. In the Extension Manager, locate the @aws/amazon-codeguru-extension extension and choose **Uninstall**.
3. The following message appears in the Extension Manager:

   "A build is needed to include the latest changes."

   Choose **Rebuild**.
4. After the rebuild is complete, a pop-up appears. Choose **Save and Reload**.
5. You might also want to remove dependent packages by running the following commands:

   ```
   pip uninstall jupyterlab-lsp
   pip uninstall python-lsp-server
   ```

Working with code scans

In CodeGuru Security, a scan is an analysis of a code resource for potential security policy violations and vulnerabilities. When you create a code scan, CodeGuru Security analyzes your code and generates findings with information about security vulnerabilities and how to remediate them.

Before scanning your code, CodeGuru Security filters out unsupported code languages, test code, and any open source or third-party code that is present in your code resource. This ensures that findings are only generated on code that is relevant and that you own. The amount of code that can be scanned per scan is limited, and varies by programming language. For information on code scan limits, see *Quotas*.

You can monitor the security posture of your code over time by choosing a scan in the console and viewing the data in the Metrics panel. For more information, see View code scan details.

You can create code scans in the CodeGuru Security console, with the AWS CLI and AWS SDKs, or through integrations with CodeGuru Security. Before you begin scanning, make sure you've completed the steps in Setting up Amazon CodeGuru Security and Getting started with CodeGuru Security.

This section covers creating, configuring, viewing, and understanding code scans.

Topics

Types of code scans

Amazon CodeGuru Security can perform code security analysis and code quality analysis in code scans. All code scans perform code security analysis, where CodeGuru Security scans your code and returns findings about detected security vulnerabilities and hardcoded secrets. You can also configure your scans to include code quality analysis, which returns findings related to the quality of your code in addition to security vulnerabilities.

Whereas security findings are used to generate finding and vulnerability resolution metrics for your account, code quality findings do not affect the metrics in your dashboard data. Rather, they are labeled as Informational findings that you can choose to address and will not affect how the security posture of your application is assessed.

This section covers types of code analysis and how to enable them in your scans.

Topics

Code security analysis

Code security analysis detects potential security policy violations and vulnerabilities in your code. Code security analysis is powered by Amazon CodeGuru detectors that are informed by years of Amazon.com and AWS security best practices. Examples of security vulnerabilities include resource leaks, hardcoded credentials, and cross-site scripting. To learn more about the security vulnerabilities CodeGuru Security detects, see the Amazon CodeGuru Detector Library.

In addition to security vulnerabilities identified by CodeGuru detectors, security analysis also includes scanning code and text files for hardcoded secrets. For more information, see Secrets detection.

All code scans include code security analysis. You do not need to take any action to enable security analysis in your scans.

Code quality analysis

Code quality analysis detects issues related to quality and maintainability in your code. You can include code quality analysis in addition to security analysis in your scans to ensure your code is meeting quality best practices. Code quality analysis returns findings with an Informational severity level that do not impact the security assessment of your code base.

Code quality analysis is available for most, but not all, integrations. The following list includes the services and integrations in which you can scan your code for both security and quality findings:

- AWS CLI

- AWS SDKs
- GitHub
- Bitbucket
- GitLab
- AWS CodePipeline
- IDE plugins
- Amazon SageMaker Studio and JupyterLab notebooks

Scans created with the console and with Amazon Inspector Lambda code scanning only generate findings related to security.

Enable quality analysis

Scans created in IDE plugins and notebook integrations automatically perform both security and quality analysis.

You can enable quality analysis in scans created with the AWS CLI, AWS SDKs, and the supported integrations by specifying the analysis type when you create a scan. By default, these scans only perform security analysis.

Specify `All` for the analysis type to perform both security and quality analysis in your scans. Specify `Security` to only scan for security vulnerabilities. For more information, see CreateScan in the CodeGuru Security API Reference.

Choose from the list in the Getting started with CodeGuru Security section to learn how to configure code scans to perform quality analysis wherever you are using CodeGuru Security.

Secrets detection

CodeGuru Security integrates with AWS Secrets Manager to use a secrets detector that finds unprotected secrets in your code and text files. Secrets detection is automatically enabled in scans, 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 scan, CodeGuru Security generates a finding with a suggested remediation that tells you about the unprotected secret. To protect secrets, you can store them in AWS Secrets Manager. For more information, see Move hardcoded secrets to AWS Secrets Manager.

Supported character types for secrets detection

CodeGuru Security can detect secrets in English. Valid characters include alphanumeric characters and ASCII special characters.

Supported file types for secrets detection

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

- Class files (*.class)
- Config files (*.config, *.cfg, *.conf, *.cnf, *.cf)
- C# files (*.cs)
- Environment files (*.env)
- Go files (*.go)
- HTML files (*.html)
- Initialization files (*.ini)
- Java files (*.java)
- JavaScript files (*.js, *.mjs, *.cjs)
- JSON files (*.json)
- Jakarta Server Pages files (*.jsp)
- Jupyter Notebook files (*.ipynb)
- Key files (*.key)
- Markdown files (*.md)
- Privacy Enhanced Mail files (*.pem)
- Properties files (*.properties)
- Property List files (*.plist)
- Python files (*.py)
- reStructuredText files (*.rst)
- Ruby files (*.rb)
- Terraform files (*.tf, *.hcl)
- Text files (*.txt, *.text)
- TypeScript files (*.ts)

- TOML files (*.toml)
- XML files (*.xml)
- YAML files (*.yml, *.yaml)

Types of secrets detected by CodeGuru Security

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

Secrets detected by CodeGuru Security

Provider	Secrets detected
Amazon Web Services (AWS)	• Amazon AWS Secret Access Key
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
Salesforce	• Private Key
SendGrid	• SendGrid API Key

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

Create code scans

This section explains how to create code scans and re-run code scans on revised files in the console, with the AWS CLI, and AWS SDKs. For all other tools and services, choose from the list in the Getting started with CodeGuru Security section to learn how code scans are created within each integration.

Topics

Create code scans in the console

This section explains how to create code scans and re-run code scans on revised files in the CodeGuru Security console.

Create a new code scan

The following steps show how to upload your code resources and scan them in the console.

1. Open the Scans page in the CodeGuru Security console at https://console.aws.amazon.com/codeguru/security/scans/.
2. Choose **Create new scan**.
3. Choose **Choose file** and upload the code file you want to scan.
4. For **Scan name**, enter a unique name for the scan. If you don't enter a name, a name will be generated for you with the name of the file, the date, and the time.
5. Choose **Create scan**.
6. Your scan name appears in the **Scans** panel. Under **Scan status**, it displays **In progress** while the scan runs. Once the scan is complete, the status will update to **Complete**.

   If your scan fails, Scan status says **Failed**. .

Scan a revised file

The following steps show how to re-run a scan on revised code files. Be sure to select the appropriate scan name and to upload the corrected version of the code you previously scanned to make sure that vulnerabilities are properly tracked across scans.

1. Choose the scan you want to rerun on the Scans page in the CodeGuru Security console.
2. Choose **Scan revised file** on the top right of the page.
3. Choose **Choose file** and upload the code file you want to scan. Upload the revised version of the file you previously scanned to make sure that vulnerabilities are properly tracked across scans.

   You can't edit the scan name when re-running a scan, since the scan name is used to track findings across revisions to a file.
4. Choose **Create scan**.

5. After the scan is complete, select the scan name to view updated scan metrics based on your revised file.

Create code scans with the AWS CLI and AWS SDKs

This section explains how to upload code resources and create a scan with the AWS CLI and AWS SDKs. You use the `CreateUploadUrl` and `CreateScan` operations, in addition to an HTTP client to upload your code resources. You can also specify the type of analysis to perform in the scan. For information on analysis types, see Types of code scans.

Topics

- Create a scan with the AWS CLI
- Create a scan with AWS SDKs
- Upload code resources

Create a scan with the AWS CLI

> **Note**
>
> You can automate this process using a shell script provided by CodeGuru Security on the Integrations page in the console. For more information, see Automate scans with the AWS CLI.

1. To upload a code resource to scan, you first run the `create-upload-url` command and specify the name of the scan you will run on the code. If this is the first time you are scanning these resources, create a new, unique scan name that you will also use when you create the scan. If you are uploading revised code files to be scanned, use the name of the scan you previously ran on these resources.

   Replace `scan-name` with the name of your scan and run the following command:

   ```
   aws codeguru-security create-upload-url \
   --scan-name scan-name
   ```

This command outputs a URL, a set of headers, and a `codeArtifactId` that you will use in the following steps.

2. Before you create the scan, you need to upload your code to the presigned URL generated in the previous step. You can use any HTTP client to upload code resources, which must be in a zipped code file. For an example, see Upload code resources.

3. After uploading your code to the URL, call run the `create-scan` command to scan your code. For `scan-name`, use the same scan name you specified in the first step. For `resource-id`, use the `codeArtifactId` that was returned in the first step. You can also add the `--analysis-type` option with either `Security` or `All` to specify the type of analysis to perform in the scan.

```
aws codeguru-security create-scan \
--scan-name scan-name
--resource-id '{"codeArtifactId":"codeArtifactId"}'
```

This command outputs a scan state of `InProgress` while CodeGuru Security scans your code. It also returns a `runId` that you can use to run the `get-scan` command to monitor when the scan is complete, and get additional information about the scan.

For more information about using the AWS CLI with CodeGuru Security, see the CodeGuru Security section of the AWS CLI Command Reference.

Create a scan with AWS SDKs

To upload code resources to scan with the AWS SDKs, first use the `CreateUploadUrl` operation to generate an upload URL, request headers, and a code artifact ID. Then, use the request headers to upload your zipped code file to the URL with an HTTP client. For an example, see Upload code resources.

To create the scan, call `CreateScan` with the same scan name you used for CreateUploadUrl and the `codeArtifactId` generated by CreateUploadUrl. You can also specify the `analysisType` option with either `Security` or `All` to specify the type of analysis to perform in the scan. For more information, see the section called “Types of code scans”.

If you are uploading revised code files to be scanned, use the name of the scan you previously ran on these resources for CreateUploadUrl and CreateScan.

Upload code resources

The following is an example of how to upload your zip file with the request headers using the `curl` command. Replace `your-zip-file` with the name of the file that contains your code. Replace `header0 key` and `header0 value` with the first header key and value returned by `CreateUploadUrl`. Add all additional headers using this format. Replace `s3Url` with the URL generated by `CreateUploadUrl`.

```
curl -X PUT \
-T your-zip-file \
-v \
-k \
-H header0 key:header0 value \
-H header1 key:header1 value
s3Url
```

Tag code scans

You can tag code scans when you create them, or tag existing scans. You can use the console, the AWS CLI, or AWS SDKs to tag scans.

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*. 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.

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 scan that you assign to an AWS CodePipeline pipeline. For more information about using tags, see Best Practices for Tagging AWS Resources.

In addition to organizing your resources with tags, you can use tags in IAM policies to help control who can view and interact with your resources. 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*.

Topics

Tag scans in the console

You can only tag scans in the console when you create them.

1. To tag a new scan, open the Scans page in the CodeGuru Security console at https://console.aws.amazon.com/codeguru/security/scans/.
2. Choose **Create new scan**. On the Create scan page, upload your code file and enter a scan name.
3. In the **Tags** panel, choose **Add new tag**. Enter a tag key, and optionally a tag value, for your scan.
4. Choose **Create scan** to create the tagged scan.

Tag scans with the AWS CLI

You can tag new or existing scans with the CLI. To tag a scan when you create it, add the `--tags` option to the `create-scan` command. Specify a tag `key` and an optional tag `value`:

```
aws codeguru-security create-scan \
--scan-name scan-name
--resource-id '{"codeArtifactId": codeArtifactId}'
--tags 'key-1=value-1,key-2=value-2'
```

For more information on creating scans with the CLI, see Create a scan with the AWS CLI.

To tag an existing scan, use the `tag-resource` command. For *`resource-arn`*, use the `scanNameArn` returned by `get-scan` or `list-scans`.

```
aws codeguru-security tag-resource \
--resource-arn scanNameArn
--tags 'key-1=value-1'
```

For more information about using the AWS CLI with CodeGuru Security, see the CodeGuru Security section of the AWS CLI Command Reference.

Tag scans with AWS SDKs

You can tag scans when you create them or tag existing scans with the AWS SDKs.

To tag a new scan, use the `CreateScan` operation and specify the tag key and optional tag value for your scan.

To tag an existing scan, use the `TagResource` operation with the resource ARN, tag key, and optional tag value. For the resource ARN, use the scan name ARN. You can retrieve the `scanNameArn` by calling `ListScans` or `GetScan`.

View all code scans

You can view a list of all your code scans on the **Scans** page in the CodeGuru Security console. The table lists the name, the status, the number of open findings, and the date of the last scan run for each scan in your account.

Scans run in Amazon Inspector, JupyterLab, and Amazon SageMaker Studio do not appear in the console.

To customize the view of the Scans table, choose the gear icon on the upper right side of the Scans table. In the **Preferences** window that appears, you can select page size, display settings, and which columns you want to see.

View scans with the AWS CLI or AWS SDKs

To get a list of all code scans in your account with the AWS CLI or AWS SDKs, use the `ListScans` operation. For more information, see the Amazon CodeGuru Security API Reference.

View code scan details

To view scan details, open the **Scans** page in the CodeGuru Security console and choose the scan you want to view details about.

The overview panel includes information about the scan, such as the number of times you've re-run the scan, the date of the last completed scan, and the scan status of the last scan.

Choose the **Metrics** tab to view data about the scan. Choose the **Findings** tab to view a list of the open findings generated by the scan. Choose the **Tags** tab to view any tags assigned to the scan.

Topics

Scan metrics

Use scan metrics to track finding data across multiple revisions of the same code resource. There are several scan metrics you can monitor for a particular scan.

- **Open findings** – The number of open findings a scan has. You can also see the date of the oldest finding.
- **Closed findings** – The number of closed findings a scan has. You can also see the closure rate, which is the number of closed findings per all findings that have been generated by a scan.
- **Severity distribution** – The number of findings in each severity category generated by the scan. The severity of a finding can be one of five categories: Critical, High, Medium, Low, and Informational. For more information on how severity is defined, see Severity definitions.
- **Vulnerability assessment** - This metric is generated on a weekly basis and tracks which vulnerabilities have generated the current open findings. The graph gives a visual breakdown of how many of each type of vulnerability are present in the scan's open findings.

Scan findings

Use the **Findings** tab on the scan summary page to view a list of a scan's open and closed findings. You can view the severity of a finding, the finding status, the age of a finding in days, and the time the finding was detected. Choose a finding to view details about it. For information about findings, see Working with findings.

You can customize the view of the Findings table by choosing the gear icon on the upper right side of the table. In the **Preferences** window that appears, you can select page size, display settings, and which columns you want to see.

View scan details with the AWS CLI or AWS SDKs

To retrieve code scan metrics with the AWS CLI or AWS SDKs, use the `GetScan` operation with the `scanName` of the scan you want to view metrics about. For more information, see the Amazon CodeGuru Security API Reference.

Working with findings

In CodeGuru Security, a finding is a potential security vulnerability in your code. Findings include information about the vulnerability that was detected in a code scan, an explanation of the issue, the suggested remediation, and the suggested code fix or inline code update to remediate the vulnerability.

You address findings by updating your code based on the suggested remediation. After you make the changes, you re-run the scan on the revised code resource to see if the vulnerability has been remediated and to close the finding. By re-scanning updated code resources, you can track findings across multiple revisions of the same file.

This section covers viewing and addressing findings.

Topics

View all findings

You can view a list of all findings in your account on the **Findings** page in the CodeGuru Security console, or use the AWS CLI and AWS SDKs.

Topics

View findings in the console

The **Overview** panel summarizes the security posture of your code based on open critical findings and provides the number of open critical findings in your account, if any. The **Closed findings** panel lists how many findings you have addressed across all scans. The **Findings summary** panel

indicates how many findings of each severity level are open across all scans. For information about the severity of findings, see Severity definitions.

The **Findings** section lists all findings in your account, including the name of the vulnerability, the name of the scan that generated the finding, the severity of the finding, the age of the finding, the time the finding was detected, and the status of the finding.

You can customize what findings are listed on the Findings page using the dropdown menus next to the search bar. Findings are grouped by severity. By default, findings with critical severity are shown. You can choose the **All severities** dropdown menu to change which finding severity you want to view. Choose the **Open status** dropdown to change the status of findings you want to view.

To customize the view of the Findings table, choose the gear icon on the upper right side of the table. In the **Preferences** window that appears, you can select page size, display settings, and which columns you want to see.

View findings with the AWS CLI or AWS SDKs

To view a list of findings in your account with the AWS CLI or AWS SDKs, use the `ListFindings` or `BatchGetFindings` operations. For more information, see the Amazon CodeGuru Security API Reference.

View finding details

To view finding details in the console, open the **Findings** page in the CodeGuru Security console, and choose the finding you want to view details about. You can also view finding details with the AWS CLI and AWS SDKs.

Topics

- Finding details
- View finding details with the AWS CLI and AWS SDKs

Finding details

The finding details page gives overview information about the finding and the suggested remediation to close the finding. The name of the vulnerability is displayed at the top of the page with a brief description.

- **Overview** - The Overview panel includes the following key information about the finding, in addition to other details like the finding ID, the name of the scan that generated the finding, the file path, and the rule ID. For additional information on these concepts, see the section called "Terminology and metrics".
  - **Vulnerability name** - The name of the detected vulnerability. You can learn more about how the vulnerability was detected by choosing the vulnerability name that links to the corresponding detector in the Amazon CodeGuru Detector Library.
  - **Status** - Finding status can be Open or Closed.
  - **Relevant CWEs** - One or more Common Weakness Enumeration types that apply to the detector that identified the security vulnerability. Choose the link to the CWE to learn more.
  - **Severity** - The severity of the finding can be critical, high, medium, low, or informational. For information about how the severity is calculated, see How severity is calculated.
  - **Vulnerability tags** - Categorizations for this type of vulnerability. You can learn more about similar types of vulnerabilities by choosing the vulnerability tag that redirects you to that tag's page in the Amazon CodeGuru Detector Library.
- **Suggested remediation** - The suggested remediation for a finding describes the vulnerability detected in your code, why it may pose a security risk, and how to remediate it.
- **Suggested code change** - Some findings include inline code updates to replace your vulnerable code. The suggested code change indicates the portion of code where the vulnerability was detected and provides the inline code update to remediate it. Several code changes may be offered for you to select from depending on what solution applies to your use case.

  For more information on updating your code with suggested changes, see Add suggested code changes with the console.
- **Code snippet** - If there is no suggested code change, the code snippet section displays the portion of your code where the vulnerability was detected, and highlights the vulnerable lines of code that need to be updated based on the suggested remediation.

View finding details with the AWS CLI and AWS SDKs

To view finding details with the AWS CLI or AWS SDKs, use the `GetFindings` or `BatchGetFindings` operations. For more information, see the Amazon CodeGuru Security API Reference.

Finding severity

CodeGuru Security defines the severity of the findings detected in your code resources so you can prioritize what vulnerabilities to remediate and track the security posture of your application. The following sections explain what methods are used to determine the severity of findings and what each level of severity means.

How severity is calculated

The severity of a security vulnerability is determined by the detector that generated the finding. Detectors in the Amazon CodeGuru Detector Library are each assigned a severity using the Common Vulnerability Scoring System (CVSS). The CVSS considers how the finding can be exploited in its context (for example, can it be done over internet, or is physical access required) and what level of access can be obtained.

The following table outlines how severity is determined based on the level of access and level of effort required for a bad actor to successfully attack a system.

	Level of Effort			
	Not exploitable	**Requires access to system**	**Internet with high LoE**	**Over internet**
Level of access				
Full control of system or its output	N/A	High	Critical	Critical
Access to sensitive information	N/A	Medium	High	High
Can crash or slow down the system	Low	Low	Medium	Medium

	Level of Effort			
	Not exploitable	**Requires access to system**	**Internet with high LoE**	**Over internet**
Provides additional security	Info	Info	Low	Low
Best practice	Info	N/A	N/A	N/A

Severity definitions

The severity levels are defined as follows.

Critical – The security vulnerability should be remediated immediately to avoid it escalating.

Critical findings suggest that an attacker can gain control of the system or modify its behavior with moderate effort. CodeGuru Security recommends that you treat critical findings with the utmost urgency. You also should consider the criticality of the resource.

High – The security vulnerability must be addressed as a near-term priority.

High severity findings suggest that an attacker can gain control of the system or modify its behavior with high effort. CodeGuru Security recommends that you treat a high severity finding as a near-term priority and that you take immediate remediation steps. You also should consider the criticality of the resource.

Medium – The security vulnerability should be addressed as a midterm priority.

Medium severity findings can lead to crash, unresponsiveness, or unavailability of the system. CodeGuru Security recommends that you investigate the implicated code at your earliest convenience. You also should consider the criticality of the resource.

Low – The security vulnerability does not require action on its own.

Low severity findings suggest programming errors or anti-patterns. You do not need to take immediate action on low severity findings, but they can provide context when you correlate them with other issues.

Informational – No recommended action.

Informational findings include suggestions for quality or readability improvements, or alternative API operations. No immediate action is necessary.

Address findings

After analyzing your finding, update your code based on the suggested remediation, and then re-run the same scan on the updated code resource to make sure the vulnerability was remediated and to close the finding.

If there is a suggested code change, see the following instructions for updating your code with inline code fixes. You can retrieve the suggested code changes from the console, the AWS CLI, and AWS SDKs.

> **Important**
>
> Be sure to keep track of scan names, so you know which scan to re-run on your updated resource. If you scan a different resource, you will compromise metrics that help to monitor the security posture of your applications.

Topics

Add suggested code changes with the console

For some findings, CodeGuru Security highlights the vulnerable sections of your code and provides inline code fixes to remediate the vulnerability. Several code changes may be offered for you to select from depending on what solution applies to your use case.

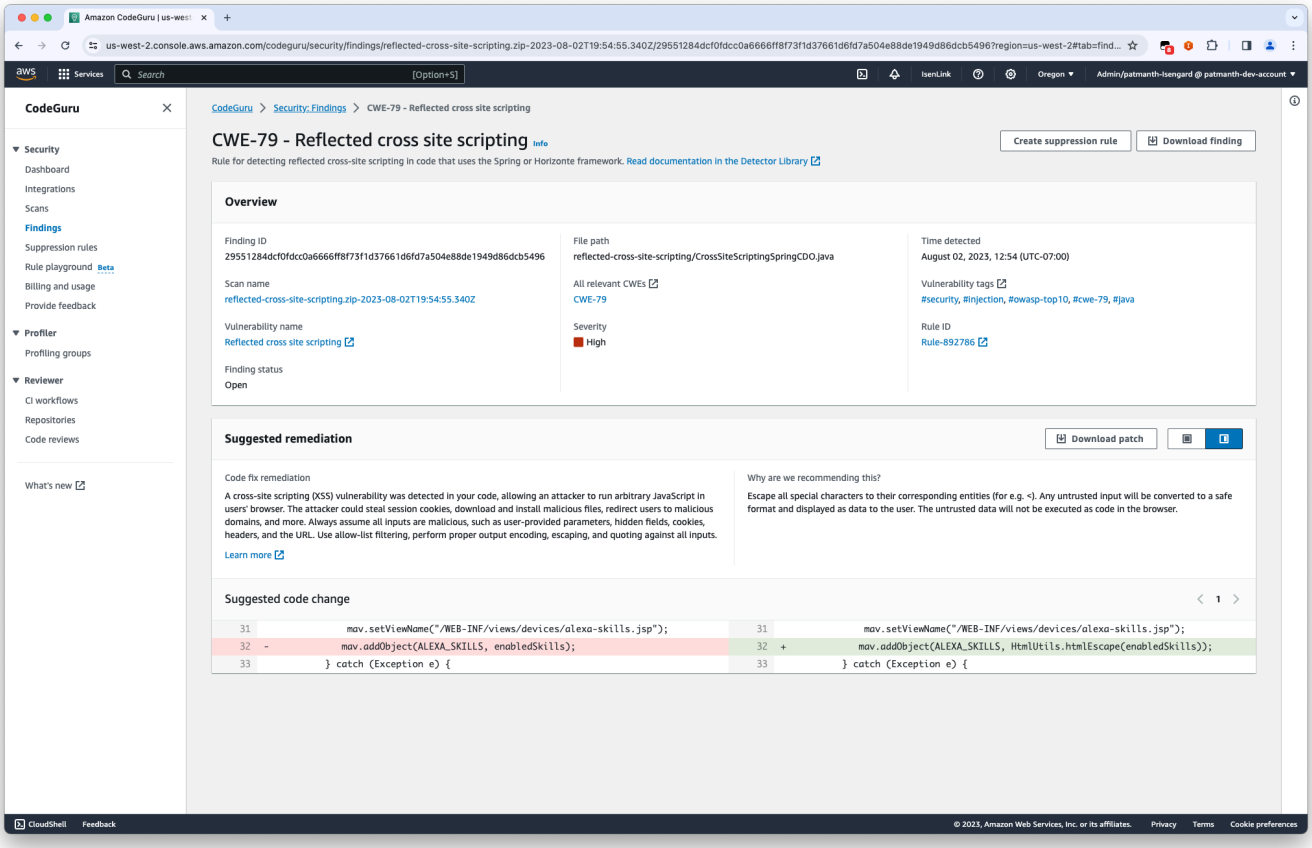
To update your code with a suggested code change using the console, you can download a diff file with the new code or copy and paste the new code into your file.

1. Open the Findings page in the CodeGuru Security console at https://console.aws.amazon.com/codeguru/security/findings/ and choose the finding you want to address.
2. In the **Suggested remediation** panel, you can view the vulnerable lines of code to be removed, the suggested code change, and where to add it. If there are alternate code change solutions, choose the arrows above the code boxes to switch between options.

3. Determine which code fix you want to use to update your code. Then, choose **Download patch** to download the diff file, which shows the vulnerable lines of code to remove and the new code to replace it with.

   Alternatively, you can manually update your resource. Be sure to remove the vulnerable lines of code and add the updated code to the correct section of your code.

   The following image is an example of a vulnerability that you can download a patch for.



4. Once your code is updated, re-run the scan to check that the vulnerability was remediated and to close the finding. For information on how to re-run a scan, see Scan a revised file in the console.

Address findings with the AWS CLI and AWS SDKs

Use the `GetFindings` or `BatchGetFindings` operations to retrieve findings, including the suggested remediation for a given vulnerability. Then, make the change that applies to your use

case and re-run your scan using the same scan name. If you need to make inline code changes, remove the vulnerable lines of code and add the new code to the correct section of your code.

Understanding dashboard metrics

The dashboard page in the CodeGuru Security console shows high-level metrics about findings generated by all scans in an account. The dashboard page visually shows key insights about security issues in your application that you can use to drive business decisions related to the security of your code. Use the dashboard as a vulnerability-tracking tool for your applications by monitoring metrics to track the security of your code over time.

To see metrics for your account, go to the **Dashboard** page in the CodeGuru Security console. There are two sections, a findings overview and a vulnerability fix overview. The findings overview includes metrics about open and critical findings, finding severity, vulnerabilities, and more. The vulnerability resolution overview section provides metrics related to closed findings.

To get account metrics with the AWS CLI or AWS SDKs, call `GetMetricsSummary` or `ListFindingsMetrics`.

This section explains the metrics in the dashboard and how to interpret them.

Topics

Findings overview metrics

Use the **Findings overview** section of the dashboard to monitor metrics related to open findings in your account. The findings overview metrics are refreshed whenever a scan is run and are calculated based on all open findings.

Open and critical findings

The **Open and critical findings** panel displays the total number of open findings and the total number of open critical findings across your account. The percentage next to the absolute number indicates the change in the metric from the previous week.

You can choose the number of findings to be redirected to the findings page and view your open findings.

Monitor the number of open and critical findings in your account periodically to track the security posture of your code.

Severity distribution

The **Severity distribution** panel is a graphical representation of the distribution of the severity of all open findings. The severity of a finding can be one of five categories: Critical, High, Medium, Low, and Informational. For information on how severity is calculated and how severity levels are defined, see Finding severity.

Check the severity distribution of your findings to monitor how severe the vulnerabilities in your code are.

Vulnerability assessment

The **Vulnerability assessment** panel displays the top four vulnerabilities that are found across all open findings. This metric is calculated on a weekly basis.

Use the vulnerability assessment metric to monitor the most common security issues in your code, if they are being remediated, and if certain vulnerabilities become more or less common.

Scans with most findings and most critical findings

The **Scans with most findings** panel lists the top three scans that have generated the most findings and how many open findings each scan has. You can choose **View all scans** to see a list of all scans in your account.

The **Scans with most critical findings** panel lists the top three scans that have generated the most critical findings and how many open critical findings each scan has. You can choose **View all critical findings** to see a list of all open critical findings in your account.

Use these metrics to track which code scans have the most open findings to target the areas of your application that are most unsecure.

Vulnerability fix overview metrics

Use the **Vulnerability fix overview** section of the dashboard to monitor metrics related to closed findings and new findings in your account. Vulnerability fix overview metrics from the past week are shown by default. To change the time period that metrics are calculated for, choose the

dropdown menu labeled **Last week** and choose a time period or a custom range. Metrics are updated whenever you run a scan in your account.

Vulnerability tracking

The **Vulnerability tracking** panel includes the following metrics:

- **Total closed findings** indicates the number of findings that have been closed during the specified time period
- **Average time to close** indicates the average time in days that findings are open, from the initial detection to being closed, during the specified time range
- **New findings** indicates the number of new findings that have been generated during the specified time period

The percentage next to the absolute number indicates the change in the metric during the specified time period.

Use vulnerability tracking metrics to monitor the status of findings over time and track the progress of your application's security posture.

Open versus closed findings

The **Open versus closed findings** panel compares the number of open and closed findings over the specified time period. You can filter the data by finding severity using the dropdown menu above the graph. Choose the **View all findings** button to view a list of all findings on Findings page.

Use open versus closed findings data to track how many findings are being addressed over the course of the chosen time period.

Average time to close

The **Average time to close** panel displays how long it takes to close findings of each severity across your account. You can filter the data by finding severity using the dropdown menu above the graph.

Use this metric to track how frequently you are remediating vulnerabilities based on severity in your application.

Security in Amazon CodeGuru Security

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

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

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

Topics

Data protection in Amazon CodeGuru Security

The AWS shared responsibility model applies to data protection in Amazon CodeGuru Security. 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.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

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

Data captured by CodeGuru Security

CodeGuru Security stores the following to create code scans:

- Scan information, such as scan name and scan ID
- Findings and suggested remediations generated by CodeGuru Security, which include source code snippets and code suggestions
- Source code

- Metadata about scans and findings, which does not include customer information

Data retention

CodeGuru Security stores customer code for up to 10 days. Source code is transiently stored in memory until its code scan is complete. This typically lasts a few hours. When the code scan is complete, it is flushed from memory, encrypted and stored in an Amazon Simple Storage Service bucket owned by CodeGuru Security for up to 10 days, and then deleted.

Code scans, findings, and suggested remediations are currently stored indefinitely.

Encryption at rest

Data collected by Amazon CodeGuru Security is stored using Amazon Simple Storage Service and Amazon DynamoDB. The data at rest is encrypted using AWS encryption solutions by default. Amazon CodeGuru Security encrypts your data, such as code resources and generated security findings, using AWS owned encryption keys from AWS Key Management Service (AWS KMS). You don't have to take any action to protect the AWS managed keys that encrypt your data. For more information see AWS owned keys in the *AWS Key Management Service Developer Guide*.

If you choose to use to protect data with customer-managed KMS keys, it is your responsibility to protect the keys that encrypt your data. For more information on customer-managed KMS keys, see Key management.

Encryption in transit

All communication between customers and CodeGuru Security and between CodeGuru Security and its downstream dependencies is protected using TLS connections that are signed using the Signature Version 4 signing process. All CodeGuru Security 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*.

Key management

CodeGuru Security encrypts your data using one of two types of KMS keys:

- An AWS owned KMS key. This is the default encryption method.

- A customer managed KMS key. To use your own KMS key, you need to create the key and then provide a key policy to grant permissions to CodeGuru Security to use the key.

The following sections explain how to create and use a customer managed KMS key to encrypt your data.

Create a customer managed KMS key

You can create customer managed KMS key using either the AWS KMS console or the `CreateKey` API. When creating the key, you can use an existing symmetric key in your account or create a symmetric customer managed KMS key. CodeGuru Security does not support asymmetric KMS keys. Additionally, you will need a AWS KMS key in the same AWS Region as your scans, or a multi-region key. For more information see Creating symmetric encryption AWS KMS keys in the *AWS KMS user guide*.

Permissions for code encryption with a customer managed KMS key

To use your encryption key, you need to create a policy that allows access to AWS KMS key actions and a policy that allows Amazon CodeGuru Security to use those actions.

If you are setting, updating, or resetting the encryption key for your account it is recommended to use an Amazon CodeGuru Security administrator policy, such as `AmazonCodeGuruSecurityFullAccess`.

The key policy for KMS must allow the following actions:

- `kms:CreateGrant`
- `kms:Decrypt`
- `kms:DescribeKey`
- `kms:GenerateDataKeyWithoutPlainText`
- `kms:Encrypt`
- `kms:RetireGrant`

After you've verified that you have the correct KMS permissions in your key policy, you must create a policy to attach to your CodeGuru Security access role that allows CodeGuru Security to use your key for encryption. Add the following statements to your policy and replace *`<region>`* with the AWS Region where you are running CodeGuru Security scans:

```
{
  "Effect": "Allow",
    "Action": "kms:CreateGrant",
    "Resource": "*",
    "Condition": {
        "ForAllValues:StringEquals": {
            "kms:GrantOperations": [
                "GenerateDataKey",
                "GenerateDataKeyWithoutPlaintext",
                "Encrypt",
                "Decrypt",
                "RetireGrant",
                "DescribeKey"
            ]
        },
        "StringEquals": {
            "kms:ViaService": [
                "codeguru-security.<region>.amazonaws.com"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:RetireGrant",
        "kms:DescribeKey",
        "kms:GenerateDataKeyWithoutPlaintext"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "codeguru-security.<region>.amazonaws.com"
            ]
        }
    }
}
```

For information on key policies, see Using Key Policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

Configure encryption with the Amazon CodeGuru Security API

To set a customer managed key for encryption, use the `UpdateAccountConfiguration` operation. In the API request, for the `EncryptionConfig` object, use the `kmsKeyArn` field to specify the ARN of the AWS KMS encryption key you want to use. If you call `UpdateAccountConfiguration` and pass `null` or nothing for `kmsKeyArn`, an AWS owned key will be used for encryption.

To view the current KMS key ARN that is being used for encryption, call `GetAccountConfiguration`. If you attempt to use `GetAccountConfiguration` when you haven't set a customer managed key, the operation returns `null` which means that an AWS owned key is being used for encryption.

If you delete the key or change it's policy to deny access to Amazon CodeGuru Security you will be unable to access your findings and dashboard metrics and code scans will fail for your account.

Identity and access management for Amazon CodeGuru Security

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

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon CodeGuru Security works with IAM
- Identity-based policy examples for Amazon CodeGuru Security
- AWS managed policies for Amazon CodeGuru Security
- Amazon CodeGuru Security permissions reference
- Troubleshooting Amazon CodeGuru Security identity and access

Audience

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

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

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

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

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 Signing AWS API requests in the *IAM User Guide*.

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

AWS account root user

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

IAM roles

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

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center 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 Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

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

Managing access using policies

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

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

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

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

Identity-based policies

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

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

Resource-based policies

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

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

Access control lists (ACLs)

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

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

Other policy types

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

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If

you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.

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

Multiple policy types

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

How Amazon CodeGuru Security works with IAM

Before you use IAM to manage access to CodeGuru Security, learn what IAM features are available to use with CodeGuru Security.

IAM features you can use with Amazon CodeGuru Security

IAM feature	CodeGuru Security support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Yes

IAM feature	CodeGuru Security support
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	No

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

Identity-based policies for CodeGuru Security

Supports identity-based policies	Yes

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

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

Identity-based policy examples for CodeGuru Security

To view examples of CodeGuru Security identity-based policies, see Identity-based policy examples for Amazon CodeGuru Security.

Resource-based policies within CodeGuru Security

Supports resource-based policies	No

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

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

Policy actions for CodeGuru Security

Supports policy actions	Yes

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

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

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

To see a list of CodeGuru Security actions, see Actions Defined by Amazon CodeGuru Security in the *Service Authorization Reference*.

Policy actions in CodeGuru Security use the following prefix before the action:

```
codeguru-security
```

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

```
"Action": [
      "codeguru-security:action1",
      "codeguru-security:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

```
"Action": "codeguru-security:List*"
```

To view examples of CodeGuru Security identity-based policies, see Identity-based policy examples for Amazon CodeGuru Security.

Policy resources for CodeGuru Security

Supports policy resources	Yes

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

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

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

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

To see a list of CodeGuru Security resource types and their ARNs, see Resources Defined by Amazon CodeGuru Security in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon CodeGuru Security.

To view examples of CodeGuru Security identity-based policies, see Identity-based policy examples for Amazon CodeGuru Security.

Policy condition keys for CodeGuru Security

Supports service-specific policy condition keys	Yes

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

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

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

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

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

To see a list of CodeGuru Security condition keys, see Condition Keys for Amazon CodeGuru Security in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon CodeGuru Security.

To view examples of CodeGuru Security identity-based policies, see Identity-based policy examples for Amazon CodeGuru Security.

ACLs in CodeGuru Security

Supports ACLs	No

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

ABAC with CodeGuru Security

Supports ABAC (tags in policies)	Yes

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

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

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

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

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

Using temporary credentials with CodeGuru Security

Supports temporary credentials	Yes

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

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your

company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

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

Cross-service principal permissions for CodeGuru Security

Supports forward access sessions (FAS)	Yes

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 roles for CodeGuru Security

Supports service roles	No

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

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

Service-linked roles for CodeGuru Security

Supports service-linked roles	No

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

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon CodeGuru Security

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

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

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

Topics

- Policy best practices
- Using the CodeGuru Security console
- Allow users to view their own permissions

Policy best practices

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

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

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

Using the CodeGuru Security console

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

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

To ensure that users and roles can still view information in the CodeGuru Security console, their permissions policy must include the following actions:

- `codeguru-security:BatchGetFindings`
- `codeguru-security:GetAccountConfiguration`
- `codeguru-security:GetFindings`
- `codeguru-security:GetMetricsSummary`
- `codeguru-security:GetScan`
- `codeguru-security:ListFindingsMetrics`
- `codeguru-security:ListScans`
- `codeguru-security:ListTagsForResource`

Allow users to view their own permissions

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
```

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

AWS managed policies for Amazon CodeGuru Security

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed

policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

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

Topics

AmazonCodeGuruSecurityFullAccess

You can attach the `AmazonCodeGuruSecurityFullAccess` policy to your IAM identities.

This policy grants full access to Amazon CodeGuru Security.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AmazonCodeGuruSecurityFullAccess",
```

```
   "Effect": "Allow",
   "Action": [
    "codeguru-security:*"
   ],
   "Resource": "*"
  }
 ]
}
```

AmazonCodeGuruSecurityScanAccess

You can attach the `AmazonCodeGuruSecurityScanAccess` policy to your IAM identities.

This policy grants permissions that allow a user to work with scans, including creating scans, viewing scan information, and viewing scan findings.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AmazonCodeGuruSecurityScanAccess",
   "Effect": "Allow",
   "Action": [
    "codeguru-security:CreateScan",
    "codeguru-security:CreateUploadUrl",
    "codeguru-security:GetScan",
    "codeguru-security:GetFindings"
   ],
   "Resource": "arn:aws:codeguru-security:*:*:scans/*"
  }
 ]
}
```

CodeGuru Security updates to AWS managed policies

View details about updates to AWS managed policies for CodeGuru Security since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CodeGuru Security Document history page.

Change	Description	Date
AmazonCodeGuruSecurityScanAccess – New policy	CodeGuru Security added a new policy that grants permissions to create and view scans and view scan findings.	May 10, 2023
AmazonCodeGuruSecurityFullAccess – New policy	CodeGuru Security added a new policy to allow full access to CodeGuru Security.	May 10, 2023
CodeGuru Security started tracking changes	CodeGuru Security started tracking changes for its AWS managed policies.	May 10, 2023

Amazon CodeGuru Security permissions reference

You can use AWS condition keys in your Amazon CodeGuru Security 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-security:` prefix followed by the API operation name (for example, `codeguru-security:CreateUploadUrl` and `codeguru-security:CreateScan`). To specify multiple actions in a single statement, separate them with commas (for example, `"Action": [ "codeguru-security:CreateUploadUrl", "codeguru-security:CreateScan" ]`).

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-security:*` specifies all Amazon CodeGuru Security actions and `codeguru-security:Get*` specifies all Amazon CodeGuru Security actions that begin with the word `Get`.

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

Troubleshooting Amazon CodeGuru Security identity and access

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

Topics

I am not authorized to perform an action in CodeGuru Security

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-security:CreateScan` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
            codeguru-security:CreateScan on resource: my-example-code-scan
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *my-example-code-scan* resource using the `codeguru-security:CreateScan` action.

I want to allow people outside of my AWS account to access my CodeGuru Security resources

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

To learn more, consult the following:

- To learn whether CodeGuru Security supports these features, see How Amazon CodeGuru Security works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.

- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for Amazon CodeGuru Security

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

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

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

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

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- 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.

Resilience in Amazon CodeGuru Security

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

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

Infrastructure security in Amazon CodeGuru Security

As a managed service, Amazon CodeGuru Security 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 Security 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.

Monitoring Amazon CodeGuru Security

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

- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see Amazon EventBridge User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Monitoring Amazon CodeGuru Security events in Amazon EventBridge

You can monitor Amazon CodeGuru Security events in EventBridge, which delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services. EventBridge routes that data to targets such as AWS Lambda and Amazon Simple Notification Service. These events are the same as those that appear in Amazon CloudWatch Events, which delivers a near real-time stream of system events that describe changes in AWS resources.

Logging Amazon CodeGuru Security API calls using AWS CloudTrail

Amazon CodeGuru Security is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CodeGuru Security. CloudTrail captures all API calls for CodeGuru Security as events. The calls captured include calls from the CodeGuru Security console and code calls to the CodeGuru Security API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for CodeGuru Security. 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 Security, 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, see the AWS CloudTrail User Guide.

CodeGuru Security information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in CodeGuru Security, that activity is recorded in a CloudTrail event along 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.

For an ongoing record of events in your AWS account, including events for CodeGuru Security, 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 upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All CodeGuru Security actions are logged by CloudTrail and are documented in the Amazon CodeGuru Security API Reference. For example, calls to the `CreateScan`, `GetScan` and `GetFindings` actions generate entries in the CloudTrail log files.

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 AWS Identity and Access Management (IAM) 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.

Understanding CodeGuru Security 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 `CreateScan` action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:i-1234567890abcdef0",
        "arn": "arn:aws:sts::123456789012:assumed-role/user-name",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/user-name",
                "accountId": "123456789012",
                "userName": "user-name"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-24T00:38:51Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-02-24T00:39:16Z",
    "eventSource": "codeguru-security.amazonaws.com",
    "eventName": "CreateScan",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "205.251.233.176",
    "userAgent": "aws-sdk-java/Linux/x.xx.fleetxen Java_HotSpot(TM)_64-Bit_Server_VM/
xx",
    "requestParameters": {
```

```
        "resourceId": {
            "codeArtifactId": "cb8c167e-EXAMPLE"
        },

        "clientToken": "e3c6f4ce-EXAMPLE"
    },
    "responseElements": {
        "scanName": "a4469191-EXAMPLE",
        "resourceId": {
            "codeArtifactId": "cb8c167e-EXAMPLE"
        },
        "runId": "a4469191-EXAMPLE",
        "scanState": "InProgress"
    },
    "requestID": "07c4a4de-EXAMPLE",
    "eventID": "711cb5a3-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Quotas for Amazon CodeGuru Security

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for CodeGuru Security, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **CodeGuru Security**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the limit increase form.

Your AWS account has the following quotas related to CodeGuru Security.

Code resources

Resource	Default
Maximum input file size	5 GB
Maximum Java source code size	300 MB
Maximum JavaScript source code size	300 MB
Maximum Python source code size	50 MB

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

The following fixed quotas apply to creating, deploying, and managing an API in CodeGuru Security, 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 two CodeGuru Security APIs is 10 requests per second per account. None of these quotas can be increased. For a list of all CodeGuru Security APIs, see Amazon CodeGuru Security Actions.

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

Action	Default quota	Can be increased
CreateUploadUrl	2 requests every second per account	No
CreateScan	2 requests every second per account	No

Document history for the CodeGuru Security User Guide

The following table describes the documentation releases for CodeGuru Security.

Change	Description	Date
Preview release	This is the preview release of the *Amazon CodeGuru Security User Guide*.	June 13, 2023
New policies: AmazonCodeGuruSecurityFullAccess and AmazonCodeGuruSecurityScanAccess	Added AmazonCodeGuruSecurityFullAccess to allow full access to CodeGuru Security and AmazonCodeGuruSecurityScanAccess to grant permissions to create and view scans and view scan findings.	May 10, 2023
Initial release	Initial release of the *Amazon CodeGuru Security User Guide*.	May 10, 2023