Amazon CodeCatalyst User Guide

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

What is Amazon CodeCatalyst? .......................................................................................................................... 1  
What can I do with CodeCatalyst? .................................................................................................................... 1  
How do I get started with CodeCatalyst? .......................................................................................................... 1  
Learn more about CodeCatalyst ......................................................................................................................... 1  

Concepts ............................................................................................................................................................ 3  
Spaces ............................................................................................................................................................... 3  
Projects .............................................................................................................................................................. 3  
Blueprints .......................................................................................................................................................... 4  
Account connections ........................................................................................................................................ 4  
AWS Builder ID ................................................................................................................................................ 4  
User profiles in CodeCatalyst ............................................................................................................................ 4  
Source repositories ......................................................................................................................................... 5  
Commits ............................................................................................................................................................ 5  
Dev Environments .......................................................................................................................................... 5  
Workflows ....................................................................................................................................................... 5  
Actions ............................................................................................................................................................ 6  
Issues ............................................................................................................................................................... 6  
Personal access tokens (PATs) .......................................................................................................................... 6  
Roles ................................................................................................................................................................. 6  

Setting up .......................................................................................................................................................... 7  
Signing up and creating resources ................................................................................................................... 8  
  Creating your first space and IAM roles ........................................................................................................ 8  
Accept an invitation .......................................................................................................................................... 11  
  Accepting an invitation and creating an AWS Builder ID ............................................................................ 12  
Sign in as a returning user ................................................................................................................................. 13  
  Trusted devices .............................................................................................................................................. 13  

Viewing and managing CodeCatalyst profiles ................................................................................................. 13  
  Viewing your CodeCatalyst profile ................................................................................................................ 14  
  Viewing another user’s CodeCatalyst profile ............................................................................................... 14  
  Updating your profile .................................................................................................................................. 14  
  Changing your CodeCatalyst password ...................................................................................................... 15  

Setting up to use the AWS CLI with CodeCatalyst ......................................................................................... 15  

Getting started tutorials ................................................................................................................................. 17  
Tutorial: Creating a project with the Modern three-tier web application blueprint ............................................ 17  
  Prerequisites ................................................................................................................................................ 19  
  Step 1: Create the Modern three-tier web application project ..................................................................... 20  
  Step 2: Invite someone to your project ......................................................................................................... 20  
  Step 3: Create issues to collaborate on and track work .............................................................................. 21  
  Step 4: View your source repository ......................................................................................................... 21  
  Step 5: Create a Dev Environment with a test branch and make a quick code change .............................. 22  
  Step 6: View the workflow that builds the modern application .................................................................. 23  
  Step 7: Ask others to review your changes .................................................................................................. 25  
  Step 8: Close the issue .................................................................................................................................. 27  
  Step 9: Clean up resources .......................................................................................................................... 27  
  Reference ....................................................................................................................................................... 28  

Tutorial: Starting with an empty project ........................................................................................................... 29  
  Prerequisites ................................................................................................................................................ 29  
  Create an empty project ............................................................................................................................... 29  
  Create a source repository .......................................................................................................................... 30  
  Create a workflow to build, test, and deploy a code change ..................................................................... 31  
  Invite someone to your project .................................................................................................................... 31  
  Create issues to collaborate on and track work ......................................................................................... 31  

Spaces .............................................................................................................................................................. 32  
Viewing projects in your space ........................................................................................................................ 35
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modern three-tier web application blueprint workflow <strong>OnPullRequest</strong> fails with permissions</td>
<td>75</td>
</tr>
<tr>
<td>error for Amazon CodeGuru</td>
<td>78</td>
</tr>
<tr>
<td>Still looking to solve your problem?</td>
<td>79</td>
</tr>
<tr>
<td>Source repositories</td>
<td>80</td>
</tr>
<tr>
<td>Source repository concepts</td>
<td>80</td>
</tr>
<tr>
<td>Projects</td>
<td>83</td>
</tr>
<tr>
<td>Source repositories</td>
<td>83</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>83</td>
</tr>
<tr>
<td>Personal access tokens (PATs)</td>
<td>86</td>
</tr>
<tr>
<td>Branches</td>
<td>88</td>
</tr>
<tr>
<td>Default branches</td>
<td>88</td>
</tr>
<tr>
<td>Commits</td>
<td>88</td>
</tr>
<tr>
<td>Pull requests</td>
<td>89</td>
</tr>
<tr>
<td>Revisions</td>
<td>89</td>
</tr>
<tr>
<td>Workflows</td>
<td>89</td>
</tr>
<tr>
<td>Setting up</td>
<td>89</td>
</tr>
<tr>
<td>Install Git</td>
<td>89</td>
</tr>
<tr>
<td>Create a personal access token</td>
<td>89</td>
</tr>
<tr>
<td>Getting started with source repositories</td>
<td>89</td>
</tr>
<tr>
<td>Creating a project using a blueprint</td>
<td>89</td>
</tr>
<tr>
<td>Viewing the repositories for a project</td>
<td>89</td>
</tr>
<tr>
<td>Creating a Dev Environment</td>
<td>89</td>
</tr>
<tr>
<td>Creating a pull request</td>
<td>90</td>
</tr>
<tr>
<td>Merging a pull request</td>
<td>90</td>
</tr>
<tr>
<td>Viewing the deployed code</td>
<td>90</td>
</tr>
<tr>
<td>Cleaning up resources</td>
<td>90</td>
</tr>
<tr>
<td>Working with source repositories</td>
<td>91</td>
</tr>
<tr>
<td>Creating a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Linking a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Viewing a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Editing the settings for a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Cloning a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Deleting a source repository</td>
<td>91</td>
</tr>
<tr>
<td>Working with branches</td>
<td>94</td>
</tr>
<tr>
<td>Creating and deleting a branch</td>
<td>94</td>
</tr>
<tr>
<td>View and change the default branch for a repository</td>
<td>94</td>
</tr>
<tr>
<td>Git commands for branches</td>
<td>94</td>
</tr>
<tr>
<td>Viewing branches and details</td>
<td>94</td>
</tr>
<tr>
<td>Working with files</td>
<td>97</td>
</tr>
<tr>
<td>Creating or adding a file</td>
<td>97</td>
</tr>
<tr>
<td>Viewing a file</td>
<td>97</td>
</tr>
<tr>
<td>Editing a file</td>
<td>97</td>
</tr>
<tr>
<td>Renaming or deleting a file</td>
<td>97</td>
</tr>
<tr>
<td>Working with pull requests</td>
<td>99</td>
</tr>
<tr>
<td>Creating a pull request</td>
<td>99</td>
</tr>
<tr>
<td>Viewing pull requests</td>
<td>99</td>
</tr>
<tr>
<td>Reviewing a pull request</td>
<td>99</td>
</tr>
<tr>
<td>Updating a pull request</td>
<td>99</td>
</tr>
<tr>
<td>Merging a pull request</td>
<td>99</td>
</tr>
<tr>
<td>Closing a pull request</td>
<td>99</td>
</tr>
<tr>
<td>Working with commits</td>
<td>102</td>
</tr>
<tr>
<td>Viewing commits to a branch</td>
<td>102</td>
</tr>
<tr>
<td>Changing how commits are displayed (CodeCatalyst console)</td>
<td>102</td>
</tr>
<tr>
<td>Quotas for source repositories</td>
<td>102</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>102</td>
</tr>
<tr>
<td>Creating a Dev Environment</td>
<td>102</td>
</tr>
<tr>
<td>Supported integrated development environments for Dev Environments</td>
<td>102</td>
</tr>
</tbody>
</table>
Deploying using workflows ................................................................. 163
  How do I deploy an application? .................................................... 164
  List of deploy actions ................................................................. 164
  Benefits of deploy actions .......................................................... 165
  Alternatives to deploy actions ...................................................... 165
  Tutorial: Deploy using CloudFormation .......................................... 165
  Tutorial: Deploy to Amazon ECS .................................................... 183
  Tutorial: Deploy to Amazon EKS .................................................... 207
  Adding the "Deploy to Amazon ECS" action ...................................... 230
  Adding the "Deploy AWS CloudFormation stack" action ....................... 231
  Adding the "AWS CDK deploy" action ............................................ 234
  Working with deployments .......................................................... 238
Working with workflows .................................................................... 246
  Creating, editing, and deleting a workflow ....................................... 246
  Working with actions ....................................................................... 249
  Working with artifacts ..................................................................... 286
  Working with compute and runtime environment Docker images ............ 293
  Working with environments ............................................................ 307
  Working with file caching ............................................................... 311
  Working with runs .......................................................................... 314
  Working with secrets ....................................................................... 322
  Working with sources ...................................................................... 325
  Working with triggers ...................................................................... 327
  Working with variables .................................................................... 337
Workflow definition reference .............................................................. 354
  Example of a workflow definition file .............................................. 354
  Syntax guidelines and conventions .................................................. 355
  Top-level properties ........................................................................ 356
  Build and test action reference ....................................................... 364
  "Amazon S3 publish" action reference .............................................. 379
  "AWS CDK bootstrap" action reference ............................................ 386
  "AWS CDK deploy" action reference ............................................... 393
  "AWS Lambda invoke" action reference .......................................... 402
  "Deploy AWS CloudFormation stack" action reference ....................... 411
  "Deploy to Amazon ECS" action reference ....................................... 424
  "Deploy to Kubernetes cluster" action reference ................................ 432
  "GitHub Actions" action reference .................................................. 439
  "Render task definition" action reference ........................................ 452
Quotas for workflows .......................................................................... 457
Issues ................................................................................................. 459
Issues concepts .................................................................................. 460
  Active issues .................................................................................. 460
  Archived issues ............................................................................. 460
  Assignee ......................................................................................... 460
  Custom fields .................................................................................. 460
  Estimate .......................................................................................... 460
  Issue ................................................................................................. 460
  Label ............................................................................................... 460
  Priority ............................................................................................ 461
  Status and status categories ......................................................... 461
  Views ............................................................................................... 461
Creating an issue ................................................................................ 461
Editing and collaborating on issues ...................................................... 462
  Editing an issue .............................................................................. 462
  Working with attachments ............................................................. 463
  Marking an issue as blocked or unblocked ........................................ 464
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding, editing, or deleting comments</td>
<td>465</td>
</tr>
<tr>
<td>Progressing an issue</td>
<td>466</td>
</tr>
<tr>
<td>Archiving an issue</td>
<td>467</td>
</tr>
<tr>
<td>Finding and viewing issues</td>
<td>468</td>
</tr>
<tr>
<td>Searching for an issue</td>
<td>468</td>
</tr>
<tr>
<td>Sorting issues</td>
<td>468</td>
</tr>
<tr>
<td>Grouping issues</td>
<td>469</td>
</tr>
<tr>
<td>Filtering issues</td>
<td>469</td>
</tr>
<tr>
<td>Creating an issues view</td>
<td>470</td>
</tr>
<tr>
<td>Exporting issues</td>
<td>470</td>
</tr>
<tr>
<td>Configuring issues settings</td>
<td>470</td>
</tr>
<tr>
<td>Enabling or disabling multiple assignees</td>
<td>471</td>
</tr>
<tr>
<td>Configuring issue effort estimation</td>
<td>471</td>
</tr>
<tr>
<td>Statuses</td>
<td>471</td>
</tr>
<tr>
<td>Labels</td>
<td>473</td>
</tr>
<tr>
<td>Custom fields</td>
<td>473</td>
</tr>
<tr>
<td>Viewing and managing attachments</td>
<td>474</td>
</tr>
<tr>
<td>Markdown tips and tricks</td>
<td>474</td>
</tr>
<tr>
<td>Quotas for issues</td>
<td>475</td>
</tr>
<tr>
<td>Identity, permissions, and access</td>
<td>476</td>
</tr>
<tr>
<td>Working with roles</td>
<td>476</td>
</tr>
<tr>
<td>Space administrator role</td>
<td>477</td>
</tr>
<tr>
<td>Power user role</td>
<td>483</td>
</tr>
<tr>
<td>Limited access role</td>
<td>484</td>
</tr>
<tr>
<td>Project administrator role</td>
<td>485</td>
</tr>
<tr>
<td>Contributor role</td>
<td>488</td>
</tr>
<tr>
<td>Reviewer role</td>
<td>492</td>
</tr>
<tr>
<td>Read only role</td>
<td>495</td>
</tr>
<tr>
<td>Viewing and changing user roles</td>
<td>496</td>
</tr>
<tr>
<td>Managing personal access tokens</td>
<td>497</td>
</tr>
<tr>
<td>Creating PATs</td>
<td>498</td>
</tr>
<tr>
<td>Viewing PATs</td>
<td>499</td>
</tr>
<tr>
<td>Deleting PATs</td>
<td>500</td>
</tr>
<tr>
<td>Multi-factor authentication (MFA) in Amazon CodeCatalyst</td>
<td>500</td>
</tr>
<tr>
<td>How to register a device for use with multi-factor authentication</td>
<td>501</td>
</tr>
<tr>
<td>Authenticator applications</td>
<td>502</td>
</tr>
<tr>
<td>Changing your MFA devices</td>
<td>502</td>
</tr>
<tr>
<td>Security</td>
<td>503</td>
</tr>
<tr>
<td>Data protection</td>
<td>504</td>
</tr>
<tr>
<td>CodeCatalyst and Identity and Access Management</td>
<td>505</td>
</tr>
<tr>
<td>Compliance validation</td>
<td>541</td>
</tr>
<tr>
<td>Resilience</td>
<td>541</td>
</tr>
<tr>
<td>Infrastructure Security</td>
<td>542</td>
</tr>
<tr>
<td>Configuration and vulnerability analysis</td>
<td>542</td>
</tr>
<tr>
<td>Your data and privacy in Amazon CodeCatalyst</td>
<td>542</td>
</tr>
<tr>
<td>Best practices for workflow actions</td>
<td>543</td>
</tr>
<tr>
<td>CodeCatalyst trust model</td>
<td>543</td>
</tr>
<tr>
<td>Monitoring in Amazon CodeCatalyst</td>
<td>545</td>
</tr>
<tr>
<td>Loggin CodeCatalyst API calls to connected accounts in AWS CloudTrail</td>
<td>546</td>
</tr>
<tr>
<td>Accessing logged events in CodeCatalyst</td>
<td>551</td>
</tr>
<tr>
<td>Quotas for identity, permission, and access</td>
<td>553</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>554</td>
</tr>
<tr>
<td>Problems signing up</td>
<td>554</td>
</tr>
<tr>
<td>Problems signing in</td>
<td>555</td>
</tr>
<tr>
<td>Problems signing out</td>
<td>555</td>
</tr>
<tr>
<td>I get a role does not exist error for a failed workflow</td>
<td>555</td>
</tr>
<tr>
<td>I get a role error for a failed workflow</td>
<td>556</td>
</tr>
</tbody>
</table>
Troubleshooting source repositories .......................................................... 585
  I have reached the maximum storage for my space and see warnings or errors ...... 585
  I receive an error when trying to clone or push to an Amazon CodeCatalyst source repository ... 585
  I need a source repository for my project ................................................ 586
  My source repository is brand-new but contains a commit ................................ 586
  I want a different branch as my default branch ........................................... 586
  I am receiving emails about activity in pull requests .................................... 587
  I forgot my personal access token (PAT) ..................................................... 587
  A pull request doesn't display the changes I expect ..................................... 587
  A pull request shows a status of Not mergeable ......................................... 587
Troubleshooting workflows ............................................................................ 588
  How do I fix "Workflow is inactive" messages? ............................................. 588
  How do I fix "Unable to locate credentials" and "ExpiredToken" errors? ............... 589
Amazon CodeCatalyst User Guide

How do I fix "Unable to connect to the server" errors? ........................................... 590
How do I fix workflow definition errors? ............................................................. 590
Why are CodeDeploy fields missing from the visual editor? .......................... 591
How do I fix IAM capabilities errors? ............................................................... 591
How do I fix "npm install" errors? ................................................................. 592
Why do multiple workflows have the same name? ............................................ 595
Can I store my workflow definition files in another folder? .......................... 595
How do I add actions in sequence to my workflow? ........................................... 596
Why does my workflow successfully validate but fail at runtime? ................... 596
Auto-discovery doesn't discover any reports for my action ....................... 596
My action fails on auto-discovered reports after I configure success criteria .... 596
Auto-discovery generates reports that I don't want ...................................... 596
Auto-discovery generates many small reports for a single test framework ...... 597
Workflows listed under CI/CD don't match those in the source repository ....... 597

Troubleshooting search .................................................................................. 598
I can't find a user in my project ....................................................................... 598
I don't see what I'm looking for in my project or space ............................... 598
Number of search results keep changing when I navigate through the pages ... 598
My search query isn't being completed ......................................................... 598

Troubleshooting associated accounts .............................................................. 599
My AWS account connection request receives an invalid token error ............. 599
My Amazon CodeCatalyst project workflow fails with an error for the configured account, environment, or IAM role .................................................. 599
I need an associated account, role, and environment to create a project ........ 600
I cannot access the Amazon CodeCatalyst Spaces page in the AWS Management Console ................................. 601
I cannot change the billing tier for my space ................................................. 601
I want a different account as my billing account ......................................... 586

Troubleshooting Dev Environments ............................................................... 602
My Dev Environment creation didn't succeed .............................................. 602
My Dev Environment didn't resume ............................................................. 602
My Dev Environment disconnected .............................................................. 602
I can't find which directory my project is in ................................................. 602
Troubleshooting IDEs .................................................................................. 602
Troubleshooting devfiles ............................................................................. 603

Troubleshooting issues .................................................................................. 605
I can't choose an assignee for my issue ....................................................... 605

Troubleshooting AWS CLI and SDK issues .................................................. 605
I receive an error when I enter <code>aws codecatalyst</code> at a command line or terminal saying it's an invalid choice ................................................. 606
I receive a credentials error when I run <code>aws codecatalyst</code> commands ........ 606

CodeCatalyst health report .......................................................................... 607
CodeCatalyst health report concepts .............................................................. 607
Incident ........................................................................................................ 607
Status ......................................................................................................... 607
Impacted capabilities .................................................................................. 607
Updated on ................................................................................................. 607

Billing ........................................................................................................... 608
Viewing your CodeCatalyst plan and billing ............................................... 608
Viewing CodeCatalyst accounts for billing in AWS .................................... 609
Turning on the Standard tier for a billing account ..................................... 609
Changing your CodeCatalyst billing tier ..................................................... 610
Setting up a billing account ....................................................................... 611
Step 1: Add the billing account to your space .......................................... 611
Step 2: Turn on the Standard tier for your billing account ................. 611
Step 3: Choose the billing account in CodeCatalyst ................................ 611
Changing a billing account ....................................................................... 612
Getting AWS billing information ............................................................... 612

x
Amazon CodeCatalyst User Guide

What can I do with CodeCatalyst?

What is Amazon CodeCatalyst?

Amazon CodeCatalyst is an integrated service for software development teams adopting continuous integration and deployment practices into their software development process. CodeCatalyst puts the tools you need all in one place. You can plan work, collaborate on code, and build, test, and deploy applications with continuous integration/continuous delivery (CI/CD) tools. You can also integrate AWS resources with your projects by connecting your AWS accounts to your CodeCatalyst space. By managing all of the stages and aspects of your application lifecycle in one tool, you can deliver software quickly and confidently.

In CodeCatalyst, you create a space to represent your company, department, or group, and then you create projects that contain the resources needed to support your development teams and tasks. CodeCatalyst resources are structured inside projects that live inside spaces. To help teams get started quickly, CodeCatalyst provides language- or tool-based project blueprints. When you create a project from a project blueprint, the project comes with resources such as a source repository with sample code, build scripts, deployment actions, virtual servers or serverless resources, and more.

What can I do with CodeCatalyst?

You and your development team can use CodeCatalyst to carry out each aspect of software development, from planning your work to deploying your applications. You can use CodeCatalyst to:

- **Iterate and collaborate on code** – Work collaboratively with your team on code with branches, merges, pull requests, and comments in your source code repositories. Create Dev Environments to work on code quickly without having to clone or set up connections to repositories.
- **Build, test, and deploy your application with workflows** – Configure workflows with build, test, and deploy actions to handle the continuous integration and delivery of your applications. You can either start workflows manually or configure them to start automatically based on events such as code pushes or creating or closing pull requests.
- **Prioritize your team’s work with issue tracking** – Use issues to create backlogs and monitor the status of in-progress tasks with boards. Creating and maintaining a healthy backlog of items for your team to work on is an important part of developing software.
- **Set up monitoring and notifications** – Monitor team activity and resource status, and configure notifications to stay up to date with important changes.

How do I get started with CodeCatalyst?

If you’re new to CodeCatalyst, we recommend that you get started by:

- Reviewing the CodeCatalyst concepts (p. 3)
- Creating additional spaces in CodeCatalyst (p. 53)
- Creating your first project by following the steps in Tutorial: Creating a project with the Modern three-tier web application blueprint (p. 17)

Learn more about CodeCatalyst

You can learn more about the functionality of CodeCatalyst in this user guide, as well as the following resources:
• AWS DevOps Blog articles about Amazon CodeCatalyst
• The Amazon CodeCatalyst API Reference Guide
• The Amazon CodeCatalyst Action Development Kit Developer Guide
• CodeCatalyst FAQ
• Testimonials
CodeCatalyst concepts

Get familiar with the key concepts to help speed up your collaboration and application development in Amazon CodeCatalyst. These concepts include terms used in source control, continuous integration and continuous delivery (CI/CD), and modeling and configuring automated release processes.

For additional conceptual information, see the following topics:

- Source repository concepts (p. 80)
- Workflows concepts (p. 132)

Topics

- Spaces (p. 3)
- Projects (p. 3)
- Blueprints (p. 4)
- Account connections (p. 4)
- AWS Builder ID (p. 4)
- User profiles in CodeCatalyst (p. 4)
- Source repositories (p. 5)
- Commits (p. 5)
- Dev Environments (p. 5)
- Workflows (p. 5)
- Actions (p. 6)
- Issues (p. 6)
- Personal access tokens (PATs) (p. 6)
- Roles (p. 6)

Spaces

A space represents your company, department, or group. You must create a space to add projects, members, and the associated cloud resources you create in CodeCatalyst. When a user accepts an invitation to a project, CodeCatalyst automatically adds them to the space that contains that project. Users with the Space administrator role have permissions to manage the space.

For more information about spaces see Spaces in CodeCatalyst (p. 32).

Projects

A project represents a collaborative effort in CodeCatalyst that supports development teams and tasks. After you have a project, you can add, update, or remove users and resources, customize your project dashboard, and monitor the progress of your team's work. You can have multiple projects within a space.

For more information about projects, see Projects in CodeCatalyst (p. 57).
Blueprints

A blueprint is a project synthesizer that generates and extends application support files and dependencies for you, along with creating your CodeCatalyst project in the console. You choose a project type from a selection of blueprints in CodeCatalyst, view the README file, and preview the project repository and resources that will be generated. Your project is generated from the base configuration specified by the blueprint. You synthesize to the project blueprint periodically, which updates your project files, such as software dependencies, and regenerates resources. Projects use a tool called Projen to synthesize projects by syncing the latest project updates and generating support files. These files may include package.json, Makefile, eslint, and more based on your application type and language. Project blueprints can generate files supporting AWS resources such as CDK constructs, AWS CloudFormation templates, and AWS Serverless Application Model templates.

For more information about project blueprints, see Project blueprint reference (p. 65).

Account connections

An account connection associates a CodeCatalyst space with your AWS account. After your account connection is set up, the AWS account is made available to the space. You can then add IAM roles to CodeCatalyst so that it can access resources in your AWS account. You can also use these roles for your CodeCatalyst workflow actions.

For more information about account connections, see Managing AWS accounts (p. 36).

AWS Builder ID

An AWS Builder ID is a personal identity you can use to sign up and sign in to CodeCatalyst and other participating applications. It is not the same as an AWS account. Your AWS Builder ID manages metadata such as user alias and email address. Your AWS Builder ID is a unique identity that supports users across all spaces in CodeCatalyst. For information about accessing your AWS Builder ID profile, see Updating your profile (p. 14). To learn more about AWS Builder ID, see AWS Builder ID in the AWS General Reference.

For more information about signing up and signing in, see Setting up CodeCatalyst (p. 7).

User profiles in CodeCatalyst

You access your CodeCatalyst user profile by choosing the profile option from the drop-down under your login initials on any page in CodeCatalyst. You can create personal access tokens (PATs) from your profile page, but you can only view or delete PATs using the AWS CLI. Your user name is the alias you chose when you signed up. You cannot change your user name. To view the profile page for another CodeCatalyst user, go to the Members tab for your project and choose the appropriate user.

You access your AWS Builder ID by viewing your CodeCatalyst profile and then choosing to go to AWS Builder ID. You will be redirected to your AWS Builder ID profile page. Your profile’s full name, email address, and password are managed by your AWS Builder ID, and you can edit that information using the AWS Builder ID page. You entered this information when you signed up. When you are ready to set up MFA to use an authenticator application for signing in, you will use the AWS Builder ID page. For more information about viewing your AWS Builder ID profile, see Updating your profile (p. 14).

For more information about signing up and signing in, see Setting up CodeCatalyst (p. 7).
Source repositories

A *source repository* is where you securely store code and files for your project. It also stores the version history of your files. By default, a source repository is shared with the other users in your CodeCatalyst project. You can have more than one source repository for a project. You can create source repositories for projects in CodeCatalyst, or you can choose to link an existing source repository hosted by another service if that service is supported by an installed extension. For example, you can link a GitHub repository to a project after you install the **GitHub Repositories** extension. For more information, see [Working with source repositories in CodeCatalyst](p. 89) and [Quickstart: Using GitHub repositories in CodeCatalyst](p. 559).

Source repositories are also where configuration information is stored for your CodeCatalyst project, such as the configuration file that defines the attributes and actions of your CI/CD workflow. If you create your project using a blueprint, a source repository will be created with project configuration information stored inside it. If you create an empty project, you must create a source repository before you can create resources that require configuration information, such as workflows.

For more concepts that can help you work with source repositories and source control, see [Source repository concepts](p. 80).

Commits

A *commit* is a change to a file or set of files. In the Amazon CodeCatalyst console, a commit saves your changes and pushes them to a source repository. The commit includes information about the change, including the identity of the user who made the change, the time and date of the change, the commit title, and any message included about the change. For more information, see [Working with commits in Amazon CodeCatalyst](p. 108).

In the context of a source repository in CodeCatalyst, commits are snapshots of the changes to the contents of your repository. Every time a user commits and pushes a change, CodeCatalyst saves information that includes who committed the change, the date and time of the commit, and the changes made as part of the commit. You can also add Git tags to commits to help identify specific commits.

For more information about commits, see [Working with commits in Amazon CodeCatalyst](p. 108).

Dev Environments

A *Dev Environment* is a cloud-based development environment that you can use in CodeCatalyst to quickly work on the code stored in the source repositories of your project. The project tools and application libraries included in your Dev Environment are defined by a devfile in the source repository of your project. If you do not have a devfile in your source repository, a default devfile will be applied automatically. The default devfile includes tools for the most frequently used programming languages and frameworks. By default, a Dev Environment is configured to have a 2-core processor, 4 GB of RAM, and 16 GiB of persistent storage.

Workflows

A *workflow* is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or *actions*, to take during a workflow run. A workflow also defines the events, or *triggers*, that cause
the workflow to start. To set up a workflow, you create a workflow definition file using the CodeCatalyst console's visual or YAML editor.

**Tip**
For a quick look at how you might use workflows in a project, create a project with a blueprint. Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

For more information about workflows, see *Build, test, and deploy with workflows in CodeCatalyst* (p. 128).

**Actions**

An action is the main building block of a workflow, and defines a logical unit of work to perform during a workflow run. Typically, a workflow includes multiple actions that run sequentially or in parallel depending on how you've configured them.

For more information about actions, see *Working with actions* (p. 249).

**Issues**

An issue is a record that tracks the work related to your project. You can create an issue for a feature, a task, a bug, or any other body of work related to your project. If you're using agile development, an issue can also describe an epic or user story.

For more information about issues, see *Issues in CodeCatalyst* (p. 459).

**Personal access tokens (PATs)**

A personal access token (PAT) is similar to a password. It is associated with your user identity for use across all spaces and projects in CodeCatalyst. You use PATs to access CodeCatalyst resources that include integrated development environments (IDEs) and Git-based source repositories. PATs represent you in CodeCatalyst and you can manage them in your user settings. A user can have more than one PAT. Personal access tokens only display once. As a best practice, be sure to store them securely on your local computer. By default, PATs expire after one year.

For more information about PATs, see *Managing personal access tokens in Amazon CodeCatalyst* (p. 497).

**Roles**

A role defines a user's access to the resources for a project or a space and which actions that user can take. You choose the role for a user when you invite them to a project. There are space-level roles and project-level roles in CodeCatalyst. A user with an administrative role at the correct level can change assigned roles. For example, a user with the Project administrator role for a project has full control over that project and can change the roles of users in that project. For information about which roles are available and which permissions each role has, see *Working with roles in Amazon CodeCatalyst* (p. 476).

For more information about roles, see *Working with roles in Amazon CodeCatalyst* (p. 476).
Setting up CodeCatalyst

There are two common paths for setting up to work in Amazon CodeCatalyst: creating a space and a project as the first user, and accepting an invitation to an existing space or project. These setup workflows are necessarily quite different. The following diagram shows both sign-up processes as follows:

1. In the first case, you create and set up a space for your company, team, or group, and create a project before inviting others to these resources. An AWS account must be provided for billing purposes, where you can still default to the Free tier.
2. In the second case, if you join CodeCatalyst by accepting an invitation to a project, someone else has already created a space and project for you. However, you'll still want to configure your profile so that you're ready to start working with others.

Tip
CodeCatalyst uses spaces to group projects and resources. When you first sign up for CodeCatalyst, you'll be prompted to create a space as well as a project.

Whether you sign up to create a space and project or you sign up as part of accepting an invitation, you create an AWS Builder ID that you will use to log in to CodeCatalyst. To create an AWS Builder ID, you provide the full name, password, and email address that you use to sign in to AWS applications. You use the email and password to sign in to CodeCatalyst after this point. You can also use this AWS Builder ID to log in to other applications that use AWS Builder ID credentials.

In CodeCatalyst and in AWS Builder ID, a profile is generated based on your login information. Your profile contains your CodeCatalyst preferences for language and notification settings in your CodeCatalyst projects.

Tip
If you encounter any problems while signing up for your Amazon CodeCatalyst profile, follow the steps provided on that page. If you need additional help, see Problems signing up (p. 554).
Sign up to create your first space and your development role

You can sign up for Amazon CodeCatalyst without an invitation to an existing space or project. When you do, you will create a space and project after creating your AWS Builder ID. As part of creating a space, you will need to add an AWS account for billing purposes.

**Tip**

If you encounter any problems while signing up for your Amazon CodeCatalyst profile, follow the steps provided on that page. If you need additional help, see *Problems signing up* (p. 554).

Here is one possible flow for a user starting out with CodeCatalyst without an invitation to a project or a space.

Mary Major is a developer who is interested in CodeCatalyst and decides to try it out. She navigates to the CodeCatalyst console and chooses the option to sign up and create an AWS Builder ID. Mary provides an email address and password to create her AWS Builder ID. She will be able to use her AWS Builder ID to sign in to CodeCatalyst and other applications. When asked to choose an alias, she specifies MaryMajor as the CodeCatalyst user name that will display in CodeCatalyst and that other project members will use to @mention Mary.

Next, Mary is automatically directed to create a space. As part of this flow, Mary is asked to associate an AWS account with the space she's creating so that she can see the sample code in her first project build and deploy. She adds that information and creates her space, where she chooses the option to create a preview development role that can be used for projects in her new space. Mary chooses to create a project, and then she views a list of blueprints for projects. After reviewing the information for the available blueprints, she decides to try the Modern three-tier web application blueprint for her first project. She fills in the required fields and creates the project. As soon as the project is ready, she's taken to a project summary page that includes recent activity as well as links to project code and the workflow that automatically builds and deploys that code. She explores both the code and the workflow, including viewing the deployed sample web application. Liking what she sees, she decides to invite some of her co-workers to the project to start exploring CodeCatalyst.

When she has a moment, Mary configures her AWS Builder ID to sign in to CodeCatalyst with multi-factor authentication (MFA). With MFA configured, Mary can sign in to CodeCatalyst using a combination of her CodeCatalyst password and a passcode or token from an approved third-party authentication app.

Creating your first space and IAM roles

Follow these steps to sign up for your Amazon CodeCatalyst profile, create a space, and add an account, a support role, and a developer role for your space.

The final procedure creates and add the developer role. The developer role is an AWS IAM role that enables your CodeCatalyst workflows to access AWS resources. The developer role is a service role used to manage AWS services and will be created in the account that is signed in. A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM
Creating your first space and IAM roles

The administrator can create, modify, and delete a service role from within IAM. The role will have a name `CodeCatalystWorkflowDevelopmentRole-spaceName`. For more information about the role and role policy, see Understanding the `CodeCatalystWorkflowDevelopmentRole-spaceName` service role (p. 516).

**Note**
As a security best practice, only assign administrative access to administrative users and developers who need to manage access to AWS resources in the space.

Before you begin, you must be ready to provide an AWS account ID for an account where you have administrative privileges. Have your 12-digit AWS account ID ready. For information about finding your AWS account ID, see Your AWS account ID and its alias.

**To sign up as a new user**

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are signed in with the same AWS account that you want to use to create your space.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. On the welcome page, choose Sign up. The Create your AWS Builder ID page displays. Your AWS Builder ID is an identity you create to sign in. It is not the same as an AWS account.
4. In Your email address, enter the email address you want to associate with CodeCatalyst. Then choose Next.
5. In Your name, provide the first and last name you want displayed in applications where you use your AWS Builder ID. Spaces are allowed. This will be your AWS Builder ID profile name, such as Mary Major. You can change the name later.

Choose Next. The Email verification page displays.

6. A verification code will be sent to the email you specified. Enter this code in Verification code, and then choose Verify. If you don't receive your code after 5 minutes and cannot find it in your spam or junk folders, then choose Resend code.
7. Once we verify your code, enter a password that meets the requirements in Password and Confirm password.

Select the checkbox confirming your agreement with the AWS Customer Agreement and the AWS Service Terms, and then choose Create AWS Builder ID.

8. On the Create your CodeCatalyst alias page, enter an alias you want to use for your unique user identifier in CodeCatalyst. Choose a shortened version of your name with no spaces, such as MaryMajor. Other CodeCatalyst users will use this to @mention you in comments and pull requests. Your CodeCatalyst profile will contain both your full name from your AWS Builder ID and your CodeCatalyst alias. You cannot change your CodeCatalyst alias later.

Your full name and your alias will display in different areas in CodeCatalyst. For example, your profile name displays for your listed activity in the activity feed, but project members will use your alias to @mention you.

Choose Next. The page updates to show the Create your CodeCatalyst space section.

9. In Name your space, enter the name of your space. You cannot change this later.

   **Note**
   Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces.

Choose Next. The page updates to show the page for adding an AWS account. This account will be used as the billing account for the space.

10. In AWS Region, choose the Region where you want to store your space and project data. You cannot change this later.

11. In AWS account ID, enter the twelve-digit ID for the account you want to connect to your space.
In AWS account verification token, copy the generated token ID. The token is automatically copied for you, but you might want to store it while you approve the AWS connection request.

12. Choose Verify in AWS.

13. The Verify Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to sign in to access the page.

In the AWS Management Console, make sure to choose the same AWS Region where you want to create your space.

To directly access the page, sign in to the Amazon CodeCatalyst Spaces in the AWS Management Console at https://console.aws.amazon.com/codecatalyst/home/.

The verification token field in the AWS Management Console is automatically populated with the token generated in CodeCatalyst.

14. Choose Verify space.

An Account verified success message displays to show that the account has been added to the space.

15. Remain on the Verify Amazon CodeCatalyst space page. Choose the following link: To enable the Standard tier or add IAM roles for this space, view space details.

The CodeCatalyst space details page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to log in to access the page.

16. To turn on the Standard tier so that the space for the AWS account can update its billing tier, choose Edit. Choose the STANDARD option, and then choose Update.

Note
This does not upgrade the billing tier to the Standard tier. However, this configures the AWS account so that you can change the billing tier for your space at any time in CodeCatalyst. You can turn on the Standard tier at any time. Without making this change, the space is only able to use the Free tier.

17. Return to the CodeCatalyst page, and then choose Next.

18. A status message displays while your space is being created. When the space is created, CodeCatalyst opens the page for your space. The view defaults to the Projects tab.

Note
If a permissions error or banner is shown, then refresh the page and try to view the page again.

19. (Optional) Proceed to creating a project for your space. For a tutorial that shows you how to create a project with a blueprint, see Tutorial: Creating a project with the Modern three-tier web application blueprint (p. 17).

To create and add the CodeCatalyst CodeCatalystWorkflowDevelopmentRole-spaceName

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.

2. Open the CodeCatalyst console at https://codecatalyst.aws/.

3. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.

4. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.

5. Choose Manage roles from AWS Management Console.

The Add IAM role to Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to log in to access the page.
6. Choose Create CodeCatalyst development administrator role in IAM. This option creates a service role that contains the permissions policy and trust policy for the development role. The role will have a name CodeCatalystWorkflowDevelopmentRole-spaceName. For more information about the role and role policy, see Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role (p. 516).

   Note
   This role is only recommended for use with developer accounts and uses the AdministratorAccess AWS managed policy, giving it full access to create new policies and resources in this AWS account.

7. Choose Create development role.

8. On the connection page, under IAM roles available to CodeCatalyst, view the CodeCatalystWorkflowDevelopmentRole-spaceName role in the list of IAM roles added to your account.

9. To return to your space, choose Go to Amazon CodeCatalyst.

To create and add the CodeCatalyst AWSRoleForCodeCatalystSupport

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.

2. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.

3. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.

4. Choose Manage roles from AWS Management Console.

   The Add IAM role to Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to sign in to access the page.

5. Under CodeCatalyst space details, choose Add CodeCatalyst Support role. This option creates a service role that contains the permissions policy and trust policy for the preview development role. The role will have a name AWSRoleForCodeCatalystSupport with a unique identifier appended. For more information about the role and role policy, see Understanding the AWSRoleForCodeCatalystSupport service role (p. 517).

6. On the Add role for CodeCatalyst Support page, leave the default selected, and then choose Create role.

7. Under IAM roles available to CodeCatalyst, view the CodeCatalystWorkflowDevelopmentRole-spaceName role in the list of IAM roles added to your account.

8. To return to your space, choose Go to Amazon CodeCatalyst.

After you create your AWS Builder ID, create your first space, and add an account, you can then create a project. For more information, see Creating a project in Amazon CodeCatalyst (p. 58). If this is your first time using CodeCatalyst, we suggest starting with Tutorial: Creating a project with the Modern three-tier web application blueprint (p. 17).

Accepting an invitation and creating your AWS Builder ID

You can sign up for Amazon CodeCatalyst as part of accepting an invitation to a project or a space. As part of accepting the invitation, you'll be prompted to create an AWS Builder ID. You'll use your AWS Builder ID to access resources in CodeCatalyst.
Tip
If you need additional help, see Problems signing up (p. 554).

Here is one possible flow for a user starting out with CodeCatalyst with an invitation to a project or a space.

Saanvi Sarkar is a developer who has received an invitation to join a CodeCatalyst project as a project administrator. Saanvi accepts the invitation, which opens the sign-in page for CodeCatalyst. She chooses to sign up and provides an email address and password to create her AWS Builder ID. Saanvi will be able to use her AWS Builder ID to sign in to CodeCatalyst and other applications. Later, she can edit her profile to change her login email address or password. When asked to choose an alias, Saanvi specifies SaanviSarkar as the CodeCatalyst alias that will display in CodeCatalyst and that other project members will use to @mention Saanvi. After she has signed up, Saanvi will also be able to use her sign-in credentials for other applications that use AWS Builder ID credentials.

Upon completing sign up, Saanvi automatically joins the CodeCatalyst project and space specified in the invitation. The invitation also provides predetermined permissions for her roles in the project and space. In the project settings, Saanvi's alias shows in the members list with her assigned project role. To work with source repositories in CodeCatalyst, Saanvi takes a moment to create a personal access token (PAT). The PAT will be used in CodeCatalyst for authentication when making source changes or actions that need an authentication token.

When Saanvi works on a project, her alias will be listed in the work activity log for the project. Issues and comments by Saanvi will show her alias, where other project members are able to @mention her in replies. To @mention another project member, Saanvi looks up their alias on their CodeCatalyst profile.

When she has a moment, Saanvi configures her AWS Builder ID to sign in to CodeCatalyst with multi-factor authentication (MFA). With MFA configured, Saanvi can sign in to CodeCatalyst using a combination of her CodeCatalyst password and a passcode or token from an approved third-party authentication app.

Accepting an invitation and creating an AWS Builder ID

When you're invited to a project or space in Amazon CodeCatalyst, you'll receive an email from notify@codecatalyst.aws asking you to accept the invitation. If you already have a AWS Builder ID and are signed in to CodeCatalyst, choosing Accept invitation will automatically open the project or space in a browser tab. If you're not signed in to the console but have a AWS Builder ID, you'll be taken to the sign-in page. For more information, see Sign in as a returning user (p. 13).

If you don't have a AWS Builder ID, choosing Accept invitation will take you to the sign-in page, where you should choose the option to create your AWS Builder ID.

To accept an invitation and create a AWS Builder ID

1. In the invitation email, choose Accept invitation.
2. On the sign in page, choose Not signed up? Create your AWS Builder ID.
   Tip
   Your AWS Builder ID is an identity you create to sign in. It is not the same as an AWS account.
3. On the Create your AWS Builder ID page, in Email address, enter the email address you want to use for your AWS Builder ID.

   In Your name, provide the first and last name you want displayed in applications where you use your AWS Builder ID. Spaces are allowed. This will be your AWS Builder ID profile name, such as Mary Major. You can change the name later.
Choose Next.

A verification code will be sent to the email you specified. Enter this code in Verification code, and then choose Verify. If you don’t receive your code after 5 minutes and cannot find it in your spam or junk folders, then choose Resend code.

4. Once your code is verified, enter a password that meets the requirements in Password and Confirm password.

5. Choose Create AWS Builder ID.

6. On the Create your alias page, enter an alias you want to use for your unique user identifier in CodeCatalyst. Choose a shortened version of your name with no spaces, such as MaryMajor. Other CodeCatalyst users will use this to @mention you in comments and pull requests. Your CodeCatalyst profile will contain both your full name from your AWS Builder ID and your CodeCatalyst alias. You cannot change your CodeCatalyst alias.

Your full name and your alias will display in different areas in CodeCatalyst. For example, your profile name displays for your listed activity in the activity feed, but project members will use your alias to @mention you.

Choose Create alias. You’ll be taken to the project or space you were invited to.

Sign in as a returning user

Follow these steps to sign in to your Amazon CodeCatalyst profile.

Note

Have you registered a device for multi-factor authentication (MFA) yet? We strongly recommend that you configure MFA in Amazon CodeCatalyst to increase your security. For more information, see How to register a device for use with multi-factor authentication (p. 501).

To sign in as a returning user

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Enter your Email address. Optionally, choose Save my email address if you want to save your email address for future sign-ins. Choose Continue.
3. Enter your Password. Choose Sign in. If you don’t remember your password, follow the steps in I forgot my password (p. 582).

Trusted devices

After you choose the option This is a trusted device from the sign-in page, Amazon CodeCatalyst considers all future sign-ins from that device as authorized. Amazon CodeCatalyst will not present an option to enter an MFA code as long as you use that trusted device. Some exceptions include signing in from a new browser or when your device has been issued an unknown IP address.

Viewing and managing CodeCatalyst profiles

You can view user profiles in Amazon CodeCatalyst to get information such as email addresses and CodeCatalyst aliases. You can also update your profile and your AWS Builder ID. If you forget your password, you can request a password reset.
Viewing your CodeCatalyst profile

You provide information at signup that will be used as your credentials to log in to Amazon CodeCatalyst and that will be managed in your profile. This includes your Name, Nickname, and the Email address you use to sign in to CodeCatalyst.

Note
The AWS Builder ID Nickname is not your CodeCatalyst alias. You selected your CodeCatalyst alias at signup.

To view your CodeCatalyst profile

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose My settings. The CodeCatalyst My settings page opens.
3. To update your AWS Builder ID email address or password, or to set up MFA, choose Manage AWS Builder ID. The AWS Builder ID page opens.

Viewing another user's CodeCatalyst profile

To view another user's CodeCatalyst profile

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. On the side navigation, choose Project settings. Choose the Members tab. View the list of members for your CodeCatalyst project.
3. Choose the member name that you want to look up or @mention. The My settings page shows the user's alias, email address, and full name. Use the CodeCatalyst alias to @mention project members.

Note
A user's AWS Builder ID Nickname is not their CodeCatalyst alias. They selected their CodeCatalyst alias at signup.

To view another user's profile in your project, choose their name in the list.

Updating your profile

In CodeCatalyst, your profile consists of personal information managed by AWS Builder ID and settings managed in CodeCatalyst.

- Your profile's full name, email address, and password are managed by AWS Builder ID. You entered this information when you signed up. When you set up MFA to use an authenticator app for application sign-in, CodeCatalyst takes you to the AWS Builder ID page.
- CodeCatalyst settings for your personal access token (PAT), CodeCatalyst notifications, and language preferences are managed in the My settings page in CodeCatalyst. For more information, see Managing personal access tokens in Amazon CodeCatalyst (p. 497).

Note
You can update your AWS Builder ID full name (CodeCatalyst display name) and first name. However, you cannot change your CodeCatalyst alias.
Updating your AWS Builder ID or email address

To update your AWS Builder ID or email address

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose My settings. The CodeCatalyst My settings page opens.
3. On the profile page, choose Manage AWS Builder ID. The AWS Builder ID page opens.
4. On the left side of the page, choose My details.
5. Under Profile information, choose Edit to update your Name or Nickname. If you did not specify a nickname, the Nickname field reflects the first name in the full name. It is not your CodeCatalyst alias.
   
   **Note**
   
   This updates the AWS Builder ID full name and first name. This does not update your CodeCatalyst alias.

Under Contact information, choose Edit to update your Email address.

**Note**

This updates the email address you will use to sign in to CodeCatalyst.

Changing your CodeCatalyst password

To change your CodeCatalyst password

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose User profile. The CodeCatalyst My settings page opens.
3. On the profile page, choose Manage AWS Builder ID. The AWS Builder ID page opens.
4. On the left side of the page, choose Security.
5. Choose Change password and follow the instructions.

Setting up to use the AWS CLI with CodeCatalyst

The Amazon CodeCatalyst console is where you'll work on most of your daily tasks. However, you might want to set up and configure the AWS CLI when you're working with Dev Environments, personal access tokens, or logs of events in CodeCatalyst. You must install the AWS CLI and configure a profile before you can use it with CodeCatalyst.

To set up the AWS CLI for CodeCatalyst

1. Install the latest version of the AWS CLI. If you already have a version of the AWS CLI installed, make sure that it is recent and includes commands for CodeCatalyst, and update it if needed. To verify that you have a version installed that includes CodeCatalyst commands, open a command prompt and run the following command:

   ```bash
   aws codecatalyst help
   ```

   If you see a list of CodeCatalyst commands, you have a version that supports CodeCatalyst. If the command is not recognized, update your version of the AWS CLI to the latest version. For more
information, see Installing or updating the latest version of the AWS CLI in the AWS Command Line Interface User Guide.

2. Run the `aws configure` command to create a profile if you don't have one or if you want to use a named profile specifically for CodeCatalyst. We recommend creating a named profile to use specifically with CodeCatalyst, but you can also use the default profile. For more information, see Configuration basics.

3. Edit the config file for the profile to add a section for connecting to CodeCatalyst as follows. The config file is located at `~/.aws/config` on Linux or macOS, or at `C:\Users\USERNAME\aws\config` on Windows.

```
[profile codecatalyst]
region = us-west-2
sso_session = codecatalyst

[sso-session codecatalyst]
sso_region = us-east-1
sso_start_url = https://view.awsapps.com/start
sso_registration_scopes = codecatalyst:read_write
```

4. Save the file.

5. Before attempting to run any CodeCatalyst commands, open a new terminal or command prompt and run the following command to request and retrieve credentials to run `aws codecatalyst` commands. Replace `codecatalyst` with the name of your profile if needed.

```
aws sso login --profile codecatalyst
```

To view examples of `codecatalyst` commands, see the following topics:

- Managing personal access tokens in Amazon CodeCatalyst (p. 497)
- Accessing logged events in CodeCatalyst (p. 551)
Getting started tutorials

Amazon CodeCatalyst provides a number of different templates to help you get started with your projects. You can also choose to start with an empty project and add resources to it. Follow the steps in these tutorials to learn some of the ways you can work in CodeCatalyst.

If this is your first time using CodeCatalyst, we suggest starting with Tutorial: Creating a project with the Modern three-tier web application blueprint (p. 17).

**Note**
In order to follow these tutorials, you must first complete setting up. For more information, see Setting up CodeCatalyst (p. 7).

**Topics**
- Tutorial: Creating a project with the Modern three-tier web application blueprint (p. 17)
- Tutorial: Starting with an empty project and manually adding resources (p. 29)

For additional tutorials that focus on specific functional areas in CodeCatalyst, see:
- Getting started with Slack notifications (p. 69)
- Getting started with CodeCatalyst source repositories and the Single-page application blueprint (p. 83)
- Getting started with workflows in CodeCatalyst (p. 134)
- Get started with the Amazon CodeCatalyst action developer guide

For in-depth tutorials, see:
- Tutorial: Upload artifacts to Amazon S3 (p. 140)
- Tutorial: Deploy a serverless application using AWS CloudFormation (p. 165)
- Tutorial: Deploy an application to Amazon ECS (p. 183)
- Tutorial: Deploy an application to Amazon EKS (p. 207)
- Tutorial: Lint code using a GitHub Action (p. 268)

Tutorial: Creating a project with the Modern three-tier web application blueprint

You can get started more quickly with developing software by creating a project with a blueprint. A project created with a blueprint includes the resources that you need, including a source repository to manage your code, and a workflow to build and deploy the application. In this tutorial, we will walk you through using the Modern three-tier web application blueprint to create a project in Amazon CodeCatalyst. The tutorial also includes viewing the deployed sample, inviting other users to work on it, and making changes to the code with pull requests that are automatically built and deployed to resources in the connected AWS account when the pull request is merged. Where CodeCatalyst creates
your project with reports, activity feeds, and other tools, your blueprint creates AWS resources in the AWS account associated with your project. Your blueprint files allow you to build and test a sample modern application and deploy it to infrastructure in the AWS Cloud.

The following illustration shows how tools in CodeCatalyst are used to create an issue for tracking, merge and automatically build the change, and then start a workflow in the CodeCatalyst project that runs actions to allow AWS CDK and AWS CloudFormation to provision your infrastructure.

The actions generate resources in the associated AWS account and deploy your application to a serverless AWS Lambda function with an API Gateway endpoint. The AWS Cloud Development Kit (AWS CDK) action converts one or more AWS CDK stacks to AWS CloudFormation templates and deploys stacks to your AWS account. Resources in your stacks include Amazon CloudFront resources to distribute dynamic web content, an Amazon DynamoDB instance for your application data, and the roles and policies that support the deployed application.

When you create a project with the **Modern three-tier web application** blueprint, your project is created with the following resources:

**In the CodeCatalyst project:**

- A source repository (p. 79) with sample code and workflow YAML
- A workflow (p. 128) that builds and deploys the sample code whenever a change is made to the default branch
- An issues board and backlog that you can use to plan and track work
- A test reports suite with automated reports included in the sample code

**In the associated AWS account:**
Prerequisites

To create a modern application project in this tutorial, you must have completed the tasks in Setting up CodeCatalyst (p. 7) as follows:

- Have an AWS Builder ID for signing in to CodeCatalyst.
- Belong to a space and have the Space administrator or Power user role assigned to you in that space. For more information, see Creating additional spaces in CodeCatalyst (p. 53), Viewing members in a space (p. 47), and Space administrator role (p. 477).
- Have an AWS account associated with your space and have the IAM role you created during sign-up. For example, during sign-up, you have the option to choose to create a service role with a role policy called the CodeCatalystWorkflowDevelopmentRole-spaceName role policy. The role will have a name CodeCatalystWorkflowDevelopmentRole-spaceName with a unique identifier appended. For more information about the role and role policy, see Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role (p. 516). For the steps to create the role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517).

Topics

- Prerequisites (p. 19)
- Step 1: Create the Modern three-tier web application project (p. 20)
- Step 2: Invite someone to your project (p. 20)
- Step 3: Create issues to collaborate on and track work (p. 21)
- Step 4: View your source repository (p. 21)
- Step 5: Create a Dev Environment with a test branch and make a quick code change (p. 22)
- Step 6: View the workflow that builds the modern application (p. 23)
- Step 7: Ask others to review your changes (p. 25)
- Step 8: Close the issue (p. 27)
- Step 9: Clean up resources (p. 27)
- Reference (p. 28)
Step 1: Create the Modern three-tier web application project

After you've created it, your project is where you will develop and test code, coordinate development tasks, and view project metrics. Your project also contains your development tools and resources.

In this tutorial, you will use the Modern three-tier web application blueprint to create an interactive application. The workflow that is created and run automatically as part of your project will build and deploy the application. The workflow only runs successfully after all roles and account information is configured for your space. After the workflow has run successfully, you can visit the endpoint URL to see the application.

To create a project with a blueprint

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the CodeCatalyst console, navigate to the space where you want to create a project.
3. Choose Create project.
4. On the Create project page, choose Start with a blueprint.
5. Choose the Modern three-tier web application blueprint, and then choose Next. You can also search on "modern" to find the blueprint more quickly.
6. As you configure your selections, you can preview the project configuration under Generate project preview. Choose View code for a preview of the project repository or repositories and the sample application code that will be generated in the project source. Choose View workflow to preview the definition file for your CI/CD workflow. When you choose the preview, the preview page will dynamically update to show definition files with updates based on your selections.
7. In Name your project, enter the name that you want to assign to your project and its associated resource names. For example, MyExampleProject.

   Note
   The name must be unique in your space. If MyExampleProject is already being used, you can enter a different project name.

8. In Account, choose the account you added during sign-up to associate it with your space. The field for the role displays.
9. In Deployment Role, choose the role that you created and added to your account during sign-up. For example, choose CodeCatalystWorkflowDevelopmentRole-spaceName.

   If there are no roles available to choose, then you must add the IAM role. To add the role, choose Add IAM role to add the role to the account in AWS. For more information about adding IAM roles, see Managing AWS accounts (p. 36).
10. In Compute platform, choose Lambda.
11. In Region, choose the AWS Region where you want to create resources for deployments.
12. In Application name, leave the default.
13. Choose Create project.

The project workflow starts as soon as you create the project. It will take a little time to finish building and deploying the code. In the meantime, go ahead and invite someone else to your project.

Step 2: Invite someone to your project

Now that you've set up your project, invite others to work with you.
To invite someone to your project

1. Navigate to the project to which you want to invite users.
2. In the navigation pane, choose Project settings.
3. On the Members tab, choose Invite new member.
4. Type the email addresses of the people you want to invite as users for your project. You can type multiple email addresses separated by a space or a comma. You can also choose from members of your space who are not members of the project.
5. Choose the role for the user.

When you have finished adding users, choose Send.

Step 3: Create issues to collaborate on and track work

CodeCatalyst helps you track features, tasks, bugs, and any other work involved in your project with issues. You can create issues to track needed work and ideas. By default, when you create an issue it is added to your backlog. You can move issues to a board where you track work in progress. You can also assign an issue to a specific project member.

To create an issue for a project

1. In the navigation pane, choose Issues.
2. Choose Create issue.
3. In Issue title, provide a name for the issue. Optionally, provide a description of the issue. In this example, use make a change in the src/mysfit_data.json file.
4. Choose the priority, estimate, status, and labels. Under assignee, choose +Add me to assign the issue to yourself.
5. Choose Create issue. The issue is now visible on the board. Choose the card to move the issue to the In progress column.

For more information, see Issues in CodeCatalyst (p. 459).

Step 4: View your source repository

Your project comes with a source repository that contains files to define and support your application or service. The contents of the source repository include the following:

- The .codecatalyst directory contains the YAML workflow configuration file for each workflow included in the blueprint. Configuration files define the workflows that run when you make a code change and are stored in your source repository.
- The cdkStacks directory contains the AWS CDK stack files that define the infrastructure in the cloud.
- The src directory contains the application source code. This includes the CDK assets folder, which contains sample code and other assets for generating the application each time you build.
- The tests directory contains files for the integ and unit tests that are run as part of the automated CI/CD workflow that runs when you build and test your application.
- The web directory contains the frontend source code. Other files include project files such as the package.json file that contains important metadata about your project, the index.html page for the website, and the Dockerfile for the container. Other files are used for app generation, such as .eslintrc.fjs and tsconfig.json. This directory also includes the README.md file with configuration information for the project.
To navigate to the source repositories for a project

1. Navigate to your project, and do one of the following:
   - On the summary page for your project, choose the repository you want from the list, and then choose View repository.
   - In the navigation pane, choose Code, and then choose Source repositories. In Source repositories, choose the name of the repository from the list. You can filter the list of repositories by typing part of the repository name in the filter bar.

2. On the home page for the repository, view the contents of the repository and information about the associated resources such as the number of pull requests and workflows. By default, the contents for the default branch are shown. You can change the view by choosing a different branch from the drop-down list.

Step 5: Create a Dev Environment with a test branch and make a quick code change

You can quickly work on the code in your source repository by creating a Dev Environment. For this tutorial, we recommend that you create a Dev Environment using AWS Cloud9, and choose the option to create a branch from the main branch when creating the Dev Environment. We use the name test for this branch, but you can enter a different branch name if you prefer. In a later step, you will use the Dev Environment to make a code change and create a pull request.

To create a Dev Environment with a new branch

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to create a Dev Environment.
3. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, choose Source repositories, and choose the repository for which you want to create a Dev Environment.
4. On the repository home page, choose Create Dev Environment.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Choose the repository to clone, choose Work in new branch, enter a branch name into the Branch name field, and choose a branch off of which to create the new branch from the Create branch from drop-down menu.
8. Optionally, choose the Dev Environment configuration edit button to edit the Dev Environment's compute, storage, or timeout configuration.
9. Choose Create. While your Dev Environment is being created, the Dev Environment status column will display Starting, and the status column will display Running once the Dev Environment has been created. A new tab will open with your Dev Environment in the IDE of your choice. You can edit code and commit and push your changes.

In this section, you will work with your generated sample application in CodeCatalyst by making changes to the code with pull requests that are automatically built and deployed to resources in the connected AWS account when the pull request is merged.

To make a change in your src/mysfit_data.json file

1. Navigate to your project Dev Environment. In AWS Cloud9, expand the side navigation menu to browse the files. Expand src and open src/mysfit_data.json.
Step 6: View the workflow that builds the modern application

After creating the modern application project, CodeCatalyst generates several resources on your behalf, including a workflow. A workflow is an automated procedure defined in a .yaml file that describes how to build, test, and deploy your code.

In this tutorial, CodeCatalyst created a workflow and started it automatically when you created your project. (The workflow might still be running depending on how long ago you created your project.) Use the following procedures to check on the workflow's progress, review the generated logs and test reports, and finally, navigate to the URL of the deployed application.

To check on the workflow progress

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose ApplicationDeploymentPipeline. This is the workflow that the CodeCatalyst blueprint generated and started when you created your project.

A workflow diagram appears. The diagram shows several blocks, each one representing a task or action. These blocks are arranged vertically, with the blocks at the top running before the ones below. The blocks are:

- **WorkflowSource** – This block represents your source repository. It shows the ID of the commit that automatically started the workflow run. It also shows the source repository (`mysfits<unique-ID>`) and branch (`main`) to which the commit was pushed. CodeCatalyst generated this commit when you created your project.
- **Build** – This block represents a grouping of two actions: `cdk_bootstrap` and `build_backend`. These actions must both complete successfully for the next action to start.
• **cdk_bootstrap** – This block represents an action that generates the cloud infrastructure that will host application deployments. The action is able to access AWS using the CodeCatalystWorkflowDevelopmentRole-`spaceName` role you created earlier.

• **build_backend** – This block represents an action that builds the application's backend components into AWS.

• **Deploy_Backend** – This block represents an action that deploys the application's backend components.

• **Tests** – This block represents a grouping of two test actions, described in the following bullets.

  • **integration_tests** – This block represents an action that runs tests on the application components.

  • **build_and_test_frontend** – This block represents an action that builds the application's frontend components into AWS.

• **Deploy_Frontend** – This block represents an action that deploys the application's frontend components.

3. Choose **YAML** to view the workflow definition file (p. 354). The file has the following noteworthy sections:

  • A `Triggers` section, at the top. This section indicates that the workflow must start whenever code is pushed to the source repository's `main` branch.

  • An `Actions` section, under `Triggers`. This section defines the actions, such as the `cdk_bootstrap` action, that you saw in the workflow diagram.

4. Choose **Visual** and choose any action in the workflow diagram.

5. Choose the **Configuration** tab to see the configuration settings for the action. Each configuration setting has a matching property in the workflow definition file.

6. Leave the console open and go to the next procedure.

**To review the build logs and test reports**

1. Choose the **Latest state** tab.

2. In the workflow diagram, choose the **deploy_frontend** action.

3. Wait for the action to finish. Watch for the "in-progress" icon (👀) to change to a "success" icon (✔).

   **Tip**
   The build takes several minutes to complete. While you wait, consider working on another part of this tutorial.

4. Choose your run, and then choose the **Visual** tab.

5. Choose the **build_frontend** action. In **Logs**, expand the `cd web && npm install` and `npm run build-only` sections to view the log messages for these steps. You can see messages related to the creation of the frontend application.

6. Choose the **deploy_frontend** action. In **Logs**, expand the **deploy_frontend** section to view the log messages for these steps. You can see messages related to the creation of the frontend stacks.

7. Choose the **Reports** tab for your workflow, and then choose the `AutoDiscovered-coverage.xml` report. CodeCatalyst displays the associated report on a separate page. The report shows the code coverage tests that were run, and indicates the proportion of lines of code that were successfully validated by testing, such as 80%.

   For more information about test reports, see **Testing using workflows in CodeCatalyst** (p. 146).

   **Tip**
   You can also view your test reports by choosing **Reports** in the navigation pane.

8. Leave the CodeCatalyst console open, and go to the next procedure.
To confirm that the modern application was deployed successfully

1. Return to the ApplicationDeploymentPipeline workflow, and choose the Deploy_Frontend action.
2. Choose the Variables tab.
3. Copy the endpointUrl value into another browser tab.
   
   A website appears with the home page opened to the list of Mysfits you can adopt.
4. To search for the pegasus Mysfit named Twilight Glitter, choose Good and Lawful. Note the value for the age. You will make a code change to update the age.

Step 7: Ask others to review your changes

Now that you have changes in a branch named test, you can ask others to review them by creating a pull request. Perform the following steps to create a pull request to compare the changes in the test branch with the main branch.

To create a pull request

1. Navigate to your project.
2. Do one of the following:
   • In the navigation pane, choose Code, choose Pull requests, and then choose Create pull request.
   • On the repository home page, choose More, and then choose Create pull request.
   • On the project page, choose Create pull request.
3. In Source repository, make sure that the specified source repository is the one that contains the committed code. This option only appears if you did not create the pull request from the repository's main page.
4. In Destination branch, choose the branch to merge the code into after it is reviewed.
5. In Source branch, choose the branch that contains the committed code.
6. In Pull request title, enter a title that helps other users understand what needs to be reviewed and why.
7. (Optional) In Pull request description, provide information such as a link to issues or a description of your changes.
8. (Optional) In Issues, choose Link issues, and then either choose an issue from the list or enter its ID. To unlink an issue, choose the unlink icon.
9. (Optional) In Required reviewers, choose Add required reviewers. Choose from the list of project members to add them. Required reviewers must approve the changes before the pull request can be merged into the destination branch.

   **Note**
   You cannot add a reviewer as both a required reviewer and an optional reviewer. You cannot add yourself as a reviewer.
10. (Optional) In Optional reviewers, choose Add optional reviewers. Choose from the list of project members to add them. Optional reviewers do not have to approve the changes as a requirement before the pull request can be merged into the destination branch.
11. Review the differences between the branches. The difference displayed in a pull request is the changes between the revision in the source branch and the merge base, which is the head commit of the destination branch at the time the pull request was created. If no changes display, the branches might be identical, or you might have chosen the same branch for both the source and the destination.
12. When you are satisfied that the pull request contains the code and changes you want reviewed, choose Create.
Note
After you create the pull request, you can add comments. Comments can be added to the pull request or to individual lines in files as well as to the overall pull request. You can add links to resources, such as files, by using the @ sign followed by the name of the file.

When you create the pull request, the OnPullRequest workflow starts. While your reviewers are approving your code change, you can observe the results by choosing the workflow and viewing the test output.

After you've had the change reviewed, you can merge the code. Merging the code to the default branch will automatically start the workflow that will build and deploy your changes.

To merge a pull request from the CodeCatalyst console

1. Navigate to your modern application project.
2. On the project page, under Open pull requests, choose the pull request you want to merge. If you do not see the pull request, choose View all pull requests and then choose it from the list. Choose Merge.
3. Choose from the available merge strategies for the pull request. Optionally select or deselect the option to delete the source branch after merging the pull request, and then choose Merge.

   Note
   If the Merge button is not active, or you see the Not mergeable label, either one or more required reviewers have not yet approved the pull request, or the pull request cannot be merged in the CodeCatalyst console. A reviewer who has not approved a pull request is indicated by a clock icon in Overview in the Pull request details area. If all required reviewers have approved the pull request but the Merge button is still not active, you might have a merge conflict. You can resolve merge conflicts for the destination branch in the CodeCatalyst console and then merge the pull request, or you can resolve conflicts and merge locally, and then push the commit that contains the merge to CodeCatalyst. For more information, see Merging a pull request (Git) (p. 107) and your Git documentation.
4. In Merge pull request, review the summary information and the merge strategy that will be used. Choose Merge to merge the pull request.

Once you've merged the changes from the test branch into the main branch, the change automatically starts the ApplicationDeploymentPipeline workflow that builds and deploys your change.

To see the merged commit run through the ApplicationDeploymentPipeline workflow

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. In Workflows, in ApplicationDeploymentPipeline, expand Show most recent runs. You can see the workflow run started by the merge commit. Optionally choose it to watch the run progress.
3. When the run completes, reload the URL you visited earlier. View the pegasus to verify that the age changed.
Step 8: Close the issue

When an issue is resolved, it can be closed on the CodeCatalyst console.

To close an issue for a project

1. Navigate to your project.
2. In the navigation pane, choose Issues.
3. Choose the issue that you want to close.
4. Choose Done from the Status dropdown menu.

For more information, see Issues in CodeCatalyst (p. 459).

Step 9: Clean up resources

Clean up in CodeCatalyst and AWS to remove traces of this tutorial from your environment.

You can choose to keep using the project you used for this tutorial, or you can delete the project and its associated resources.

**Note**
Deleting this project will delete all repositories, issues, and artifacts in the project for all members.

To delete a project

1. Navigate to your project, and then choose Project settings.
2. Choose the General tab.
3. Under the project name, choose Delete project.
To delete resource stacks in AWS CloudFormation

1. Sign in to the AWS Management Console with the same account that you added to your space, and then choose AWS CloudFormation.
2. Choose the stack to delete, and then choose Delete. To confirm, choose Delete stack.
3. Repeat these steps for each stack.

Reference

As a reference, you can learn more about the AWS resources that are created as part of this tutorial.

Resources for this project will be created by the automated resources generated in the stacks created for your project as follows:

- The `cdk_bootstrap` action generates the `CDKToolkit` stack in the associated account.

  Deploying stacks with the AWS CDK requires dedicated Amazon S3 buckets and other containers to be available to AWS CloudFormation during deployment. The `cdk_bootstrap` action provisions these resources for the AWS CDK.

  **Note**
  
  Note that the AWS CDK stack is not torn down and recreated for each deployment. This is a stack that is initiated in each account to support the AWS CDK.

  The `CDKToolkit` stack defines the following resources for the Modern three-tier web application blueprint:
  
  - An Amazon S3 staging bucket, bucket policy, and the AWS KMS key used to encrypt the bucket.
  - An IAM deployment role for the deploy action.
  - AWS IAM roles and policies in support of the resources in the stack.

- The `deploy_backend` action generates the `mysfits<ID>AppStack` stack. The `mysfits<ID>AppStack` stack defines the following resources for the Modern three-tier web application blueprint:

  - An Amazon API Gateway endpoint.
  - AWS IAM roles and policies in support of the resources in the stack.
  - An AWS Lambda function and layer provides the serverless compute platform for the modern application.
  - An IAM policy and role for the bucket deployment and Lambda function.

- The `deploy_frontend` action generates the `mysfits<ID>WebStack` stack. The `mysfits<ID>WebStack` stack defines the following resources for the Modern three-tier web application blueprint:

  - AWS IAM roles and policies in support of the resources in the stack.
  - An Amazon CloudFront distribution and access identity.
  - An AWS CDK custom bucket deployment.

See also

For more information about the resources that are created as part of this tutorial, see the following:

- Amazon S3 – A resource to host your frontend assets on an object storage service offering industry-leading scalability, data high availability, security, and performance. For more information, see Amazon S3 User Guide.
- Amazon CloudFront – A resource for speeding up distribution of the application's frontend content such as .html, .css, .js, and image files to your users. For more information, see Amazon CloudFront Developer Guide.
• Amazon API Gateway – A resource for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. For more information, see API Gateway Developer Guide.
• AWS Cloud Development Kit (AWS CDK) – The AWS CDK includes the CDK Toolkit, a command line tool for working with your AWS CDK apps and stacks. The Toolkit converts AWS CDK stacks to AWS CloudFormation templates and related assets, and then deploys your stacks to an AWS account. For more information, see AWS Cloud Development Kit (AWS CDK) Developer Guide.
• AWS Lambda – A resource to invoke your code on a high availability compute infrastructure without provisioning or managing servers. For more information, see AWS Lambda Developer Guide.
• AWS IAM – A resource for securely controlled access to AWS and its resources. For more information, see IAM User Guide.

Tutorial: Starting with an empty project and manually adding resources

You can create an empty project without any predefined resources inside it by choosing the Empty project blueprint when you create the project. After you create an empty project, you can create and add resources to it according to your project needs. Because projects created without a blueprint are empty on creation, this option requires more knowledge of creating and configuring CodeCatalyst resources to get started.

Topics
• Prerequisites (p. 29)
• Create an empty project (p. 29)
• Create a source repository (p. 30)
• Create a workflow to build, test, and deploy a code change (p. 31)
• Invite someone to your project (p. 31)
• Create issues to collaborate on and track work (p. 31)

Prerequisites

To create an empty project, you must have the Space administrator or Power user role assigned to you. If this is your first time signing in to CodeCatalyst, see Setting up CodeCatalyst (p. 7).

Create an empty project

Creating a project is the first step in being able to work together. If you want to create your own resources, such as source repositories and workflows, you can start with an empty project.

To create an empty project

1. Navigate to the space where you want to create a project.
2. On the space dashboard, choose Create project.
3. Choose Start from scratch.
4. Under Give a name to your project, enter the name that you want to assign to your project. The name must be unique within your space.
5. Choose Create project.

Now that you have an empty project, the next step is to create a source repository.
Create a source repository

Create a source repository to store and collaborate on your project's code. Project members can clone this repository to their local computers to work on code. Alternatively, you can choose to link a repository hosted in a supported service, but that is not covered in this tutorial. For more information, see Linking a source repository (p. 90).

To create a source repository

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Create repository.
5. In Repository name, provide a name for the repository. In this guide, we use codecatalyst-source-repository, but you can choose a different name. Repository names must be unique within a project. For more information about requirements for repository names, see Quotas for source repositories in CodeCatalyst (p. 110).
6. (Optional) In Description, add a description for the repository that will help other users in the project understand what the repository is used for.
7. (Optional) Add a .gitignore file for the type of code you plan to push.
8. Choose Create.

Note
CodeCatalyst adds a README.md file to your repository when you create it. CodeCatalyst also creates an initial commit for the repository in a default branch named main. You can edit or delete the README.md file, but you can't change or delete the default branch.

You can quickly add code in your repository by creating a Dev Environment. For this tutorial, we recommend that you create a Dev Environment using AWS Cloud9, and choose the option to create a branch from the main branch when creating the Dev Environment. We use the name test for this branch, but you can enter a different branch name if you prefer.

To create a Dev Environment with a new branch

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to create a Dev Environment.
3. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, choose Source repositories, and choose the repository for which you want to create a Dev Environment.
4. On the repository home page, choose Create Dev Environment.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Choose the repository to clone, choose Work in new branch, enter a branch name into the Branch name field, and choose a branch off of which to create the new branch from the Create branch from drop-down menu.
8. Optionally, choose the Dev Environment configuration edit button to edit the Dev Environment's compute, storage, or timeout configuration.
9. Choose Create. While your Dev Environment is being created, the Dev Environment status column will display Starting, and the status column will display Running once the Dev Environment has been created. A new tab will open with your Dev Environment in the IDE of your choice. You can edit code and commit and push your changes.
Create a workflow to build, test, and deploy a code change

In CodeCatalyst, you organize the building, testing, and deployment of your applications or services in workflows. Workflows consist of actions and can be configured to run automatically after specified source repository events occur, such as code pushes or opening or updating a pull request. For more information about workflows, see Build, test, and deploy with workflows (p. 128).

Follow the instructions in Getting started with workflows in CodeCatalyst (p. 134) to create your first workflow.

Invite someone to your project

Now that you've set up your custom project, invite others to work with you.

To invite someone to your project

1. Navigate to the project to which you want to invite users.
2. In the navigation pane, choose Project settings.
3. On the Members tab, choose Invite new member.
4. Type the email addresses of the people you want to invite as users for your project. You can type multiple email addresses separated by a space or a comma. You can also choose from members of your space who are not members of the project.
5. Choose the role for the user.

When you have finished adding users, choose Send.

Create issues to collaborate on and track work

CodeCatalyst helps you track features, tasks, bugs, and any other work involved in your project with issues. You can create issues to track needed work and ideas. By default, when you create an issue it is added to your backlog. You can move issues to a board where you track work in progress. You can also assign an issue to a specific project member.

To create an issue for a project

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
   Make sure that you are navigating in the project where you want to create issues. To view all projects, in the navigation pane, choose Amazon CodeCatalyst, and if needed, choose View all projects. Choose the project where you want to create or work with issues.
2. In the navigation pane, choose Track, and then choose Backlog.
3. Choose Create issue.
4. In Issue title, provide a name for the issue. Optionally provide a description of the issue. Choose the status, priority, and estimate for the issue if desired. You can also assign the issue to a project member from the list of project members.
5. Choose Save.

After you have created issues, you can assign them to project members, estimate them, and track them on a Kanban board. For more information, see Issues in CodeCatalyst (p. 459).
Spaces in CodeCatalyst

You create a space that represents you, your company, department, or group, and provides a place where your development teams can manage projects. You must create a space to add projects, members, and the associated cloud resources you create in Amazon CodeCatalyst.

**Note**
Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces.

When you create a space, you are automatically assigned the **Space administrator** role. You can add this role to other users in the space.

With the **Space administrator** role, you can manage the space as follows:

- Add other space administrators to the space
- Change member roles and permissions
- Edit or delete the space
- Create projects and invite members to the project
- View a list of all projects in the space
- View the activity feed for all projects in the space

When you create a space, you are automatically added to the space with two roles: the **Space administrator** role, and the **Project administrator** role for the project you created as part of creating the space. Additional users are added as members to the space automatically when they accept invitations to projects. This membership in the space does not grant any permissions in the space. What users can do in a space is determined by the role the user has in a specific project.

CodeCatalyst projects are managed at the project level by a Space administrator or a Project administrator. After the Space administrator creates a project, the Space administrator and the Project administrator can manage the project, edit the project information, and invite members to the project from the space or external to the space.

A space represents your company, department, or group. You must create a space to add projects, members, and the associated cloud resources you create in CodeCatalyst. When a user accepts an invitation to a project, CodeCatalyst automatically adds them to the space that contains that project. Users with the **Space administrator** role have permissions to manage the space.

For more information about roles, see [Working with roles in Amazon CodeCatalyst](p. 476).
Account connections for billing – Billing for CodeCatalyst is provided by AWS and charges, if applicable, are applied to a specified AWS account. Even when you’re using the CodeCatalyst Free tier, you must add an AWS account for billing. For more information, see Billing (p. 608).

AWS is the billing provider for CodeCatalyst accounts. The account you specify will show CodeCatalyst charges on your AWS bill. Only one AWS account can be used as a billing account for a space in CodeCatalyst. If an account is already used for a space, you must use a different billing account for any additional spaces.

Account connections for deployments – A service role is required for workflows to deploy successfully. If your project workflows contain CodeCatalyst actions that require AWS resources, CodeCatalyst uses a combination of AWS account, IAM role, and CodeCatalyst environment to access resources as follows:

- **Environments** are used to track deployment activity for a CodeCatalyst project. Environments also provide a link between IAM roles and workflow actions so that the actions can authenticate to, and operate in, AWS services.

  **Note**
  All workflow actions that require IAM roles, including build actions, must use a CodeCatalyst environment in order to run.

- **AWS accounts** provide a link between the CodeCatalyst environment and the AWS IAM roles and policies used as a service role.

- **IAM roles** and role policies provide the permissions to perform actions within CodeCatalyst workflows.
The following diagram provides a functional overview of connections between your AWS account and CodeCatalyst. In this example, AWS account 1 has been added to the space and set up for billing. AWS account 2 and AWS account 3 are added to the space and available to CodeCatalyst projects in the space. AWS account 2 and AWS account 3 each connect to a project that requires IAM roles for CodeCatalyst actions that use AWS resources. One project does not require IAM roles and so does not require a connection to an account.

The following are additional considerations for added accounts:

- There is a one-to-one mapping of account connection to AWS account for a space. A single AWS account can be added to multiple different spaces. AWS accounts you deploy to do not need to be unique and can be used by more than one space.
- AWS accounts added to a CodeCatalyst space can be used in any project in that space.
- While each environment can support multiple AWS accounts, you can only use one account per environment in an action.
- Billing is configured at the space level. Multiple accounts can be configured for billing, but only one can be active in a CodeCatalyst space. Only one AWS account can be used as a billing account for a space in CodeCatalyst. If an account is already used for a space, you must use a different billing account for the additional space.
- After you create a connection, you must add AWS IAM roles to your connection if your workflow must access those IAM roles with your CodeCatalyst environment. For more information about how environments are used, see Working with environments (p. 307).

The CodeCatalyst workflow actions that require an environment are build or deploy actions that run AWS resources or generate AWS resource stacks.

The Environment, Connection, and Role properties are required to run CodeCatalyst workflow build and deploy actions with AWS resources. The IAM roles and account information are specified in the YAML that you have configured for your workflow. For an example, see the CodeCatalyst build action reference YAML parameters for Environment (p. 367), Connections (p. 368), and Role (p. 368).
Viewing projects in your space

To see projects in your space, you can use the CodeCatalyst console to view the projects in your CodeCatalyst space where you have permissions.

To view the projects in your space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.

   Tip
   If you belong to more than one space, choose a space in the top navigation bar.
3. The page defaults to Projects.
4. To view all projects in your space, choose All projects or choose Your projects to see the projects of which you are a member.

   Note
   If you expect to see a project and don’t see it, ask the Space administrator or Project administrator to resend the project invite.
5. To view a specific project, choose the selector on the card, and then choose to view details.

Viewing activity for space resources

To see recently created projects and status updates, you can use the CodeCatalyst console to view an activity feed that shows updates for space resources.

In the activity feed, you can view metrics such as failed workflow runs and created projects.

To view activity in your space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
Tip
If you belong to more than one space, choose a space in the top navigation bar.

3. Choose Activity.
4. View the information in Activity.
5. To filter by activity, choose the selector on the upper right.
6. To view all activity in your space, choose Any activity type.

Managing AWS accounts

You can use resources from your AWS accounts in Amazon CodeCatalyst spaces. To do so, you must set up a connection between the AWS accounts and your space in CodeCatalyst. Creating a connection like this means that projects and workflows within your CodeCatalyst space can interact with resources in your AWS accounts. You must create one connection for each AWS account you want to use with your CodeCatalyst space.

After you create a connection, you can choose to associate AWS IAM roles with it.

Topics
- Adding an AWS account to a space (p. 37)
- Adding IAM roles to account connections (p. 40)
- Adding the account connection and IAM roles to your deploy environment (p. 41)
- Viewing account connections (p. 41)
- Removing an account from a space (in CodeCatalyst) (p. 42)
- Removing an account from a space (in AWS) (p. 42)
- Tagging account connections (p. 43)

You can set up CodeCatalyst to use authorized AWS accounts by adding the accounts to your space. By adding AWS accounts to your CodeCatalyst space, you can give your project workflows access to AWS account resources and your billing configuration.

Adding an AWS account creates a connection that authorizes CodeCatalyst to use this account. You can use added AWS accounts to do the following:

- Set up billing for a CodeCatalyst space. See Billing (p. 608).
- Allow CodeCatalyst to assume IAM roles to access AWS resources and deploy to AWS services in the account. See Managing AWS accounts (p. 36).

The following diagram shows how account connections are created by completing authorization with the AWS account. After the connection is created, you further configure the connection for workflows and projects to use by adding IAM roles.
If this is your first time using CodeCatalyst, we suggest starting with Tutorial: Creating a blueprint-based web application project. You can also follow CodeCatalyst tutorials for deploying with AWS resources and the requisite IAM roles for those resources, such as using AWS CloudFormation or Amazon ECS to deploy an application.

Here is one possible flow for adding an AWS account in CodeCatalyst:

Li Juan has the Project administrator role in a CodeCatalyst project with a workflow that builds and deploys the application to AWS infrastructure in the cloud. To deploy to the AWS infrastructure, CodeCatalyst must use an authorized AWS account to access the AWS resources for the build action in the workflow. Li Juan works with Mary Major, who has the Space administrator role, and Mateo Jackson, who has AWS administrator permissions in the AWS account to create a connection between the space and the AWS account. Before creating the connection, Mateo Jackson creates an IAM role in that account called codecatalyst-build-role with the IAM permissions policy for the AWS Cloud Development Kit (AWS CDK) stack he wants to use to build the application in the AWS account.

As the next step, Mary Major edits the CodeCatalyst space settings, completes an authorization flow with Mateo Jackson, and adds the AWS account and role to the list of AWS accounts and roles available for the CodeCatalyst space. Li Juan uses the CodeCatalyst environments page to add the account and role to the environment for his CodeCatalyst project. Li Juan also adds the role Amazon Resource Name (ARN) to the Role field for the CodeCatalyst workflow YAML.

**Adding an AWS account to a space**

You use the CodeCatalyst console and the AWS Management Console to connect your space to an AWS account.

Before adding an AWS account to a space in CodeCatalyst, complete the following prerequisites:
• Create an AWS account and acquire permissions to create AWS IAM roles in the account you want to connect.
• Create the IAM role or roles you want to associate with your account connection, including the IAM policies with permissions for the roles.
• Acquire the Space administrator role in the CodeCatalyst space where you want to create the connection.

Topics
• Step 1: Creating a connection request (p. 38)
• Step 2: Accepting an account connection request (p. 38)
• Step 3: Review an approved connection (p. 39)
• Step 4: Add IAM roles to your connection (p. 39)
• Next steps: Create additional IAM roles for your account connection (p. 39)

Step 1: Creating a connection request

Creating a connection request in the CodeCatalyst console generates a connection token that you can use to complete authorization.

You must have the Space administrator or Power user role in the CodeCatalyst space where you want to create the connection. You must also have administrative permissions for the AWS account you want to add.

To create a connection

1. In the AWS Management Console, make sure you are logged in with the same account that you want to create a connection with.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.
4. Choose Add an AWS account.
5. On the Associate AWS account with Amazon CodeCatalyst page, in AWS account ID, enter the twelve-digit ID for the account you want to connect to your space. For information about finding your AWS account ID, see Your AWS account ID and its alias.
6. (Optional) In Amazon CodeCatalyst display name, enter a reference name for the account.
7. (Optional) In Connection description, enter a description for the account that will help you choose the projects where the account and role or roles will apply.
8. Choose Associate AWS account.
9. The page returns to the AWS account details page where a success banner displays.

Step 2: Accepting an account connection request

After you submit a request in the CodeCatalyst console to connect to your AWS account, you work with your AWS administrator to accept the connection request by submitting it with the provided connection token.

Make sure you have administrator permissions for your account, and you're signed in to the AWS Management Console with the same AWS account for which you're creating the connection.

To approve a connection request (console)

1. In the AWS Management Console, make sure you are logged in with the same account that you want to create a connection with.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.
4. On the AWS account details page, choose Complete setup in the AWS Management Console.
5. The Verify Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to log in to access the page.

   To directly access the page, sign in to the Amazon CodeCatalyst Spaces in the AWS Management Console at https://console.aws.amazon.com/codecatalyst/home/.

   The verification token is automatically entered in Verification token. A success message shows a message that the token is a valid token.

6. Choose Verify space.

   An Account verified success message displays to show that the account has been added to the space.

**Step 3: Review an approved connection**

After getting a connection approved, you can view the connection in the console, along with the IAM roles you added to it.

**To review an approved connection**

1. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.
2. The account connection is listed with the date it was created.
3. Choose the account display name. The AWS account details page displays.

**Step 4: Add IAM roles to your connection**

If you're using an IAM role configured for a CodeCatalyst deploy action, add the role to your deployment environment. For more information, see Adding IAM roles to account connections (p. 40).

**Next steps: Create additional IAM roles for your account connection**

After you create a connection, you can create additional IAM roles to add to it. The IAM roles that you add are dependent on your workflows. For example, a CodeCatalyst build action requires the CodeCatalyst build role.

To connect your account, you will need the Amazon Resource Name (ARN) for the roles you created. Copy the ARN for your role or roles as detailed here. For more information about working with ARNs for IAM roles, see Amazon Resource Name (ARN).

To access your IAM role ARN

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles.
3. In the search box, enter the name of the role you want to add.
4. Choose the role from the list.

   The role's Summary page appears.

5. At the top, copy the Role ARN value.
Adding IAM roles to account connections

Part of creating your account connection includes adding the IAM role or roles you want to use with projects in your CodeCatalyst space.

**Note**
To use IAM roles with an account connection, make sure that the trust policy is updated to use the CodeCatalyst service principal.

### Add IAM roles to an account connection (console)

1. In the AWS Management Console, make sure you are logged in with the same account that you want to manage.
2. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
3. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **AWS accounts**.
4. Choose your account connection, and then choose **Manage roles from AWS Management Console**.

   The **Add IAM role to Amazon CodeCatalyst space** page displays.

5. Do one of the following:
   - To create a service role that contains the permissions policy and trust policy for the developer role, choose **Create CodeCatalyst development administrator role in IAM**. The role will have a name **CodeCatalystWorkflowDevelopmentRole-spaceName** with a unique identifier appended. For more information about the role and role policy, see [Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role](p. 516).
   - To add a role that you have already created in IAM, choose **Add an existing IAM role**. In **Select existing IAM role**, choose the role from the drop-down list.

   Choose **Add role**.

   The page opens in the AWS Management Console. You might need to log in to access the page.

6. In the **Amazon CodeCatalyst Spaces** page navigation pane, choose **Spaces**.

   To directly access the page, sign in to the Amazon CodeCatalyst Spaces in the AWS Management Console at [https://console.aws.amazon.com/codecatalyst/home/](https://console.aws.amazon.com/codecatalyst/home/).

7. Choose the account added for your CodeCatalyst space. The connection page is shown.

8. On the connection page, under **IAM roles available to CodeCatalyst**, view the list of IAM roles added to your account. Choose **Associate IAM role to CodeCatalyst**.

9. On the **Associate an IAM role** pop-up, in **Role ARN**, enter the Amazon Resource Name (ARN) of the IAM role you want to associate with your CodeCatalyst space.

   Under **Purpose**, choose a role purpose that describes how you want to use the role in your account connection. Specify RUNNER for roles that you use to run actions in workflows. Specify SERVICE for roles that you use to access another service.

   You can specify more than one purpose.

   **Note**
   Choosing a purpose for the role ARN is required.

10. Choose **Associate an IAM role**. Repeat these steps for additional IAM roles.
Adding the account connection and IAM roles to your deploy environment

To access AWS resources, such as Amazon ECS or AWS Lambda resources for deployments, CodeCatalyst build and deploy actions require IAM roles with permissions to access those resources. With the **Space administrator** or **Power user** role, you can connect your CodeCatalyst account to the AWS account where your resources are created. You then add the IAM role to your account connection. For deploy actions, you must then add the IAM role to a CodeCatalyst environment.

You must add the IAM roles that you want to use with deployment environments in your projects. Adding the roles to the account connection does not add the roles and the connection to the project deploy environments. To add your account connection and IAM roles to your deploy environment, make sure that the account connection and roles are created as detailed in [Step 4: Add IAM roles to your connection](p. 39).

Then, use the **Environments** page in the CodeCatalyst console to add your account connection and IAM role to a deploy environment in a project.

**Note**
You only add an IAM role to an environment if the IAM role is used for a CodeCatalyst action that requires an IAM role. All workflow actions that require IAM roles, including build actions, must use a CodeCatalyst environment.

To add your account connection and IAM roles to your deploy environment
1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the project with the deployment environment where you want to add the account connection and IAM roles.
3. Expand **CI/CD**, and then choose **Environments**.
4. Choose your environment, and then the additional tabs display.
5. Choose the **AWS account connections** tab. Under **Connection name**, the accounts that have been added to the environment, if any, are listed.
6. Choose **Associate AWS account**. The **Associate AWS account with <environment_name>** page displays.
7. Under **Connection**, choose the name of the account connection with the IAM roles that you want to add. Choose **Associate**.

Viewing account connections

You can view a list of your connections and view details about each connection.

You must have the **Space administrator** or **Power user** role to manage connections for your space.

**To view all connections for a CodeCatalyst space**
1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the space with the account connection that you want to view.
3. Choose the **AWS accounts** tab.
4. Under **AWS accounts**, view the list of account connections for the space, including the account ID and status for each connection.

**To view account connection details**
1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
Removing an account from a space (in CodeCatalyst)

You can delete an account connection that you no longer need. For this procedure, you will use CodeCatalyst to delete an account connection that you have previously added to your space. This deletes the account connection from your space, provided that the account is not the billing account for the space.

**Important**
After an account connection is deleted, you cannot reconnect it. You must create a new account connection and then associate IAM roles and environments, or set up billing, as needed.

A billing account must be designated for your CodeCatalyst space, even if usage for the space will not exceed the Free tier. Before you can remove a space for an account that is a designated billing account, you will need to add another account for your space. See the steps in [Changing a billing account](p. 612).

**Important**
While you can use these steps to remove an account, this is not recommended. The account might also be set up to support workflows in CodeCatalyst.

To manage account connections for your space, you must have the **Space administrator** or **Power user** role.

**To delete an account connection**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **AWS accounts**.
3. In **Connection name**, choose the selector next to the account connection that you want to remove.
4. Choose **Remove AWS account**. Confirm the deletion by entering the name in the field, and then choose **Disconnect**.

A success banner displays, and the account connection is removed from the list of connections.

Removing an account from a space (in AWS)

You can use the page for CodeCatalyst in AWS to remove an account that has been added to a space. For this procedure, using administrative permissions for the specific account you are managing, you sign in the Amazon CodeCatalyst Spaces page in the AWS Management Console to remove an AWS account from your space. To remove an account that is a designated billing account for your CodeCatalyst space, make sure to first specify a new billing account.

An account that has been removed can be added again later, but you must create a new connection between the account and the space. You will need to re-associate any IAM roles to the added account.

A billing account must be designated for your CodeCatalyst space, even if usage for the space will not exceed the Free tier. Before you can remove a space for an account that is a designated billing account, you will need to add another account for your space. See the steps in [Changing a billing account](p. 612).

**Important**
While you can use these steps to remove an account, this is not recommended. The account might also be set up to support workflows in CodeCatalyst.
Tagging account connections

Connections are represented by a connection resource Amazon Resource Name (ARN) that is unique to the connection between a specific AWS account and a specific space in CodeCatalyst.

Using tags, you can organize and control access to your resources with tag-based IAM policies. For an example IAM policy, see Identity-based policy examples for CodeCatalyst connections (p. 507).

Adding tags for account connections

Use these steps to create and apply tags to an account connection.

To add a tag

1. Make sure you're signed in to the AWS Management Console with the AWS account for the account connection you want to manage.
2. If the Details page doesn't display, choose Spaces. From the list under Amazon CodeCatalyst space, choose the space that corresponds with the account connection you want to tag.

   The Amazon CodeCatalyst space details page displays.
3. Choose Manage tags. The Manage tags page displays.
4. Choose Add tag. Add a tag by entering a key and a value for the key-value pair.
5. Choose Save changes.

Editing tags for account connections

Use these steps to create and apply tags for your account connection resources.

You must have the **Space administrator** or **Power user** role to manage account connections for your space.

Only one AWS account can be used as a billing account for a space in CodeCatalyst. If an AWS account is already used for a space, you must use a different billing account for the additional space.

**To remove an added account**

1. In the AWS Management Console, make sure you are logged in with the same account that you want to manage.
2. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
3. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **Billing**.
4. View the billing account information on the page to make sure the account you want to remove is not the designated billing account for the space.
5. Choose **Manage billing in AWS**. This opens the Amazon CodeCatalyst Spaces in the AWS Management Console. If you're prompted to log in, log in to AWS, and then choose the button again to load the page.
6. On the **Amazon CodeCatalyst Spaces** page, choose the space with the account that you want to remove. The details page for the space displays.
7. Choose **Remove space**.
8. In **Remove CodeCatalyst space from this account**, enter the space name to confirm. Choose **Remove**.
Managing IAM roles for connected accounts

You create roles in AWS Identity and Access Management (IAM) for the account that you want to add to CodeCatalyst. If you are adding a billing account, you do not need to create roles.

In your AWS account, you must have permissions to create roles for the AWS account you want to add to your space. For more information about IAM roles and policies, including IAM references and example policies, see Identity and Access Management and Amazon CodeCatalyst (p. 505). For more information about the trust policy and service principals used in CodeCatalyst, see CodeCatalyst trust model (p. 543).

In CodeCatalyst, you must be signed in with the Space administrator role to complete the steps to add accounts (and the roles, if applicable) to your space.

You can add roles to your account connections by using one of the following methods.

To edit a tag

1. Make sure you’re signed in to the AWS Management Console with the AWS account for the account connection you want to manage.
3. If the Details page doesn't display, from the side navigation, choose Spaces. From the list under Amazon CodeCatalyst space, choose the space that corresponds with the account connection you want to tag.
   
   The Amazon CodeCatalyst space details page displays.
5. In the row for the tag you want to edit, change the key pair value by editing the key, the value, or both.
6. Choose Save changes.

Removing tags for account connections

Use these steps to remove tags for an account connection.

To remove a tag

1. Make sure you’re signed in to the AWS Management Console with the AWS account for the account connection you want to manage.
3. If the Details page doesn't display, from the side navigation, choose Spaces. From the list under Amazon CodeCatalyst space, choose the space that corresponds with the account connection you want to tag.
   
   The Amazon CodeCatalyst space details page displays.
5. For each tag you want to remove, choose Remove on the row for that tag. You can remove multiple tags.
6. Choose Save changes.
To create a service role that contains the permissions policy and trust policy for the CodeCatalystWorkflowDevelopmentRole-spaceName role, see CodeCatalystWorkflowDevelopmentRole-spaceName role (p. 45).

For an example of creating a role and adding a policy to create a project from a blueprint, see Creating an IAM role and using the CodeCatalyst trust policy (p. 46).

For a list of sample role policies to use when creating your IAM roles, see IAM roles for Amazon CodeCatalyst access to AWS resources (p. 515).

For detailed steps to create roles for workflow actions, see the workflow tutorial for that action as follows:

- Tutorial: Upload artifacts to Amazon S3 (p. 140)
- Tutorial: Deploy a serverless application using AWS CloudFormation (p. 165)
- Tutorial: Deploy an application to Amazon ECS (p. 183)
- Tutorial: Lint code using a GitHub Action (p. 268)

Topics

- CodeCatalystWorkflowDevelopmentRole-spaceName role (p. 45)
- AWSRoleForCodeCatalystSupport role (p. 46)
- Creating an IAM role and using the CodeCatalyst trust policy (p. 46)

**CodeCatalystWorkflowDevelopmentRole-spaceName role**

You create the developer role as a 1-click role in IAM. You must have the Space administrator or Power user role in the space where you want to add the account. You must also have administrative permissions for the AWS account you want to add.

Before you start the procedure below, you must log in to the AWS Management Console with the same account that you want to add to your CodeCatalyst space. Otherwise, the console will return an unknown account error.

**To create and add the CodeCatalyst CodeCatalystWorkflowDevelopmentRole-spaceName**

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.

2. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).

3. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.

4. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.

5. Choose Manage roles from AWS Management Console.

   The Add IAM role to Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to log in to access the page.

6. Choose Create CodeCatalyst development administrator role in IAM. This option creates a service role that contains the permissions policy and trust policy for the development role. The role will have a name CodeCatalystWorkflowDevelopmentRole-spaceName. For more information about the role and role policy, see Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role (p. 516).
Note
This role is only recommended for use with developer accounts and uses the AdministratorAccess AWS managed policy, giving it full access to create new policies and resources in this AWS account.

7. Choose Create development role.
8. On the connection page, under IAM roles available to CodeCatalyst, view the CodeCatalystWorkflowDevelopmentRole-spaceName role in the list of IAM roles added to your account.
9. To return to your space, choose Go to Amazon CodeCatalyst.

AWSRoleForCodeCatalystSupport role

You create the support role as a 1-click role in IAM. You must have the Space administrator or Power user role in the space where you want to add the account. You must also have administrative permissions for the AWS account you want to add.

Before you start the procedure below, you must log in to the AWS Management Console with the same account that you want to add to your CodeCatalyst space. Otherwise, the console will return an unknown account error.

To create and add the CodeCatalyst AWSRoleForCodeCatalystSupport

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.
2. Navigate to your CodeCatalyst space. Choose Settings, and then choose AWS accounts.
3. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.
4. Choose Manage roles from AWS Management Console.

The Add IAM role to Amazon CodeCatalyst space page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to sign in to access the page.

5. Under CodeCatalyst space details, choose Add CodeCatalyst Support role. This option creates a service role that contains the permissions policy and trust policy for the preview development role. The role will have a name AWSRoleForCodeCatalystSupport with a unique identifier appended. For more information about the role and role policy, see Understanding the AWSRoleForCodeCatalystSupport service role (p. 517).
6. On the Add role for CodeCatalyst Support page, leave the default selected, and then choose Create role.
7. Under IAM roles available to CodeCatalyst, view the CodeCatalystWorkflowDevelopmentRole-spaceName role in the list of IAM roles added to your account.
8. To return to your space, choose Go to Amazon CodeCatalyst.

Creating an IAM role and using the CodeCatalyst trust policy

IAM roles to be used in CodeCatalyst with AWS account connections must be configured to use the trust policy provided here. Use these steps to create an IAM role and attach a policy that allows you to create projects from blueprints in CodeCatalyst.
As an alternative, you can create a service role that contains the permissions policy and trust policy for the CodeCatalystWorkflowDevelopmentRole-<spaceName> role. For more information, see Adding IAM roles to account connections (p. 40).

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Roles, and then choose Create role.
3. Choose Custom trust policy.
4. Under the Custom trust policy form, paste the following trust policy.

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "codecatalyst-runner.amazonaws.com",
        "codecatalyst.amazonaws.com"
      ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/*"
      }
    }
  }
]
```

5. Choose Next.
6. Under Add permissions, search for and select a custom policy that you have already created in IAM.
7. Choose Next.
8. For Role name, enter a name for the role, for example: codecatalyst-project-role
9. Choose Create role.
10. Copy the role Amazon Resource Name (ARN). You'll need to provide this information when adding the role to your account connection or environment.

Viewing members in a space

You can view the users in your space, including information about their display names, aliases, and the role they have for the space. There are three roles for members in a space:

- **Space administrator** – This role has all permissions in CodeCatalyst, including creating projects. Only assign this role to users who need to administer every aspect of a space, such as accessing all projects in the space.

  You cannot change this role later without removing the user first. For more information, see Space administrator role (p. 477).

- **Power user** – This role is the second-most powerful role in Amazon CodeCatalyst spaces, but it has no access to projects in a space. It is designed for users who need to be able to create projects in a space and help manage the users and resources for the space. For more information, see Power user role (p. 483).
• **Limited access** – This role is assigned by default for users who join the space by accepting invitations to projects in the space. Project members are assigned a role in a project. For information about managing project members, see Managing project members (p. 62).

The **Space administrators** table shows users with the **Space administrator** role. These users are not shown in the **Space members** because they are automatically (implicitly) assigned to all projects in the space and do not have a role in a project.

The **Space members** table shows all members in the space that have a role in a project while not having the **Space administrator** role.

Users are shown based on whether the user has the **Space administrator** role in CodeCatalyst as follows:

- A user with the **Space administrator** role who later accepts a project invitation and role will not show in the **Space members** table under spaces or on the **Project members** table under projects. They will continue to be shown in the **Space administrators** table in both places. In each project, all users with the **Space administrator** role are shown in the project **Space administrators** table for that project.
- A user who accepts a project invitation to join with a project role is added to the space with the **Limited access** role. If the user's role later changes to the **Space administrator** role, but will also move from the **Space members** table to the **Space administrators** table. Under the project, the user will move from the **Project members** table to the **Space administrators** table.

**To view users and roles in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your space.
   
   **Tip**
   
   If you belong to more than one space, choose a space in the top navigation bar.
3. Choose **Settings**, and then choose **Members**.

   Users who are members of the space are shown in the **Space members** table.

   **Tip**
   
   If you have the **Space administrator** role, you can view which projects you have been directly invited to. Navigate to **Project settings** for the project, and then choose **My projects**.

   In the **Status** column, the following are valid values:

   - **Invited** – CodeCatalyst sent the invitation but the user has not yet accepted or declined.
   - **Member** – The user accepted the invitation.

---

**Inviting a user directly to a space**

You can invite users directly to your CodeCatalyst space. This is useful when you want to invite that user to help you manage the space by assigning them the **Space administrator** or **Power user** role. Assigning one of those roles to other users can help you distribute the responsibilities of managing the space across more people without having to invite these users to any projects.

**Note**

You must have the **Space administrator** or **Power user** role to invite members.

The **Space administrators** table shows users with the **Space administrator** role. These users are not shown in the **Space members** table because they are automatically (implicitly) assigned to all projects in the space and do not have a role in a project.
Members who accept a project invitation are added to the space by default. The Project members table shows all members in the space that have a role in a project.

For more information about how to accept an invitation and sign in for the first time, see Setting up CodeCatalyst (p. 7).

To invite a user to your space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.
3. Choose Settings, and then choose Members.
5. Enter the email of the person you would like to invite to join your space. In Role, choose the role you want to assign that user in the space.
6. Choose Invite

Canceling an invitation for a space

If you want to cancel an invitation to join a space that you sent recently, and it has not yet been accepted, you can cancel it.

To manage space invitations, you must have the Space administrator or Power user role.

To cancel a space member invitation

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.
   Tip
   If you belong to more than one space, choose a space in the top navigation bar.
3. Choose Settings, and then choose Members.
4. Verify that the member has a status of Invited.
   Note
   You can only cancel an invitation that has not yet been accepted.
5. Choose the option next to the row with the invited member, and then choose Cancel invitation.
6. A confirmation window displays. Choose Cancel invitation to confirm.

Changing the role for a space member

You can change the assigned role for a member of your space. You must have the Space administrator role to change the role of a user in the space.

The Space administrators table shows users with the Space administrator role. These users are not shown in the Space members table because they are automatically (implicitly) assigned to all projects in the space.

To change the role for a user in your space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.
   Tip
   If you belong to more than one space, choose a space in the top navigation bar.
3. Choose Settings, and then choose Members.
4. In the Space members table, choose the user whose role you want to change. Choose Change role.

Removing a space member

You can remove a member of your space when they do not need to access any of the space resources. You must have the Space administrator role to remove a member from a space.

The Space administrators table shows users with the Space administrator role. These users are not shown in the Space members table because they are automatically (implicitly) assigned to all projects in the space and do not have a role in a project. You can only directly remove a member of your space in this table.

To remove a user from the Project members table
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.
   Tip
   If you belong to more than one space, choose a space in the top navigation bar.
3. Choose Settings, and then choose Members.
4. Choose the user in the Project members table. Choose Remove.
   Note
   Removing a member from the space will remove the user from all projects in the space, along with permissions associated with the resources in those projects.

Removing or changing the role for a user with the Space administrator role

You can remove or change the role for a user with the Space administrator role for your space.

You must have the Space administrator role to remove a user with the Space administrator role from a space. Changing the role for a user with the Space administrator role essentially removes the user from the Space administrators table. If that user does not have a project role in any projects in the space, removing the Space administrator role from the user will remove the user from the space.

   Note
   As a user with the Space administrator role, you cannot remove yourself. Contact another user with the Space administrator role.

To remove a user with the Space administrator role from the Space members table

   Note
   For a user who has not been added explicitly to a project, they do not have any project roles (Project administrator or Contributor). If the Space administrator role is the user's only role, then the user is removed from the space entirely.
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space where you want to remove or change the role for a user with the Space administrator role.
3. Choose Settings, and then choose Members.
4. View the invitation status for the list of members, and make sure that the list contains no unauthorized pending invites to the space (a status of Invited).

   **Important**
   Before removing a user with the Space administrator role, you must verify that no pending invites have been initiated.

5. Choose the Members tab. In the Space administrators table, choose the user, and then choose Remove.

   On the Remove member dialog box, do one of the following.

   • Choose the option to remove only the user’s Space administrator role. Choose Remove.

     **Important**
     If the user does not have any other role assigned, then changing the role from Space administrator removes the user from the space.

   • Choose the option to remove a user with the Space administrator role from the space and all its projects. Choose Remove.

6. Refresh the Members tab. The user is automatically added to the list of project members in any project where the user had membership through project roles. If the Space administrator role was the user’s only role, then the user is removed from the space entirely.

---

**Administering Dev Environments for a space**

All Dev Environments are created as part of a project within a space. Space members can create their own Dev Environments within a project at the source repository level. Space administrators can then use the Amazon CodeCatalyst console to view, edit, delete, and stop Dev Environments on behalf of space members. In short, space administrators maintain Dev Environments at the space level.

**Considerations for administering Dev Environments**

- You must have the Space administrator role to view the Dev Environments page under Settings and to manage Dev Environments at the space level.
- Space members manage the Dev Environments that they create in projects through their CodeCatalyst accounts. When administering Dev Environments as a space administrator, you are maintaining these resources on behalf of space members.
- Dev Environments default to a specific compute and storage configuration. For information about billing and rates for upgrading your configuration, see the Amazon CodeCatalyst pricing page.

For other considerations about Dev Environments, including stopping running instances, default compute configuration, upgrading your compute, incurring costs, and configuring timeouts, see Dev Environments in CodeCatalyst (p. 113).

**Topics**

- Viewing Dev Environments for your space (p. 51)
- Editing a Dev Environment for your space (p. 52)
- Stopping a Dev Environment for your space (p. 52)
- Deleting a Dev Environment for your space (p. 53)

---

**Viewing Dev Environments for your space**

You can view the type, status, and details for all Dev Environments in your space. For more information about creating and running Dev Environments, see Creating a Dev Environment (p. 113).
You must have the **Space administrator** role to view this page and to manage Dev Environments at the space level.

**To view Dev Environments in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   - **Tip**
     If you belong to more than one space, choose a space in the top navigation bar.
3. Choose **Settings**, and then choose **Dev Environments**.

   The page lists all Dev Environments in your space. You can view the **Resource** name, the resource **alias** if applicable, the type of **IDE**, the default or configured **Compute** and **Storage**, and the configured **Timeout** for each Dev Environment.

**Editing a Dev Environment for your space**

You can edit the configuration for a Dev Environment, such as the configured length of timeout, if any, for an idle Dev Environment to stop running. For more information about editing a Dev Environment, see **Editing a Dev Environment** (p. 117).

You must have the **Space administrator** role to view this page and to manage Dev Environments at the space level.

**To edit Dev Environments in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   - **Tip**
     If you belong to more than one space, choose a space in the top navigation bar.
3. Choose **Settings**, and then choose **Dev Environments**.
4. Choose the selector next to the Dev Environment you want to manage. Choose **Edit**.
5. Make the changes you want to the compute or inactivity timeout for the Dev Environment.
6. Choose **Save**.

**Stopping a Dev Environment for your space**

You can stop a running Dev Environment before it becomes idle if the Dev Environment is configured to have a timeout. Otherwise, a Dev Environment with an elapsed timeout will already be stopped. For more information about stopping a Dev Environment, see **Stopping a Dev Environment** (p. 115).

You must have the **Space administrator** role to view this page and to manage Dev Environments at the space level.

**To stop a Dev Environments in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   - **Tip**
     If you belong to more than one space, choose a space in the top navigation bar.
3. Choose **Settings**, and then choose **Dev Environments**.
4. Choose the selector next to the Dev Environment you want to manage. Choose **Stop**.

### Deleting a Dev Environment for your space

You can delete a Dev Environment that is no longer needed or that no longer has an owner. For more information about considerations for deleting a Dev Environment, see *Deleting a Dev Environment* (p. 118).

You must have the **Space administrator** role to view this page and to manage Dev Environments at the space level.

**To delete Dev Environments in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   
   **Tip**
   If you belong to more than one space, choose a space in the top navigation bar.
3. Choose **Settings**, and then choose **Dev Environments**.
4. Choose the selector next to the Dev Environment you want to manage. Choose **Delete**. To confirm, type `delete`, and then choose **Delete**.

### Creating additional spaces in CodeCatalyst

When you first sign up in Amazon CodeCatalyst, you are required to create a space. For more information, see *Setting up CodeCatalyst* (p. 7). You can choose to create additional spaces to meet your business needs.

**Note**
Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces.

To create additional spaces, you must be assigned the Space administrator role.

**Note**
When you create an additional space, you are not prompted to create a project. To learn how to create projects in a space, see *Creating a project in Amazon CodeCatalyst* (p. 58).

**To create another space**

1. In the AWS Management Console, make sure you are signed in with the same AWS account that you want to associate with your CodeCatalyst space.
2. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
3. Navigate to your space.
   
   **Tip**
   If you belong to more than one space, choose a space in the top navigation bar.
4. Choose **Create space**.
5. On the **Create a space** page, in **Space name**, enter a name for the space. You cannot change this later.
   
   **Note**
   Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces.
6. In **AWS Region**, choose the Region where you want to store your space and project data. You cannot change this later.

7. In **AWS account ID**, enter the twelve-digit ID for the account you want to connect to your space.

   In **AWS account verification token**, copy the generated token ID. The token is automatically copied for you, but you might want to store it while you approve the AWS connection request.

8. Choose **Verify in AWS**.

9. The **Verify Amazon CodeCatalyst space** page opens in the AWS Management Console. This is the **Amazon CodeCatalyst Spaces** page. You might need to sign in to access the page.

   In the AWS Management Console, make sure to choose the same AWS Region where you want to create your space.

   To directly access the page, sign in to the Amazon CodeCatalyst Spaces in the AWS Management Console at https://console.aws.amazon.com/codecatalyst/home/.

   The verification token is automatically entered in **Verification token**. A success banner shows a message that the token is a valid token.

10. Choose **Verify space**.

    An **Account verified** success message displays to show that the account has been added to the space.

11. Remain on the **Verify Amazon CodeCatalyst space** page. Choose the following link: **To enable the Standard tier or add IAM roles this space, view space details**.

    The **CodeCatalyst space details** page opens in the AWS Management Console. This is the **Amazon CodeCatalyst Spaces** page. You might need to log in to access the page.

12. Under **IAM roles available to CodeCatalyst**, choose **Add IAM role**.

    The **Add IAM roles available to CodeCatalyst** page displays.

13. Choose **Create CodeCatalyst development administrator role in IAM**. This option creates a service role that contains the permissions policy and trust policy for the development role.

    The developer role is an AWS IAM role that enables your CodeCatalyst workflows to access AWS resources such as Amazon S3, Lambda, and AWS CloudFormation. The role will have a name `CodeCatalystWorkflowDevelopmentRole-spaceName` with a unique identifier appended. For more information about the role and role policy, see **Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role** (p. 516).

14. Choose **Create development role**.

15. On the connection page, under **IAM roles available to CodeCatalyst**, view the developer role in the list of IAM roles added to your account.

16. Choose **Go to Amazon CodeCatalyst**.

17. On the creation page in CodeCatalyst, choose **Create space**.

---

**Editing the space description in CodeCatalyst**

You can change the description of a space to help users better understand what it’s for.

You must have the **Space administrator** role to edit space details.

**To edit a space description**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your space.
Deleting a space

Tip
If you belong to more than one space, choose a space in the top navigation bar.

3. On the Space settings tab, choose Edit. Make the changes you want to the space description, and then choose Save.

Deleting a space in CodeCatalyst

You can delete a space to remove access to all of the space's resources. You must have the Space administrator role to delete a space.

Note
You cannot undo a space deletion.

After you have deleted a space, all space members will be unable to access space resources. Billing for space resources will also stop, and any workflows that are prompted by third-party source repositories will be stopped.

Note
Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces.

To delete a space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your space.

   Tip
   If you belong to more than one space, choose a space in the top navigation bar.

3. Choose Settings, and then choose Delete.
4. Type delete to confirm the deletion.
5. Choose Delete.

   Note
   If you belong to more than one space, you're redirected to the space overview page. If you belong to one space, you're redirected to the space creation page.

Quotas for spaces in CodeCatalyst

The following table describes quotas and limits for spaces in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see Quotas for CodeCatalyst (p. 620).

<table>
<thead>
<tr>
<th>Quota</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of active spaces per user per AWS Region</td>
<td>Five</td>
</tr>
<tr>
<td>Number of space creations per Region per month per user</td>
<td>Five</td>
</tr>
<tr>
<td>Space descriptions</td>
<td>Space descriptions are optional. If specified, they must be between 0 and 200 characters in length. They can contain any combination of letters, numbers, spaces, periods, underscores, commas, dashes, and the following special characters: ? &amp; $ % + = / \ ; : \n \t \r</td>
</tr>
</tbody>
</table>

55
### Space names

Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces. Space names must be between 3 and 63 characters in length. They must also begin with an alphanumeric character. Space names can contain any combination of letters, numbers, periods, underscores, and dashes. They cannot contain any of the following characters:

! ? @ # $ % ^ & * ( ) + = { } [ ] | \ > < ~ ` ' " ; :
Projects in CodeCatalyst

You use projects in Amazon CodeCatalyst to establish a collaboration space where development teams can conduct development tasks with shared continuous integration/continuous delivery (CI/CD) workflows and repositories. When you create a project, you can add, update, or remove resources. You can also monitor the progress of your team's work. You can have multiple projects within a space.

Spaces in CodeCatalyst are made up of projects. You can see every project within your space, but you can only use the projects of which you are a member. When you create a project, default roles for your project are generated, which you assign to users that you invite to your project.

- Anyone assigned to the project with a project role, such as the Contributor role, can access project resources, such as a source repository.
- Anyone with the Space administrator Project administrator role can send invitations to join a project.
- Users with the Project administrator role can track activity, status, and other settings across shared resources.
- Users with the Limited access role can manage project assignments for features, code fixes, and tests as part of CI/CD workflows.

Workflows are used to build, test, and release or update applications as a CI/CD pipeline. You can assemble workflows by adding actions that transfer and work on your source artifacts. When you run actions, your project cloud resources are used to provide on-demand compute ability for your workflow actions. You might configure more CI/CD workflows based on the activity and output you want to set up. For example, you might create a workflow for build and test actions only, where you can view test results and complete the workflow without a deployment while you fix bugs. Then, you might create another workflow to build and deploy your application to a staging environment.

When you create a project, you can use a blueprint to create a project that contains sample code and creates resources, or you can start with an empty project. If you create a project using a blueprint, the blueprint you choose determines which resources are added to your project and the tools that CodeCatalyst creates or configures so you can track and use your project resources. You can manually add or remove resources after you have created a project. The following resources may be created or configured by CodeCatalyst:

- **Issues** – You track work related to your project in distinct record called issues. You can make issues for features, tasks, bugs, and any other work for your projects.
- **Notifications** – You configure notifications by choosing the resource you want to watch, the event that you want to watch for, and the destination client or email where you want to receive notifications.
- **Search** – You can search your project for code, issues, users, pull requests, and packages. You can search in a single project or you can search in all projects.
- **Source repository** – You work with your source code in a repository for your project. When you commit a source code change or merge a pull request in a specified branch, CodeCatalyst updates your source.

Each project tracks project activity as a list of events by user, such as when a project is created or a resource is modified. Project activity is monitored and aggregated at the space level. For more information about working with activity data, see [Viewing all projects](#).

If your project uses AWS resources, you can connect your CodeCatalyst account to an AWS account where you have administrative permissions to integrate resources for your project.

You can add source repositories, issues, and other resources to your project after you create it. You must have the Space administrator role to create projects.
Creating a project in Amazon CodeCatalyst

With CodeCatalyst projects, you can conduct development tasks with shared continuous integration/continuous delivery (CI/CD) workflows and repositories, manage resources, track issues, and add users.

Before you create a project, you must have the **Space administrator** or **Power user** role.

**Topics**

- Creating a project with a blueprint (p. 58)
- Creating an empty project in Amazon CodeCatalyst (p. 58)
- Creating a project with a linked GitHub repository (p. 59)

Creating a project with a blueprint

You can provision all of your project resources and sample code with a project blueprint. For information about blueprints, see the blueprint reference at [Project blueprint reference](p. 65).

**To create a project with a blueprint**

1. In the CodeCatalyst console, navigate to the space where you want to create a project.
2. On the space dashboard, choose **Create project**.
3. Choose **Start with a blueprint**.
4. Choose a blueprint, and then choose **Next**.
5. Under **Give a name to your project**, enter the name that you want to assign to your project and its associated resource names. The name must be unique within your space.
6. Under **Project resources**, configure common project parameters.
7. (Optional) To view definition files with updates based on the project parameter selections you made, choose **View code** or **View workflow** from **Generate project preview**.
8. (Optional) Choose **View details** from the blueprint's card to view specific details about the blueprint, such as an overview of the blueprint's architecture, required connections and permissions, and the kind of resources the blueprint creates.
9. Choose **Create project**.

For more information about project blueprints, see [Project blueprint reference](p. 65).

Creating an empty project in Amazon CodeCatalyst

You can create an empty project with no resources and manually add the resources you want at a later time.

Before you create a project, you must have the **Space administrator** or **Power user** role.

**To create an empty project**

1. Navigate to the space where you want to create a project.
2. On the space dashboard, choose **Create project**.
3. Choose **Start from scratch**.
4. Under **Give a name to your project**, enter the name that you want to assign to your project. The name must be unique within your space.
5. Choose **Create project**.
Creating a project with a linked GitHub repository

You can create a new CodeCatalyst project that links to a GitHub source repository. You can then use your linked GitHub source repository in your CodeCatalyst project.

Before you create a CodeCatalyst project, you must have the Space administrator or Power user role. For more information, see Creating additional spaces in CodeCatalyst (p. 53) and Inviting a user directly to a space (p. 48).

You must already have a GitHub account, and you need to have already created the repository you want to link to.

To create a project in CodeCatalyst that links to a source repository in your GitHub account, you'll need to complete the following three tasks:

1. Install the GitHub repositories extension.
2. Connect your GitHub account to CodeCatalyst.
3. Create a CodeCatalyst project linked to your GitHub account.

To install the GitHub repositories extension

1. Navigate to the space where you want to create a project.
2. On the space dashboard, choose Create project.
3. Choose Bring your own code.

   If the GitHub repositories extension is not already installed, an install prompt displays.

4. Choose Install. Review the permissions required by the extension, and if you want to continue, choose Install again.

After you install the GitHub repositories extension, the next step is to connect your GitHub account to your CodeCatalyst space.

Note
You can't use empty or archived GitHub repositories with CodeCatalyst projects. The GitHub repositories extension isn't compatible with GitHub Enterprise Server repositories.

To connect your GitHub account to CodeCatalyst

1. On the Create project page in CodeCatalyst, if no GitHub account is connected, a prompt displays. Choose Connect GitHub account to go to the external site for GitHub.
2. Sign in to your GitHub account using your GitHub credentials, and then choose the account where you want to install Amazon CodeCatalyst.

   Tip
   If you have previously connected a GitHub account to the space, you will not be prompted to re-authorize. You will instead see a dialog box asking you where you would like to install the extension if you are a member or collaborator in more than one GitHub space, or the configuration page for the Amazon CodeCatalyst application if you only belong to one GitHub space. Configure the application for the repository access that you want to allow, and then choose Save. If the Save button is not active, make a change to the configuration, and then try again.

3. Choose whether you want to allow CodeCatalyst to access all current and future repositories, or choose the specific GitHub repositories you want to use in CodeCatalyst. The default option is to include all GitHub repositories in the GitHub account, including future repositories that will be accessed by CodeCatalyst.
4. Review the permissions given to CodeCatalyst, and then choose Install.
After connecting your GitHub account to CodeCatalyst, you will be able to link the GitHub repositories from that account to your CodeCatalyst projects.

**To create your project**

1. On the **Create project** page, from the **GitHub account** dropdown menu, do one of the following:
   - Choose a GitHub account that you have already connected to CodeCatalyst.
   - (Optional) If you don't see the GitHub account you want to use, choose **Connect a GitHub account** to go to the page for extensions in CodeCatalyst. For more information, see Using GitHub repositories in CodeCatalyst (p. 558).
2. In the **GitHub repository** dropdown menu, the GitHub repositories for your connected GitHub account are shown in the dropdown. Choose the GitHub repository that you want to link to your project.
3. In the **Name your project** text input field, enter the name that you want to assign to your project. The name must be unique within your space.
4. Choose **Create project**.

After your project is ready, you can add resources and tasks.

- To learn about the CI/CD workflows created with your project, see Getting started with workflows in CodeCatalyst (p. 134).
- To work with build actions similar to those in your new project that deploy build artifacts to an Amazon S3 bucket, see Building using workflows in CodeCatalyst (p. 138) and Tutorial: Upload artifacts to Amazon S3 (p. 140).
- To start with an empty project and work with deploying a similar serverless application with an AWS CloudFormation stack deployment, see Tutorial: Deploy a serverless application using AWS CloudFormation (p. 165).
- To add an issues planning board, see Issues in CodeCatalyst (p. 459).
- To view the project overview, project status, recent team activity, and assigned work, see Viewing a project (p. 60).
- To view source code or create a pull request, see Source repositories in CodeCatalyst (p. 79).
- To set up notifications that send status alerts for workflow run success or failure, see Managing notifications in Amazon CodeCatalyst (p. 71).
- To invite members to your project, see Managing project members (p. 62).
- To set up Dev Environments, see Dev Environments in CodeCatalyst (p. 113).

**Viewing a project**

From your CodeCatalyst space, you can view details of each project where you have project permissions.

To view a project, you must be a member of the project or have the **Space administrator** role for the space.

If you have not created a project yet, see Creating a project in Amazon CodeCatalyst (p. 58). You must have the **Space administrator** role for the space where you want to create a project.

- On the project overview, you can view project members, source repositories, workflow runs, open pull requests, project Dev Environments, and issues.
- Under project settings, you can view and manage project details, delete the project, invite new members to the project, manage project members, and configure notifications.
Viewing project tasks and Dev Environments

To view a summary of project tasks, such as open issues and pull requests that are assigned to you or created by you, and the project's associated Dev Environments, use the console.

To view a project, you must be a member of the project or have the Space administrator role for the space.

To view your source repositories, workflow runs, issues, pull requests, Dev Environments, and issues

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space with the project you want to view. Under Projects, choose your project.
3. In the navigation pane, choose Overview.
4. View the project tasks assigned to you and created by you.
   - View the Members + View all list to view a list of the project members.
   - View the Repositories card to view the source repositories that are associated with the project.
   - View the Workflow runs card to view the workflows that are associated with the project.
   - View the Open pull requests card to view a summary of code repository status, in addition to pull requests assigned to you and created by you.
   - View the My Dev Environments card to view a summary of the Dev Environments associated with the project.
   - View the Issues card to view a summary of your assigned tasks or tasks that you created.

Viewing all projects

In the Projects list for your space, you can view all projects where you have permissions.

To view a summary of project tasks, such as open issues and pull requests that are assigned to you or created by you, and the project's associated Dev Environments, use the console.

To view a project, you must be a member of the project or have the Space administrator role for the space.

To view your source repositories, workflow runs, issues, pull requests, Dev Environments, and issues

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space with the project you want to view. Under Projects, choose your project.
3. In the navigation pane, choose Project settings.
4. View the project name, path, project ID, and description.

Viewing project settings

In the Project settings, you can view project members, source repositories, workflow runs, open pull requests, project Dev Environments, and issues.

To view a summary of project tasks, such as open issues and pull requests that are assigned to you or created by you, and the project's associated Dev Environments, use the console.
To view your source repositories, workflow runs, issues, pull requests, Dev Environments, and issues

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space with the project you want to view. Under Projects, choose your project.
3. In the navigation pane, choose Project settings.
4. View the project name, path, project ID, and description.

Changing to a different project in CodeCatalyst

To change to a different project, use the console to choose from a list of projects you have access to.

To change to a different project

1. In the CodeCatalyst console, choose the project selector at the top.
2. Expand the drop-down and choose the project you want to navigate to.

Deleting a project in Amazon CodeCatalyst

You can delete a project to remove all access to the project’s resources. You must have the Space administrator or Project administrator role to delete a project. Once you have deleted a project, project members will be unable to access project resources, and any workflows that are prompted by third-party source repositories will be stopped.

To delete your project

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space with the project you want to view. Under Projects, choose your project.
3. In the navigation pane, choose Project settings.
4. Choose Delete project.
5. Enter delete to confirm the deletion.
6. Choose Delete project.

Managing project members

You can manage the members in your projects using the Amazon CodeCatalyst console. You can add or remove users, manage current members’ roles, send invitations to join your project, and cancel invitations that have not yet been accepted.

Viewing members in a project

When you add a user to your project, you assign a role that grants project permissions as follows:

- The Project administrator role has all permissions in a project. Only assign this role to users who need to administer every aspect of a project, including editing project settings, managing project permissions, and deleting the project. For more information, see Project administrator role (p. 485).
- The Contributor role has the permissions required to work in a project. Assign this role to those users who need to work with code, workflows, issues, and actions in a project. For more information, see Contributor role (p. 488).
You do not need to invite a user with the **Space administrator** role to your project because they already have implicit access to all projects in the space.

When you invite a user to your project (without assigning the **Space administrator** role), the user will show in the **Project members** table under projects and in the **Project members** table under spaces.

**To view users and roles in your space**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the space with the project you want to view. Under **Projects**, choose your project.
3. In the navigation pane, choose **Project settings**.
4. Choose the **Members** tab.

The **Project members** table shows all members that have a role in a project.

**Tip**
If you have the **Space administrator** role, you can view which projects you have been directly invited to. Navigate to **Project settings** for the project, and then choose **My projects**.

The **Space administrators** table shows users with the **Space administrator** role. These users are automatically (implicitly) assigned to all projects in the space and do not have a role in a project.

In the **Status** column, the following are valid values:

- **Invited** – CodeCatalyst sent the invitation but the user has not yet accepted or declined.
- **Member** – The user accepted the invitation.

**Topics**
- **Inviting a user to your project** (p. 63)
- **Canceling an invitation** (p. 64)
- **Removing a user from your project** (p. 64)
- **Accepting or declining an invitation for a project** (p. 65)

---

**Inviting a user to your project**

You can use the console to invite users to your project. You can invite members of your space or add names from outside your space.

To invite users to your project, you must be signed in with the **Project administrator** or **Space administrator** role.

You do not need to invite a user with the **Space administrator** role to your project because they already have implicit access to all projects in the space.

When you invite a user to your project (without assigning the **Space administrator** role), the user will show in the **Project members** table under projects and in the **Project members** table under spaces.

**To invite a member to your project from the Project settings tab**

1. Navigate to your project.

**Tip**
You can choose which project to view in the top navigation bar.

2. In the navigation pane, choose **Project settings**.
3. Choose the Members tab.
4. In Project members, choose Invite new member.
5. Type the new member’s email address, choose the role for this member, and then choose Invite. For more information about roles, see Working with roles in Amazon CodeCatalyst (p. 476).

To invite a member to your project from the Project overview page
1. Navigate to your project.
   Tip
   You can choose which project to view in the top navigation bar.
2. Choose the Members + button.
3. Type the new member’s email address, choose the role for this member, and then choose Invite. For more information about roles, see Working with roles in Amazon CodeCatalyst (p. 476).

Canceling an invitation

If you recently sent an invitation, you can cancel it as long as the invitation hasn’t yet been accepted.

To manage project invitations, you must have the Project administrator or Space administrator role.

To cancel a project member invitation
1. Navigate to the project where you have sent an invitation that you want to cancel.
2. In the navigation pane, choose Project settings.
3. View the Members tab and verify that the member has a status of Invited.
   Note
   You can only cancel an invitation that has not yet been accepted.
4. Choose the option next to the row with the invited member, and then choose Cancel invitation.
5. A confirmation window displays. Choose Cancel invitation to confirm.

Removing a user from your project

You can use the console to remove a user from your project.

To remove a user from your project, you must be signed in with the Project administrator or Space administrator role.

   Note
   Removing a user from all projects within a space automatically removes the user from that space.

To remove a user from a project
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the space with the project you want to view. Under Projects, choose your project.
3. In the navigation pane, choose Project settings.
4. Choose the Members tab.
5. Choose the selector next to the profile you want to remove, and then choose Remove.
6. Confirm that you want to remove the user, and then choose Remove.
Accepting or declining an invitation for a project

You might receive an email invitation to join an Amazon CodeCatalyst project. You can accept or decline the invitation.

To accept or decline an invitation

1. Open the invitation email.
2. Choose the project link in the email.
3. Choose Accept or Decline.

If you choose Decline, an email is sent to the project management account notifying them that you declined the invitation.

Quotas for projects in CodeCatalyst

The following table describes quotas and limits for projects in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see Quotas for CodeCatalyst (p. 620).

<table>
<thead>
<tr>
<th>Quota Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of projects per space</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of projects to which a user can belong</td>
<td>1,000</td>
</tr>
<tr>
<td>Maximum number of members that can belong to a project</td>
<td>10,000</td>
</tr>
<tr>
<td>Project names</td>
<td></td>
</tr>
<tr>
<td>Must be unique within a space</td>
<td></td>
</tr>
<tr>
<td>Must be between 3 and 63 characters</td>
<td></td>
</tr>
<tr>
<td>Names are case sensitive</td>
<td></td>
</tr>
<tr>
<td>Must begin with an alphanumeric character</td>
<td></td>
</tr>
<tr>
<td>Valid characters: A-Z, a-z, 0-9, spaces, and _</td>
<td></td>
</tr>
<tr>
<td>(underscore) - (hyphen)</td>
<td></td>
</tr>
<tr>
<td>Project names cannot contain any of the following</td>
<td></td>
</tr>
<tr>
<td>characters: ! ? @ # $ % ^ &amp; * ( ) + = { } [ ] \ / &gt; &lt; ~ ` ' &quot; ; :</td>
<td></td>
</tr>
<tr>
<td>Project descriptions</td>
<td></td>
</tr>
<tr>
<td>Can be up to 200 characters</td>
<td></td>
</tr>
<tr>
<td>Valid characters: A-Z, a-z, 0-9, spaces, and _</td>
<td></td>
</tr>
<tr>
<td>(underscore) - (hyphen)</td>
<td></td>
</tr>
<tr>
<td>Project descriptions are optional.</td>
<td></td>
</tr>
</tbody>
</table>

Project blueprint reference

When you create a project using a blueprint, CodeCatalyst creates a full project with a source repository, sample source code, CI/CD workflows, build and test reports, and integrated issue tracking tools. A project blueprint uses code to provision cloud infrastructure, resources, and sample source artifacts for different types of applications and frameworks.

For more information, see Creating a project in Amazon CodeCatalyst (p. 58). You must be the Space administrator to create a project.

Topics
Available blueprints

<table>
<thead>
<tr>
<th>Blueprint name</th>
<th>Blueprint description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET Core web API</td>
<td>This blueprint creates a .NET 6 ASP.NET Core web API application. The blueprint uses the AWS Deployment tool for .NET and provides an option to configure Amazon Elastic Container Service, AWS App Runner, or AWS Elastic Beanstalk as a deployment target.</td>
</tr>
<tr>
<td>AWS Glue ETL</td>
<td>This blueprint creates a sample extract transform load (ETL) reference implementation using AWS CDK, AWS Glue, AWS Lambda, and Amazon Athena to convert comma-separated values (CSVs) to Apache Parquet.</td>
</tr>
<tr>
<td>Clickstream real-time analytics</td>
<td>This blueprint creates an event-driven application architecture for stream processing. The blueprint showcases the best practices using Amazon Kinesis Data Streams, AWS Glue, and Amazon Athena.</td>
</tr>
<tr>
<td>DevOps deployment pipeline</td>
<td>This blueprint creates a deployment pipeline using the AWS Deployment Pipeline Reference Architecture that deploys a reference application to AWS across multiple stages.</td>
</tr>
<tr>
<td>Java API with AWS Fargate</td>
<td>This blueprint creates a containerized web service project. The project uses AWS Copilot CLI to build and deploy a containerized Spring Boot Java web service backed by Amazon DynamoDB on Amazon ECS. The project deploys a containerized app to an Amazon ECS cluster on AWS Fargate serverless compute. The app stores data in a DynamoDB table. After your workflow runs successfully, the sample web service is publicly available through the Application Load Balancer.</td>
</tr>
<tr>
<td>Modern three-tier web application</td>
<td>This blueprint generates code in Python for the application layer and Vue front-end framework to build and deploy a well-architected 3-tier modern web application.</td>
</tr>
<tr>
<td>.NET serverless application</td>
<td>This blueprint creates AWS Lambda functions using .NET CLI Lambda tools. The blueprint provides options for the AWS Lambda functions, including a choice of C# or F#.</td>
</tr>
<tr>
<td>Node.js API with AWS Fargate</td>
<td>This blueprint creates a containerized web service project. The project uses AWS Copilot CLI to build and deploy a containerized Express/Node.js web service on Amazon Elastic Container Service. The</td>
</tr>
</tbody>
</table>
### Finding project blueprint information

Several project blueprints are available in CodeCatalyst. For each blueprint, there is an accompanying summary and README file. The summary describes the resources that are installed by the blueprint, while the README file explains the blueprint in detail and provides instructions on how to use it.

<table>
<thead>
<tr>
<th>Blueprint name</th>
<th>Blueprint description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serverless Application Model (SAM)</td>
<td>This blueprint creates a project that uses a serverless application model (SAM) to create and deploy an API. You can choose SDK for Java, TypeScript, or SDK for Python as the programming language.</td>
</tr>
<tr>
<td>Serverless image handler</td>
<td>This blueprint creates an application for high-speed image processing without reducing image quality.</td>
</tr>
<tr>
<td>Serverless RESTful microservice</td>
<td>This blueprint creates a REST API that uses AWS Lambda and Amazon API Gateway with a To Do service reference. You can choose SDK for Java, TypeScript, or SDK for Python as the programming language.</td>
</tr>
<tr>
<td>Single-page application</td>
<td>This blueprint creates a single-page application (SPA) that uses React, Vue, and Angular frameworks. For hosting, choose from AWS Amplify Hosting or Amazon CloudFront and Amazon S3.</td>
</tr>
<tr>
<td>Static website</td>
<td>This blueprint creates a static website using <strong>Hugo</strong> or <strong>Jekyll</strong> static site generators. Static site generators use text input files (such as Markdown) to generate static web pages. They are ideal for rarely-changing, informative content, such as product pages, documentation, and blogs. The blueprint uses the AWS CDK to deploy static web pages to either AWS Amplify or Amazon S3 + CloudFront.</td>
</tr>
<tr>
<td>To Do web application</td>
<td>This blueprint creates a To Do serverless web application with frontend and backend components. You can choose SDK for Java, TypeScript, or SDK for Python as the programming language.</td>
</tr>
<tr>
<td>Video-on-demand web service</td>
<td>This blueprint creates a video-on-demand service that provides the ability to take in, transcode, and deliver content. The blueprint uses AWS Lambda, Amazon S3, Amazon CloudWatch, and AWS Elemental MediaConvert.</td>
</tr>
</tbody>
</table>
Working with notifications in CodeCatalyst

You can set up notifications to monitor your projects and resources in CodeCatalyst. Users can choose the project events about which they want to receive emails in any project where they are a member. You can also choose to configure notifications sent to an entire team in a team messaging application, such as Slack, by configuring access between a CodeCatalyst space and a Slack workspace, and then configuring notifications for a project to be sent to one or more channels in that Slack workspace. Once you’ve configured access between a CodeCatalyst space and a Slack workspace, project members will also have the option to add their own Slack member IDs so that they can be notified directly about CodeCatalyst events in connected Slack workspaces and channels.

Note
The set of project events that can be sent to Slack are not the same set of events that users can choose to be notified about in email.

Topics
- How do notifications work? (p. 68)
- Getting started with Slack notifications (p. 69)
- Managing notifications in Amazon CodeCatalyst (p. 71)

How do notifications work?
You can set up your project to provide notifications to your team messaging application, such as Slack.

What permissions are necessary for notifications?
Any project member can configure, view, update, or delete notification settings for a channel in CodeCatalyst. However, only users with the Space administrator role can add or delete Slack workspaces. All users can configure what project events they want to receive emails about for the projects they belong to in CodeCatalyst.

What CodeCatalyst events can I configure notifications about?
You can configure CodeCatalyst to deliver notifications to one or more Slack channels about workflow events. Once notifications have been configured between a CodeCatalyst project and Slack, project users can choose to add their own Slack member ID in order to receive direct messages in Slack channels about CodeCatalyst events. Users who add their Slack member IDs will receive direct mentions to their IDs in the Slack channels configured for their projects, helping raise awareness about events they care about.

You can also choose what events you want to receive emails about. These emails are sent to the email address configured for your AWS Builder ID.

How are notifications surfaced?
You can configure CodeCatalyst to deliver notifications to one or more Slack channels. You need to authorize CodeCatalyst to grant permissions to access your Slack workspace. Once the authorization is provided, CodeCatalyst can deliver notifications to the Slack channels you configure. If a project member chooses to add their Slack member ID, they can receive mentions about CodeCatalyst events in the Slack channels configured for that project.

How do I set up notifications?
Email notifications are configured as part of CodeCatalyst. Project users can choose what events they’d like to receive emails about in their My settings page.
To set up Slack notifications for your project resources, you must complete the following high-level tasks.

**To set up notifications (high-level tasks)**

1. In CodeCatalyst, you **set up a connection** between CodeCatalyst and a messaging client, such as Slack. Once a Slack workspace is connected, it will be available to all projects in the space.
   
   **Note**
   Only users with a Space administrator role can add or delete a Slack workspace.

2. In your project in CodeCatalyst, **add the channel** where you want your team to receive notifications.

3. In CodeCatalyst, you **turn on notifications** for various events, such as workflow run failure, and specify the channel where you want them sent.

For detailed steps, see [Getting started with Slack notifications (p. 69)](#).

Once notifications have been configured between a CodeCatalyst space and Slack, users can choose to add their own Slack member IDs to receive direct messages about CodeCatalyst events in the Slack channels configured for their projects.

**Getting started with Slack notifications**

After you create a project, you can set up Slack notifications that help your team to monitor project resources.

These steps walk you through setting up Slack notifications for the first time in CodeCatalyst. If you have already configured notifications, see [Managing notifications in Amazon CodeCatalyst (p. 71)](#).

**Note**

The set of project events that can be sent to notification channels are not the same set of events that users can choose to be notified about in email. For more information, see [Managing notifications in Amazon CodeCatalyst (p. 71)](#).

**Topics**

- [Prerequisites (p. 69)](#)
- [Step 1: Connect CodeCatalyst to your Slack workspace (p. 70)](#)
- [Step 2: Add your Slack channel to CodeCatalyst (p. 70)](#)
- [Step 3: Test notifications from CodeCatalyst to Slack (p. 71)](#)
- [Step 4: Next steps (p. 71)](#)

**Prerequisites**

Before you begin, you need the following:

- A CodeCatalyst space. For information about creating a CodeCatalyst space and signing in for the first time, see [Setting up CodeCatalyst (p. 7)](#).
- A CodeCatalyst project. For more information, see [Creating a project in Amazon CodeCatalyst (p. 58)](#).
- A CodeCatalyst account with the **Project administrator** or **Space administrator** role. For more information, see [Working with roles in Amazon CodeCatalyst (p. 476)](#).
- A Slack account and Slack workspace that can be accessed by CodeCatalyst.

A Slack channel where CodeCatalyst will send notifications. The channel can be public or private.
Step 1: Connect CodeCatalyst to your Slack workspace

Only users with the **Space administrator** role can add or delete Slack workspaces. Adding or deleting a Slack workspace affects all projects in the space. To establish the connection between CodeCatalyst and Slack, CodeCatalyst performs a secure OAuth authentication handshake with your Slack workspace.

Use the following instructions to connect CodeCatalyst to your Slack workspace.

**Note**
This only needs to be done once for each Slack workspace. You can then set up notifications by Slack channel.

**To connect CodeCatalyst to your Slack workspace**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your project.
3. In the navigation pane, choose **Project settings**.
4. Choose the **Notifications** tab.
5. Choose **Configure notifications**.
6. Choose **Connect to Slack workspace**.
7. Read the dialog box contents, and then choose **Connect to Slack workspace**.
8. On the **AWS Chatbot** message:
   a. In the upper right, choose the Slack workspace that contains your channel.
   b. Choose **Allow**.

You are returned to the CodeCatalyst console.

9. Continue to **Step 2: Add your Slack channel to CodeCatalyst** (p. 70).

Step 2: Add your Slack channel to CodeCatalyst

You need the Slack channel ID to add your channel to CodeCatalyst.

**To get your Slack channel ID**

1. Sign in to Slack. For more information, see **Sign in to Slack**.
2. Go to the Slack workspace that contains the channel where you want notifications to go. For more information, see **Switch between Slack workspaces** or **Sign in to additional Slack workspaces**.
3. In the navigation pane, open the context (right click) menu for the channel where you want notifications to go, and choose **Open channel details**.
   The channel ID is displayed at the bottom of the dialog box.
4. Copy the **Channel ID** value. You'll need it in the next step.

Using the channel ID you just copied, you can now connect your Slack channel to CodeCatalyst.

**To add your Slack channel to CodeCatalyst**

1. Before you begin, if your Slack channel is private, add the AWS Chatbot app to the channel as follows:
   a. In your Slack channel's message box, enter `@aws` and choose **aws app** from the dialog box.
b. Press Enter.

A Slackbot message appears, indicating that AWS Chatbot is not in the private channel.

c. Choose Invite Them to invite AWS Chatbot to the channel.

2. In the CodeCatalyst console, choose Next.

3. In Channel ID, paste the Slack channel ID you obtained earlier.

4. In Channel name, enter a name. We recommend using the Slack channel name.

5. Choose Next.

6. In Select notification events, choose the type of event you want to receive notifications for.

7. Choose Finish.

**Step 3: Test notifications from CodeCatalyst to Slack**

After your project is configured to send notifications for workflow status, you can view your notifications in Slack.

**To view your notifications in Slack**

1. In your CodeCatalyst project, start a workflow manually (p. 314) in order to complete a workflow run and receive a status notification when the run finishes.

2. In Slack, view the channel you set up for notifications. Your notifications show the latest status from each workflow run, and whether it failed or succeeded.

**Step 4: Next steps**

Once a Slack workspace is configured for your CodeCatalyst space, you can add additional Slack channels existing CodeCatalyst projects, and add them for new projects after you create them. You can also let project users know that they can configure personal Slack notifications for their Slack member IDs, and configure the events for which they'll receive emails. For more information, see Managing notifications in Amazon CodeCatalyst (p. 71).

**Managing notifications in Amazon CodeCatalyst**

You can configure CodeCatalyst to send notifications about events in your project. You can send notifications to messaging clients such as Slack channels. Project users can choose what project events they will be notified about by emails sent to the email address configured for their profile.

**Note**

The set of project events that can be sent to notification channels are not the same set of events that users can choose to be notified about in email.

**Topics**

- Managing notifications sent directly to you (p. 71)
- Managing notifications sent to channels (p. 72)

**Managing notifications sent directly to you**

You can choose to have email notifications sent to you about events in any project where you are a member. These emails will be sent to the email address configured in your AWS Builder ID. By default, you will receive emails about all project events for which emails can be sent.
If a project has been configured to send notifications to a Slack channel, you can add your Slack member ID to receive direct mentions about CodeCatalyst events in that Slack channel. This can help raise your awareness of events happening in the projects where you have a role.

**To configure email notifications for project events**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the top menu bar, choose your profile badge, and then choose My settings. The CodeCatalyst My settings page opens.
   - **Tip**
     You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.
3. In Email notifications, find the project in the list where you want to configure email notifications, and choose Edit.
4. Select the events for which you want to receive emails, and then choose Save.

**To configure personal Slack notifications**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the top menu bar, choose your profile badge, and then choose My settings. The CodeCatalyst My settings page opens.
   - **Tip**
     You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.
3. In Personal Slack notifications, choose Connect Slack ID, and then choose Connect to Slack workspace. A separate window will open.
   - **Tip**
     This option is not configurable unless a user with the Space administrator role has added a Slack workspace for your CodeCatalyst space. For more information, see Getting started with Slack notifications (p. 69) and Managing notifications sent to channels (p. 72).
4. In the permissions request window, make sure that the name of the workspace matches the Slack workspace configured for the CodeCatalyst space. Choose Allow to allow AWS Chatbot access to the workspace. The window will close, and the Slack workspace will show the Connection status as Connected.
   - **Tip**
     If the connection status does not change, check to see if an error occurred connecting the Slack workspace. You might have to scroll up to see the error.
5. To stop receiving personal Slack notifications, choose the connected Slack workspace, and then choose Disconnect Slack ID.

**Managing notifications sent to channels**

You can choose to add and manage notifications about project events in CodeCatalyst sent to team resources, such as a team Slack channel. By doing this, you can help ensure that your entire team is aware of important events, such as when a workflow run fails.

**Note**

Any member of a project can manage notifications sent to channels for that project. However, only users with the Space administrator role can add or delete Slack workspaces.

**Adding a notification channel for a project**

You can add a channel where you want to receive notifications, such as your team's Slack channel.
Managing notifications

To add a Slack channel for notifications

1. If you’re adding your first Slack channel, see instead Getting started with Slack notifications (p. 69).
   
   After setting up your first channel, return to this procedure to set up additional channels.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. Navigate to your project.
4. In the navigation pane, choose Project settings.
5. Choose the Notifications tab.
6. Choose Add channel.
7. Choose Choose workspace, and then select the Slack workspace that contains the channel where you want to send notifications.
   
   If your Slack workspace is not in the list, you can add it by following the instructions in Getting started with Slack notifications (p. 69).
8. Before entering a Channel ID, if the Slack channel you want to add is private, complete these steps:
   a. In your Slack channel's message box, enter @aws and choose aws app from the pop-up.
   b. Press Enter.
      
      A Slackbot message appears, indicating that AWS Chatbot is not in the private channel.
   c. Choose Invite Them to invite AWS Chatbot to the channel.
9. In CodeCatalyst's Channel ID field, enter the Slack channel ID. To find the ID, go to Slack, and in the navigation pane, right-click the channel and choose Open channel details.
   
   The channel ID is displayed at the bottom of the dialog box.
10. In Channel name, enter a name. We recommend using the Slack channel name.
11. In Select notification events, choose the type of event you want to receive notifications for.
12. Choose Add.

Editing notifications for a notification channel

You can change which channels notifications go to, and you can turn off specific notifications altogether.

To edit notifications

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project.
3. In the navigation pane, choose Project settings.
4. Choose the Notifications tab.
5. Choose Edit notifications.
6. Do one of the following:
   a. To send a notification to a specific channel, choose the channel from the drop-down list.
   b. To turn off a notification globally, choose the toggle next to the notification.
   c. To stop sending a notification to a specific channel, choose the X on the channel.
7. Choose Save.
Removing a channel

To remove a channel

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project. In the navigation pane, choose Project settings.
3. On the Project settings page, choose the Notifications tab.
4. Choose the indicator next to the channel you want to remove and then choose Remove channel. When prompted, choose Ok in the confirmation window.

Troubleshooting projects and blueprints

This section can help you troubleshoot some common issues you might encounter while working with projects and blueprints in Amazon CodeCatalyst.

Java API with AWS Fargate blueprint missing dependencies for apache-maven-3.8.6

Issue: For a project created from the Java API with AWS Fargate blueprint, the workflow fails with an error for missing apache-maven-3.8.6 dependencies. The workflow fails with output similar to the following example:

```
---> Running in 1851ce6f4d1b
[91m--2023-03-10 01:24:55 --  https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz
[0m[91mResolving dlcdn.apache.org (dlcdn.apache.org)...
[0m[91m151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443...
[0m[91mConnected.
[0m[91mHTTP request sent, awaiting response... [0m[91m404 Not Found
2023-03-10 01:24:55 ERROR 404: Not Found.
[0mThe command '/bin/sh -c wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz -P /tmp' returned a non-zero code: 8
[Container] 2023/03/10 01:24:55 Command failed with exit status 8
```

Solution: Update the blueprint Dockerfile using the following steps.

1. In the search bar, enter apache-maven-3.8.6 to locate the dockerfile inside the project created with the Java API with AWS Fargate blueprint.
2. Update the Dockerfile (/static-assets/app/Dockerfile) to use maven:3.9.0-amazoncorretto-11 as a base image and remove the dependency on the apache-maven-3.8.6 package.
3. (Recommended) We also recommend updating the Maven heap size to 6 GB.

Below is an example Dockerfile.

```
FROM maven:3.9.0-amazoncorretto-11 AS builder
```
Modern three-tier web application blueprint workflow **OnPullRequest** fails with permissions error for Amazon CodeGuru

**Issue:** When I try to run a workflow for my project, the workflow fails to run with the following message:

```
Failed at codeguru_codereview: The action failed during runtime. View the action's logs for more details.
```

**Solution:** One possible cause of this action failure might be due to missing permissions in the IAM role policy, where your version of the service role used by CodeCatalyst in the connected AWS account is missing required permissions for the `codeguru_codereview` action to run successfully. To fix this problem, either the service role must be updated with the required permissions, or you must change the service role used for the workflow to one that has the required permissions for Amazon CodeGuru and Amazon CodeGuru Reviewer. Using the following steps, find your role and update the role policy permissions in order to allow the workflow to run successfully.

**Note**
These steps apply for the following workflows in CodeCatalyst:

- The **OnPullRequest** workflow provided for projects created with the Modern three-tier web application blueprint in CodeCatalyst.
- Workflows added to projects in CodeCatalyst with actions that access Amazon CodeGuru or Amazon CodeGuru Reviewer.

Each project contains workflows with actions that use a role and environment provided by the AWS account connected to your project in CodeCatalyst. The workflow with the actions and their designated policy is stored in your source repository in the directory `.codecatalyst/workflows`. Modifying the workflow YAML is not required unless you are adding a new role ID to the existing workflow. For information about YAML template elements and formatting, see [Workflow definition reference](p. 354).

These are the high-level steps to follow to edit your role policy and verify the workflow YAML.

**To reference your role name in the workflow YAML and update the policy**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space. Navigate to your project.
3. Choose **CI/CD**, and then choose **Workflows**.
4. Choose the workflow titled `OnPullRequest`. Choose the **Definition** tab.

5. In the workflow YAML, in the `Role:` field under the `codeguru_codereview` action, make a note of the role name. This is the role with the policy that you will modify in IAM. The following example shows the role name.

6. Do one of the following:
   - (Recommended) Update the service role connected to your project with the required permissions for Amazon CodeGuru and Amazon CodeGuru Reviewer. The role will have a
name CodeCatalystWorkflowDevelopmentRole-**spaceName** with a unique identifier appended. For more information about the role and role policy, see Understanding the CodeCatalystWorkflowDevelopmentRole-**spaceName** service role (p. 516). Proceed to the next steps to update the policy in IAM.

**Note**

You must have AWS administrator access to the AWS account with the role and policy.

- Change the service role used for the workflow to one that has the required permissions for Amazon CodeGuru and Amazon CodeGuru Reviewer or create a new role with the required permissions.

7. Sign in to the AWS Management Console and open the IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/).

   In the IAM console, find the role from step 5, such as CodeCatalystPreviewDevelopmentRole.

8. In the role from step 5, change the permission policy to include the codeguru-reviewer:* and codeguru:* permissions. After adding these permissions, the permission policy should look similar to the following:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Action": [
           "cloudformation:*",
           "lambda:*",
           "apigateway:*",
           "ecr:*",
           "ecs:*",
           "ssm:*",
           "codedeploy:*",
           "s3:*",
           "iam:DeleteRole",
           "iam:UpdateRole",
           "iam:Get*",
           "iam:TagRole",
           "iam:PassRole",
           "iam:CreateRole",
           "iam:AttachRolePolicy",
           "iam:DetachRolePolicy",
           "iam:PutRolePolicy",
           "iam:CreatePolicy",
           "iam:DeletePolicy",
           "iam:CreatePolicyVersion",
           "iam:DeletePolicyVersion",
           "iam:PutRolePermissionsBoundary",
           "iam:DeleteRolePermissionsBoundary",
           "sts:AssumeRole",
           "elasticloadbalancing:DescribeTargetGroups",
           "elasticloadbalancing:DescribeListeners",
           "elasticloadbalancing:ModifyListener",
           "elasticloadbalancing:DescribeRules",
           "elasticloadbalancing:ModifyRule",
           "cloudwatch:DescribeAlarms",
           "sns:Publish",
           "sns:ListTopics",
           "codeguru-reviewer:*",
           "codeguru:*"
         ],
         "Resource": "*",
         "Effect": "Allow"
       }
     ]
   }
   ```
9. After you make the policy corrections, return to CodeCatalyst and start the workflow run again.

Still looking to solve your problem?

You can go to Amazon CodeCatalyst or fill out a Support Feedback form. In the Request information section, under How can we help you, include that you are an Amazon CodeCatalyst customer. Provide as much detail as possible so that we can most efficiently address your issue.
Source repositories in CodeCatalyst

CodeCatalyst source repositories are Git repositories hosted in Amazon CodeCatalyst. You can use source repositories in CodeCatalyst to securely store, version, and manage assets for a project.

Assets in a CodeCatalyst repository can include:

- documents
- source code
- binary files

CodeCatalyst also uses the source repository for a project to store configuration information for your project, such as workflow configuration files.

You can have more than one source repository in a CodeCatalyst project. For example, you might want to have separate source repositories for front-end source code, back-end source code, utilities, and documentation.

Here is one possible workflow for working with code in source repositories, pull requests, and Dev Environments in CodeCatalyst:

Mary Major creates a web application project in CodeCatalyst using a blueprint, which creates a source repository with sample code in it. She invites her friends Li Juan, Saanvi Sarkar, and Jorge Souza to work on the project with her. Li Juan looks at the sample code in the source repository and decides to make some quick changes to add a test to the code. Li creates a Dev Environment, chooses AWS Cloud9 as the IDE, and specifies a new branch, `test-code`. The Dev Environment opens. Li quickly adds the code, then commits and pushes the branch with the changes to the source repository in CodeCatalyst. Li then creates a pull request. As part of creating that pull request, Li adds Jorge Souza and Saanvi Sarkar as reviewers to ensure that the code is reviewed.

While reviewing the code, Jorge Souza remembers that he has his own project repository on GitHub that contains a prototype of the app they’re working on. He asks Mary Major to install and configure the extension that will allow him to link the GitHub repository to the project as an additional source repository. Mary reviews the repository on GitHub and works with Jorge to configure the GitHub extension so that he can link the GitHub repository as an additional source repository for the project.

CodeCatalyst source repositories support the standard functionality of Git and work with your existing Git-based tools. You can create and use personal access tokens (PATs) as an application-specific password when cloning and working with source repositories from a Git client or integrated development environments (IDEs). These PATs are associated with your CodeCatalyst user identity. For more information, see Managing personal access tokens in Amazon CodeCatalyst (p. 497).

CodeCatalyst source repositories support pull requests. This is a simple way for you and other project members to review and comment on code changes before you merge them from one branch to another. You can view the changes in the CodeCatalyst console and comment on lines of code.

Pushes to branches in a CodeCatalyst source repository can automatically start a run in a workflow, where changes can be built, tested, and deployed. If your source repository was created as part of a project using a project template, one or more workflows are configured for you as part of the project. You can add additional workflows for repositories at any time. The YAML configuration files for workflows in a project are stored in the source repositories configured in the source action for those workflows. For more information, see Getting started with workflows in CodeCatalyst (p. 134).
Source repository concepts

Here are some concepts to know about as you work with CodeCatalyst source repositories.

Projects

A project represents a collaborative effort in CodeCatalyst that supports development teams and tasks. After you have a project, you can add, update, or remove users and resources, customize your project dashboard, and monitor the progress of your team's work. You can have multiple projects within a space.

Source repositories are specific to the projects where you create or link them in a space. You can't share a repository between projects, and you can't link a repository to more than one project in a space. Users with the Contributor or Project administrator role in a project can interact with the source repositories associated with that project according to the permissions granted to those roles. For more information, see Working with roles in Amazon CodeCatalyst (p. 476).

Source repositories

A source repository is where you securely store code and files for your project. It also stores the version history of your files. By default, a source repository is shared with the other users in your CodeCatalyst project. You can have more than one source repository for a project. You can create source repositories for projects in CodeCatalyst, or you can choose to link an existing source repository hosted by another
service if that service is supported by an installed extension. For example, you can link a GitHub repository to a project after you install the **GitHub Repositories** extension. For more information, see Working with source repositories in CodeCatalyst (p. 89) and Quickstart: Using GitHub repositories in CodeCatalyst (p. 559).

### Dev Environments

A **Dev Environment** is a cloud-based development environment that you can use in CodeCatalyst to quickly work on the code stored in the source repositories of your project. The project tools and application libraries included in your Dev Environment are defined by a devfile in the source repository of your project. If you do not have a devfile in your source repository, a default devfile will be applied automatically. The default devfile includes tools for the most frequently used programming languages and frameworks. By default, a Dev Environment is configured to have a 2-core processor, 4 GB of RAM, and 16 GiB of persistent storage.

You can choose to clone an existing branch of your source repository into your Dev Environment, or you can choose to create a new branch as part of creating the Dev Environment.

### Personal access tokens (PATs)

A **personal access token (PAT)** is similar to a password. It is associated with your user identity for use across all spaces and projects in CodeCatalyst. You use PATs to access CodeCatalyst resources that include integrated development environments (IDEs) and Git-based source repositories. PATs represent you in CodeCatalyst and you can manage them in your user settings. A user can have more than one PAT. Personal access tokens only display once. As a best practice, be sure to store them securely on your local computer. By default, PATs expire after one year.

When working with integrated development environments (IDEs), PATs are the equivalent of a Git password. Provide the PAT when asked for a password when setting up your IDE to work with a Git repository. For more information about how to connect your IDE with a Git-based repository, see the documentation for your IDE.

### Branches

A **branch** is a pointer or reference to a commit in Git and in CodeCatalyst. You can use branches to organize your work. For example, you can use branches to work on a new or different version of files without affecting files in other branches. You can use branches to develop new features, store a specific version of your project, and more. A source repository can have one branch or many branches. When you create a project using a template, the source repository created for the project contains sample files in a branch called **main**. The **main** branch is the default branch for the repository.

### Default branches

Source repositories in CodeCatalyst have a default branch regardless of how you create them. If you choose to create a project using a template, the source repository created for that project includes a README.md file in addition to sample code, workflow definitions, and other resources. If you create a source repository without using a template, a README.md file is added for you as a first commit and a default branch is created for you as part of creating the repository. This default branch is named **main**. This default branch is the one used as the base or default branch in local repositories (repos) when users clone the repository. You can change which branch is used as the default branch. For more information, see View and change the default branch for a repository (p. 95).

You can't delete the default branch for a source repository. Search results only include results from the default branch.
Commits

A commit is a change to a file or set of files. In the Amazon CodeCatalyst console, a commit saves your changes and pushes them to a source repository. The commit includes information about the change, including the identity of the user who made the change, the time and date of the change, the commit title, and any message included about the change. For more information, see Working with commits in Amazon CodeCatalyst (p. 108).

In the context of a source repository in CodeCatalyst, commits are snapshots of the contents and changes to the contents of your repository. You can also add Git tags to commits, to identify specific commits.

Pull requests

A pull request is the primary way you and other users review, comment on, and merge code changes from one branch to another in a source repository. You can use pull requests to review code changes collaboratively for minor changes or fixes, major feature additions, or new versions of your released software. In a pull request, you can review the changes between the source and destination branches or the differences between revisions of those branches. You can add comments to individual lines of code changes as well as comments on the pull request overall.

Tip
While you are creating a pull request, the difference displayed is the difference between the tip of the source branch and the tip of the destination branch. Once the pull request has been created, the displayed difference will be between the revision of the pull request you choose and the commit that was the tip of the destination branch when you created the pull request. For more information about differences and merge bases in Git, see git-merge-base in the Git documentation.

Revisions

A revision is an updated version of a pull request. Each push to the source branch of a pull request creates a revision that contains the changes made in the commits included in that push. You can view the differences between revisions of a pull request in addition to the differences between the source and destination branches. For more information, see Working with pull requests in Amazon CodeCatalyst (p. 100).

Workflows

A workflow is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or actions, to take during a workflow run. A workflow also defines the events, or triggers, that cause the workflow to start. To set up a workflow, you create a workflow definition file using the CodeCatalyst console's visual or YAML editor.

Tip
For a quick look at how you might use workflows in a project, create a project with a blueprint. Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

A source repository can also store the configuration files and other information for workflows, notifications, issues, and other configuration information for the project. The configuration files are created and stored in the source repository when you create resources that require configuration files, or when you specify the repository as a source action for a workflow. If you create a project from a blueprint, you will have configuration files already stored in the source repository created for you as part of the project. This configuration information is stored in a folder named .codecatalyst in the default branch of your repository. Whenever you create a branch of the default branch, you create a copy of this folder and its configuration in addition to all the other files and folders in that branch.
Setting up for working with source repositories

When you work with source repositories in Amazon CodeCatalyst on your local machine, you can use Git either on its own or in a supported integrated development environment (IDE) to make code changes and push and pull your code. As a best practice, we recommend that you use the latest versions of Git and other software.

Note
If you use Dev Environments, you do not have to install Git. A recent version of Git is included in your Dev Environment.

Version compatibility information for CodeCatalyst

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Git</td>
<td>latest</td>
</tr>
</tbody>
</table>

Install Git

To work with files, commits, branches, and other information in source repositories from a Git client without an IDE, install Git on your local machine.

To install Git, we recommend websites such as Git Downloads.

Create a personal access token

To clone source repositories to your local machine or to your preferred IDE, you must create a personal access token (PAT).

To create a personal access token (PAT)

1. In the top menu bar, choose your profile badge, and then choose My settings.
   
   Tip
   You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.

2. In PAT name, enter a descriptive name for your PAT.

3. In Expiration date, leave the default date or choose the calendar icon to select a custom date. The expiration date defaults to one year from the current date.

4. Choose Create.

   You can also create this token when you choose Clone repository for a source repository.

5. Save the PAT secret in a secure location.

   Important
   The PAT secret only displays once. You cannot retrieve it after you close the window.

Getting started with CodeCatalyst source repositories and the Single-page application blueprint

Follow the steps in this tutorial to learn how to work with source repositories in Amazon CodeCatalyst.
The quickest way to get started working with source repositories in Amazon CodeCatalyst is to create a project using a template. When you create a project using a template, resources are created for you, including a source repository that includes sample code. You can use this repository and code example to learn how to:

- View a project's source repositories and browse their contents
- Create a Dev Environment with a new branch where you can work on code
- Change a file, and commit and push your changes
- Create a pull request and review your code changes with other project members
- See the workflow for your project automatically build and test the changes in the source branch of the pull request
- Merge your changes from your source branch into the destination branch and close the pull request
- See the merged changes automatically built and deployed

To get the most out of this tutorial, invite others to your project so you can work together on a pull request. You can also explore additional features in CodeCatalyst, such as creating issues and associating them with a pull request, or configuring notifications and getting alerts when the associated workflow runs. For a full exploration of CodeCatalyst, see Getting started tutorials (p. 17).

Creating a project using a blueprint

Creating a project is the first step in being able to work together. You can use a blueprint to create your project, which will also create a source repository with sample code and a workflow that will automatically build and deploy your code when you change it. In this tutorial, we’ll walk you through a project created with the Single-page application blueprint, but you can follow the procedures for any project with a source repository. Make sure to choose an IAM role or add an IAM role if you don't have one as part of creating the project. We recommend that you use the CodeCatalystWorkflowDevelopmentRole-spaceName service role for this project.

If you already have a project, you can skip ahead to Viewing the repositories for a project (p. 85).

Note
Only users with the Space administrator or Power user role can create projects in CodeCatalyst. If you do not have this role and you need a project to work on for this tutorial, ask someone with one of these roles to create a project for you and add you to the created project. For more information, see Working with roles in Amazon CodeCatalyst (p. 476).

To create a project with a blueprint

1. In the CodeCatalyst console, navigate to the space where you want to create a project.
2. On the space dashboard, choose Create project.
3. Choose Start with a blueprint.
4. Choose a blueprint, and then choose Next.
5. Under Give a name to your project, enter the name that you want to assign to your project and its associated resource names. The name must be unique within your space.
6. Under Project resources, configure common project parameters.
7. (Optional) To view definition files with updates based on the project parameter selections you made, choose View code or View workflow from Generate project preview.
8. (Optional) Choose View details from the blueprint's card to view specific details about the blueprint, such as an overview of the blueprint's architecture, required connections and permissions, and the kind of resources the blueprint creates.
9. Choose Create project.

For more information about project blueprints, see Project blueprint reference (p. 65).
Viewing the repositories for a project

As a member of a project, you can view the source repositories for the project. You can also choose to create additional repositories. If someone with the **Space administrator** role has installed and configured the **GitHub repositories** extension, you can also add links to GitHub repositories in the GitHub accounts configured for the extension. For more information, see Creating a source repository (p. 89) and Quickstart: Using GitHub repositories in CodeCatalyst (p. 559).

**Note**

For projects created with the Single-page application blueprint, the default name for the source repository that contains the sample code is `spa-app`.

To navigate to the source repositories for a project

1. Navigate to your project, and do one of the following:
   - On the summary page for your project, choose the repository you want from the list, and then choose View repository.
   - In the navigation pane, choose Code, and then choose Source repositories. In Source repositories, choose the name of the repository from the list. You can filter the list of repositories by typing part of the repository name in the filter bar.

2. On the home page for the repository, view the contents of the repository and information about the associated resources such as the number of pull requests and workflows. By default, the contents for the default branch are shown. You can change the view by choosing a different branch from the drop-down list.

The overview page for the repository includes information about the workflows and pull requests that are configured for the branches of this repository and its files. If you just created the project, the initial workflows to build, test, and deploy the code will still be running, as they take a few minutes to complete. You can view the related workflows and their status by choosing the number beneath Related workflows, but this opens the Workflows page in CI/CD. For this tutorial, stay on the overview page and explore the code in the repository. The contents of the README.md file are rendered on this page below the repository files. In Files, the contents of the default branch are shown. You can change the file view to show the contents of another branch if you have one. The .codecatalyst folder contains code used for other parts of the project, such as workflow YAML files.

To view the content of folders, choose the arrow next to the folder name to expand it. For example, choose the arrow next to src to view the files for the single-page web application contained in that folder. To view the contents of a file, choose it from the list. This will open View files, where you can browse the contents of multiple files. You can edit single files in the console as well, but to edit multiple files, you'll want to create a Dev Environment.

Creating a Dev Environment

You can add and change files in a source repository in the Amazon CodeCatalyst console. However, to work effectively with multiple files and branches, we recommend using a Dev Environment or cloning the repository to your local computer. In this tutorial, we'll create an AWS Cloud9 Dev Environment with a branch named develop. You can choose a different branch name, but by naming the branch develop, a workflow will automatically run to build and test your code when you create a pull request later in this tutorial.
Tip
If you decide to clone a repository locally instead of or in addition to using a Dev Environment, make sure that you have Git on your local computer or that your IDE includes Git. For more information, see Setting up for working with source repositories (p. 83).

To create a Dev Environment with a new branch
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to create a Dev Environment.
3. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, choose Source repositories, and choose the repository for which you want to create a Dev Environment.
4. On the repository home page, choose Create Dev Environment.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Choose the repository to clone, choose Work in new branch, enter a branch name into the Branch name field, and choose a branch off of which to create the new branch from the Create branch from drop-down menu.
8. Optionally, choose the Dev Environment configuration edit button to edit the Dev Environment's compute, storage, or timeout configuration.
9. Choose Create. While your Dev Environment is being created, the Dev Environment status column will display Starting, and the status column will display Running once the Dev Environment has been created. A new tab will open with your Dev Environment in the IDE of your choice. You can edit code and commit and push your changes.

Once you've created the Dev Environment, you can edit files, commit your changes, and push your changes to the test branch. For this tutorial, edit the content between the <p> tags in App.tsx file in the src folder to change the text that's displayed on the webpage. Commit and push your change, and then return to the CodeCatalyst tab.

To make and push a change from an AWS Cloud9 Dev Environment
1. In AWS Cloud9, expand the side navigation menu to browse the files. Expand src, and open App.tsx.
2. Make a change the text inside the <p> tags.
3. Save the file, and then commit and push your changes by using the Git menu. Alternatively, in the terminal window, commit and push your changes with the git commit and git push commands.

```bash
git commit -am "Making an example change"
git push
```

Tip
You might need to change directories in the terminal to the Git repository directory before you can successfully run the Git commands.

Creating a pull request

You can use pull requests to review code changes collaboratively for minor changes or fixes, major feature additions, or new versions of your released software. In this tutorial, you will create a pull request to review the changes you made to the test branch compared to the main branch. Creating a pull request in a project created with a template will also start a run of its associated workflows, if any.
To create a pull request

1. Navigate to your project.
2. Do one of the following:
   - In the navigation pane, choose Code, choose Pull requests, and then choose Create pull request.
   - On the repository home page, choose More, and then choose Create pull request.
   - On the project page, choose Create pull request.
3. In Source repository, make sure that the specified source repository is the one that contains the committed code. This option only appears if you did not create the pull request from the repository's main page.
4. In Destination branch, choose the branch to merge the code into after it is reviewed.
5. In Source branch, choose the branch that contains the committed code.
6. In Pull request title, enter a title that helps other users understand what needs to be reviewed and why.
7. (Optional) In Pull request description, provide information such as a link to issues or a description of your changes.
8. (Optional) In Issues, choose Link issues, and then either choose an issue from the list or enter its ID. To unlink an issue, choose the unlink icon.
9. (Optional) In Required reviewers, choose Add required reviewers. Choose from the list of project members to add them. Required reviewers must approve the changes before the pull request can be merged into the destination branch.
   
   **Note**
   You cannot add a reviewer as both a required reviewer and an optional reviewer. You cannot add yourself as a reviewer.
10. (Optional) In Optional reviewers, choose Add optional reviewers. Choose from the list of project members to add them. Optional reviewers do not have to approve the changes as a requirement before the pull request can be merged into the destination branch.
11. Review the differences between the branches. The difference displayed in a pull request is the changes between the revision in the source branch and the merge base, which is the head commit of the destination branch at the time the pull request was created. If no changes display, the branches might be identical, or you might have chosen the same branch for both the source and the destination.
12. When you are satisfied that the pull request contains the code and changes you want reviewed, choose Create.
   
   **Note**
   After you create the pull request, you can add comments. Comments can be added to the pull request or to individual lines in files as well as to the overall pull request. You can add links to resources, such as files, by using the @ sign followed by the name of the file.

You can view information about associated workflows started by the creation of this pull request by choosing Overview and then reviewing the information in the Pull request details area under Workflow runs. To view the workflow run, choose the run.

**Tip**
If you named your branch something other than develop, a workflow will not automatically run to build and test your changes. If you want to configure that, edit the YAML file for the onPullRequestBuildAndTest workflow. For more information, see Creating, editing, and deleting a workflow (p. 246).

You can comment on this pull request and ask other project members to comment on it. You can also choose to add or change optional or required reviewers. You can choose to make more changes to the source branch for the repository, and see how those committed changes create revisions for the pull
Merging a pull request

Once a pull request has been reviewed and has received approvals from required reviewers, you can merge its source branch to the destination branch in the CodeCatalyst console. Merging a pull request will also start a run of the changes through any workflows associated with the destination branch. In this tutorial, you'll merge the test branch into main, which will start a run of the `onPushToMainDeployPipeline` workflow.

To merge a pull request (console)

1. In Pull requests, choose the pull request you created in the previous step. In the pull request, choose Merge.
2. Choose from the available merge strategies for the pull request. Optionally select or deselect the option to delete the source branch after merging the pull request, and then choose Merge. After the merge completes, the status of the pull request changes to Merged and no longer appears in the default view of pull requests. The default view shows pull requests with a status of Open. You can still view a merged pull request, but you cannot approve it or change its status.

   Note
   If the Merge button is not active, or you see the Not mergeable label, either a required reviewer has not yet approved the pull request, or the pull request cannot be merged in the CodeCatalyst console. A reviewer who has not approved a pull request is indicated by a clock icon in Overview in the Pull request details area. If all required reviewers have approved the pull request but the Merge button is still not active, you might have a merge conflict. You can resolve merge conflicts for the destination branch in a Dev Environment, push the changes, and then merge the pull request, or you can resolve conflicts and merge locally, and then push the commit that contains the merge to CodeCatalyst. For more information, see [Merging a pull request (Git)](p. 107) and your Git documentation.

Viewing the deployed code

Now it's time to view the originally deployed code that was in the default branch, and your merged changes once they are automatically built, tested, and deployed. To do so, you can return to the overview page for the repository and choose the number next to the related workflows icon, or in the navigation pane, choose CI/CD, and then choose Workflows.

To view the deployed code

1. In Workflows, in onPushToMainDeployPipeline, expand Recent runs.

   Note
   This is the default name of the workflow for projects created with the Single-page application blueprint.

2. The most recent run is the one started by your merged pull request commit to the main branch and will likely show a status of In progress. Choose a successfully completed run from the list to open the details of that run.

3. Choose Variables. Copy the value for AppURL. This is the URL for the deployed single page web application. Open a new browser tab and paste in the value to view the built and deployed code. Leave the tab open.

4. Return to the list of workflow runs and wait for the most recent run to complete. When it does, return to the tab you opened to view the web application and refresh your browser. You should see the changes that you made in your merged pull request.
Cleaning up resources

Once you’ve explored working with a source repository and pull request, you might want to remove any resources you don’t need. You cannot delete pull requests, but you can close them. You can delete any branches you created.

If you no longer need the source repository or the project, you can also delete those resources. For more information, see Deleting a source repository (p. 93) and Deleting a project in Amazon CodeCatalyst (p. 62).

Working with source repositories in CodeCatalyst

A source repository is where you securely store code and files for your project. It also stores your source history, from the first commit through the latest changes. If you choose a blueprint that includes a source repository, that repository also contains the configuration files and other information for workflows and notifications for the project. This configuration information is stored in a folder named .codecatalyst.

You can create a source repository in CodeCatalyst either by creating a project with a blueprint that creates a source repository as part of creating a project, or by creating a source repository in an existing project. Project users will automatically see and be able to use the repositories you create for a project. You can also choose to link a Git repository hosted on GitHub to your project. When you do so, your project users can view and access that linked repository in the list of repositories for the project.

Note
Before you can link the repository, you must install the extension for the service that hosts it. You cannot link an empty repository, you can’t use it in CodeCatalyst until you have initialized it with an initial commit that creates a default branch.

For more information, see Installing an extension (p. 557).

By default, a source repository is shared with other members of your Amazon CodeCatalyst project. You can create additional source repositories for a project or link repositories to the project. All members of a project can view, add, edit, and delete files and folders in the project’s source repositories.

To quickly work on code in a source repository, you can create a Dev Environment that clones a specified repository and branch into it where you can work on the code in the integrated development environment (IDE) you choose for the Dev Environment. You can clone a source repository on your local computer and pull and push changes between your local repo and the remote repository in CodeCatalyst. You can also work with source repositories by configuring access to them in your preferred IDE as long as that IDE supports credential management.

Repository names must be unique within a CodeCatalyst project.

Topics
- Creating a source repository (p. 89)
- Linking a source repository (p. 90)
- Viewing a source repository (p. 91)
- Editing the settings for a source repository (p. 91)
- Cloning a source repository (p. 92)
- Deleting a source repository (p. 93)

Creating a source repository

When you create a project using a blueprint in Amazon CodeCatalyst, CodeCatalyst creates a source repository for you. That source repository contains sample code in addition to configuration information.
for the workflows and other resources created for you. This is the recommended way to get started with repositories in CodeCatalyst. You can choose to create repositories for a project. Those repositories will contain a single file, a README.md file that you can edit or delete at any time. Depending on your choices when you create a source repository, repositories might also contain a .gitignore file.

To create a source repository

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Create repository.
5. In Repository name, provide a name for the repository. In this guide, we use codecatalyst-source-repository, but you can choose a different name. Repository names must be unique within a project. For more information about requirements for repository names, see Quotas for source repositories in CodeCatalyst (p. 110).
6. (Optional) In Description, add a description for the repository that will help other users in the project understand what the repository is used for.
7. (Optional) Add a .gitignore file for the type of code you plan to push.
8. Choose Create.

Note
CodeCatalyst adds a README.md file to your repository when you create it. CodeCatalyst also creates an initial commit for the repository in a default branch named main. You can edit or delete the README.md file, but you can't change or delete the default branch.

Linking a source repository

When linking a source repository to a project, you can include repositories that have a CodeCatalyst extension for the service that hosts the repository, if that extension is installed for your space. Only users with the Space administrator role can install extensions. Once the extension is installed, you can link to repositories configured for access by that extension. For more information, see Installing and uninstalling extensions in CodeCatalyst (p. 557) or follow Quickstart: Using GitHub repositories in CodeCatalyst (p. 559).

Note
You can link a repository to only one project in a space. You cannot link an archived repository. While you can link an empty repository, you can't use it in CodeCatalyst until you have initialized it with an initial commit that creates a default branch.

To link a source repository

1. Navigate to the project where you want to link a repository.
   
   Note
   Before you can link a repository, a user with the Space administrator role must first install the extension for the provider that hosts the repository. For more information, see Installing an extension (p. 557).
2. In the navigation pane, choose Code, and then choose Source repositories.
3. Choose Add repository, and then choose Link repository.
4. In Repository provider, choose the name of the provider. In GitHub account, choose the linked account you want to use for working in the repository, and then in Repository, choose the name of the repository.
   
   Note
   You can only link an active repository. You cannot link an archived repository.
5. Review the information, and then choose Link.
Note
You can only link a repository to one project in a space.

To unlink a repository from a project, you must have the Space administrator role. For more information, see Managing GitHub repositories in CodeCatalyst (p. 563).

Viewing a source repository

You can view the source repositories associated with a project in Amazon CodeCatalyst. For source repositories in CodeCatalyst, the overview page for a repository provides a quick overview of information and activity in that repository, including:

- The description of the repository, if any
- The number of branches in the repository
- The number of open pull requests for the repository
- The number of related workflows for the repository
- The files and folders in the default branch, or the branch that you choose
- The title, author, and date of the last commit to the displayed branch
- The contents of the README.md file rendered in Markdown, if any README.md file is included

This page also provides links to the commits, branches, and pull requests for the repository, as well as a quick way to open, view, and edit individual files.

Note
You cannot view this information about linked repositories in the CodeCatalyst console. To view information about linked repositories, choose the link in the list of repositories to open that repository in the service that hosts it.

To navigate to the source repositories for a project

1. Navigate to your project, and do one of the following:
   - On the summary page for your project, choose the repository you want from the list, and then choose View repository.
   - In the navigation pane, choose Code, and then choose Source repositories. In Source repositories, choose the name of the repository from the list. You can filter the list of repositories by typing part of the repository name in the filter bar.

2. On the home page for the repository, view the contents of the repository and information about the associated resources such as the number of pull requests and workflows. By default, the contents for the default branch are shown. You can change the view by choosing a different branch from the drop-down list.

Tip
You can also quickly navigate to your project's repositories by choosing See project code from the project summary page.

Editing the settings for a source repository

You can edit the description of a repository in CodeCatalyst. This can help project members understand what the repository is used for, and provide information about the best practices and processes used by the team.

Note
You can't edit the name of a source repository.
You can't edit the name, description, or other information for a linked repository in CodeCatalyst. To modify information about a linked repository, you must edit it in the provider that hosts the linked repository. For more information, see the documentation for the service that hosts the linked repository.

To edit the description of a repository

1. In the CodeCatalyst console, navigate to the project that contains the source repository that you want to edit.
2. On the summary page for your project, choose the repository you want from the list, and then choose View repository. Alternatively, in the navigation pane, choose Code, and then choose Source repositories. Choose the name of the repository from the list of source repositories for the project.
3. On the overview page for the repository, choose More, and then choose Edit repository.
4. Edit the description, and then choose Save.

Cloning a source repository

To work effectively with multiple files, branches, and commits in source repositories, clone the source repository to your local computer and use a Git client or an integrated development environment (IDE) to make changes. Commit and push your changes to the source repository in order to work with CodeCatalyst features such as issues and pull requests. You can also choose to create a Dev Environment to work on code. Creating a Dev Environment automatically clones the repository and branch you specify into the Dev Environment.

Note
You cannot clone linked repositories in the CodeCatalyst console or create Dev Environments for them. To clone a linked repository locally, choose the link in the list of repositories to open that repository in the service that hosts it, and then clone it. For more information, see the documentation for the service that hosts the linked repository.

To create a Dev Environment from a source repository

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose Code, and then choose Source repositories.
3. Choose the source repository where you want to work on code.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Do one of the following:
   - Choose Work in existing branch, and then choose a branch from the Existing branch drop-down menu.
   - Choose Work in new branch, enter a branch name into the Branch name field, and choose a branch off of which to create the new branch from the Create branch from drop-down menu.
7. Optionally add a name for the Dev Environment or edit its configuration.
8. Choose Create.

To clone a source repository

1. Navigate to your project.
2. On the summary page for your project, choose the repository you want from the list, and then choose View repository. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.
Deleting a source repository

If a source repository for an Amazon CodeCatalyst project is no longer needed, you can delete it. Deleting a source repository also deletes any project information stored in the repository. If any workflows depend on the source repository, those workflows will be deleted from the list of project workflows after the repository is deleted. Issues that reference the source repository will not be deleted or altered, but any links to the source repository added to issues will fail once the repository is deleted.

Important
Deleting a source repository cannot be undone. After you delete a source repository, you are no longer able to clone it, pull data from it, or push data to it. Deleting a source repository does not delete any local copies of that repository (local repos). To delete a local repo, use your local computer's directory and file management tools.

Note
You cannot delete a linked repository in the CodeCatalyst console. To delete a linked repository, choose the link in the list of repositories to open that repository in the service that hosts it, and then delete it. For more information, see the documentation for the service that hosts the linked repository. To remove a linked repository from a project, see Managing GitHub repositories in CodeCatalyst [p. 563].

To delete a source repository

1. Navigate to the project that contains the source repository you want to delete.
2. On the summary page for your project, choose the repository you want from the list, and then choose View repository. Alternatively, in the navigation pane, choose Code, and then choose Source repositories. Choose the name of the repository from the list of source repositories for the project.
Working with branches in Amazon CodeCatalyst

In Git, branches are pointers or references to a commit. In development, they’re a convenient way to organize your work. You can use branches to separate work on a new or different version of files without affecting work in other branches. You can use branches to develop new features, store a specific version of your project, and more.

Source repositories in Amazon CodeCatalyst have contents and a default branch regardless of how you create them. Linked repositories might not have a default branch or content, but are not usable by CodeCatalyst until you initialize them and create a default branch. When you create a project using a blueprint, CodeCatalyst creates a source repository for that project that includes a README.md file, sample code, workflow definitions, and other resources. When you create a source repository without using a blueprint, a README.md file is added for you as a first commit, and a default branch is created for you. This default branch is named `main`. This default branch is the one used as the base or default branch in local repositories (repos) when users clone the repository.

**Note**

You can't delete the default branch. The first branch created for a source repository is the default branch for that repository. Additionally, search only displays results from the default branch. You can't search for code in other branches.

Creating a repository in CodeCatalyst also creates a first commit, which creates a default branch with a README.md file included in it. The name of that default branch is `main`. This is the default branch name used in the examples in this guide.

**Topics**

- [Creating and deleting a branch](#) (p. 94)
- [View and change the default branch for a repository](#) (p. 95)
- [Git commands for branches](#) (p. 96)
- [Viewing branches and details](#) (p. 97)

Creating and deleting a branch

You can use the CodeCatalyst console to create and delete branches in a CodeCatalyst repository. The branches you create will be visible to other users the next time they pull changes from the repository. When you delete a branch, copies of that branch remain in the clones of the repository on local computers until users pull and synchronize those changes.

**Tip**

You can also create branches as part of creating a Dev Environment to work on your code. For more information, see [Creating a Dev Environment](#) (p. 113).

You can also use Git to create and delete branches. For more information, see [Common Git commands for branches](#) (p. 96).

**To create a branch (console)**

1. In the CodeCatalyst console, navigate to the project where your repository resides.
2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

Choose the repository where you want to create a branch.

3.

4. On the overview page of the repository, choose More, and then choose Create branch.

5. Enter a name for the branch.

6. Choose a branch to create the branch from, and then choose Create.

To delete a branch (console)

1. Navigate to the project where your repository resides.

2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

Choose the repository where you want to delete a branch.

3. On the overview page of the repository, choose the drop-down selector next to the branch name, and then choose View all.

4. Choose the branch that you want to delete and then choose Delete branch.

Note
You cannot delete the default branch for a repository.

5. A confirmation dialog box appears. It shows the repository, number of open pull requests, and number of workflows associated with the branch.

6. To confirm deletion of the branch, type delete into the text box, and then choose Delete.

View and change the default branch for a repository

You can specify which branch to use as the default branch in a source repository in Amazon CodeCatalyst. All source repositories in CodeCatalyst have contents and a default branch regardless of how you create them. If you use a blueprint to create a project, the default branch in the source repository created for that project is named main. The contents of the default branch are displayed automatically on the overview page for that repository.

The default branch is treated a little differently than all other branches in a source repository. It has a special label next to its name, Default. The default branch is the one used as the base or default branch in local repositories (repos) when users clone the repository to local computers with a Git client. It is also the default used when creating workflows for storing workflow YAML files, and for storing information for issues. When using search in CodeCatalyst, only the default branch of a repository is searched. Because the default branch is fundamental to so many aspects of projects, you cannot delete a branch if it is specified as the default branch. However, you can choose to use a different branch as the default branch.

Note
You must have the Project administrator role to change the default branch for source repositories in CodeCatalyst projects. This does not apply to linked repositories.

To view and change the default branch for a repository

1. Navigate to the project where your repository resides.

2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

Choose the repository where you want to view the settings, including the default branch.
3. On the overview page of the repository, choose More, and then choose Manage settings.
4. In Default branch, the name of the branch specified as the default branch is displayed along with a label called Default next to the name. This same label appears next to the branch name in the list of branches in Branches.
5. To change the default branch, choose Edit.
6. Choose the name of the branch you want to make the default branch from the drop-down list and then choose Save.

Git commands for branches

You can use Git to create, manage, and delete branches in the clone of the source repository you have on your computer (your local repo) or in your Dev Environments, and then commit and push your changes to your CodeCatalyst source repository (the remote repository). For example:

Common Git commands for branches

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>git branch</code></td>
<td>Lists all branches in the local repo with an asterisk (*) displayed next to your current branch.</td>
</tr>
<tr>
<td><code>git fetch</code></td>
<td>Pulls information about all existing branches in the remote repository to the local repo.</td>
</tr>
<tr>
<td><code>git branch -a</code></td>
<td>Lists all branches in the local repo and remote tracking branches in the local repo.</td>
</tr>
<tr>
<td><code>git branch -r</code></td>
<td>Lists only remote tracking branches in the local repo.</td>
</tr>
<tr>
<td><code>git branch branch-name</code></td>
<td>Creates a branch in the local repo using the specified branch name. This branch will not appear in the remote repository until you commit and push the change.</td>
</tr>
<tr>
<td><code>git checkout -b branch-name</code></td>
<td>Creates a branch in the local repo using the specified branch name, and then switches to it.</td>
</tr>
<tr>
<td><code>git checkout other-branch-name</code></td>
<td>Switches to another branch in the local repo using the specified branch name.</td>
</tr>
<tr>
<td><code>git push -u remote-name branch-name</code></td>
<td>Pushes a branch from the local repo to the remote repository using the local repo's specified nickname for the remote repository and the specified branch name. Also sets up upstream tracking information for the branch in the local repo.</td>
</tr>
<tr>
<td><code>git merge from-other-branch-name</code></td>
<td>Merges changes from another branch in the local repo to the current branch in the local repo.</td>
</tr>
<tr>
<td><code>git branch -d branch-name</code></td>
<td>Deletes a branch in the local repo unless it contains work that has not been merged.</td>
</tr>
<tr>
<td><code>git push remote-name :branch-name</code></td>
<td>Deletes a branch in the remote repository using the specified nickname the local repo has for the remote repository and the specified branch name. (Note the use of the colon (:).) Alternatively, specify --delete as part of the command.</td>
</tr>
<tr>
<td><code>git push remote-name --delete branch-name</code></td>
<td>Deletes a branch in the remote repository using the specified nickname the local repo has for the remote repository and the specified branch name. (Note the use of the colon (:).) Alternatively, specify --delete as part of the command.</td>
</tr>
</tbody>
</table>
Viewing branches and details

You can view information about remote branches in Amazon CodeCatalyst, including specifics of the files, folders, and most recent commit for a specific branch, in the Amazon CodeCatalyst console. You can also use Git commands and your local operating system to view this information for remote and local branches.

To view branches (console)

1. In the CodeCatalyst console, navigate to the project that contains the source repository where you want to view branches. Choose Code, choose Source repositories, and then choose the source repository.
2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

Choose the repository where you want to view a branch.

3. The default branch of the repository is displayed. You can see a list of files and folders in the branch, information about the most recent commit, and the contents of the README.md file, if it exists in the branch. To view the information for a different branch, choose it from the drop-down list of branches for the repository.
4. To view all the branches for a repository, choose View all.

For more information about how to use Git and your operating system to view branches and details, see Common Git commands for branches (p. 96), your Git documentation, and your operating system documentation.
Creating or adding a file

file, such as a README.md file, in a branch in a repository. When working on more than one file, consider creating a Dev Environment (p. 113).

To create a Dev Environment from a source repository

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose Code, and then choose Source repositories.
3. Choose the source repository where you want to work on code.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Do one of the following:
   - Choose Work in existing branch, and then choose a branch from the Existing branch drop-down menu.
   - Choose Work in new branch, enter a branch name into the Branch name field, and choose a branch off of which to create the new branch from the Create branch from drop-down menu.
7. Optionally add a name for the Dev Environment or edit its configuration.
8. Choose Create.

To create a file in the CodeCatalyst console

1. Navigate to the project where you want to create a file. For more information about how to navigate to a repository, see Viewing a source repository (p. 91).
2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.
   - Choose the repository where you want to create the file.
3. (Optional) Choose the branch where you want to create the file, if you want to create the file in a different branch than the default branch.
5. Enter the name of the file in File name. Add the contents of the file in the editor.
   - Tip
     If you want to create the file in a sub-folder or subdirectory of the root of the branch, include that structure as part of the file name.
   - When you are satisfied with your changes, choose Commit.
6. In File name, review the name of the file and make any changes you might want to it. Optionally choose the branch where you want to create the file from the list of available branches in Branch. In Commit message, optionally enter a brief but informative description of why you made this change. This will be displayed as the basic commit information for the commit that adds the file to the source repository.
7. Choose Commit to commit and push the file to the source repository.

You can also add files to a source repository by cloning it to your local computer and using a Git client or connected integrated development environment (IDE) to push your files and changes.

Note

If you want to add a Git submodule, you must use a Git client or a Dev Environment and run the git submodule add command. You cannot add or view Git submodules in the CodeCatalyst console or view the differences in Git submodules in pull requests. For more information about Git submodules, see the Git documentation.
To add a file using a Git client or connected integrated development environment (IDE)

1. Clone your source repository to your local computer. For more information, see Cloning a source repository (p. 92).
2. Create files in your local repo or copy files into your local repo.
3. Create and push a commit by doing one of the following:
   • If you are using a Git client, at the terminal or command line, run the `git add` command, specifying the names of the files you want to add. Alternatively, to add all added or changed files, run the `git add` command followed by either a single or double period to indicate whether you want to include all the changes at the current directory level (single period) or all changes in the current directory and all subdirectories (double period). To commit the changes, run the `git commit -m` command and provide a commit message. To push your changes to the source repository in CodeCatalyst, run `git push`. For more information about Git commands, see your Git documentation and Git commands for branches (p. 96).
   • If you are using a Dev Environment or an IDE, create files and add files in the IDE, and then commit and push your changes. For more information, see Dev Environments in CodeCatalyst (p. 113) or consult your IDE documentation.

Viewing a file

You can view the files in your source repository in the Amazon CodeCatalyst console. You can view the files in the default branch and in any other branches. File contents might vary depending on the branch you choose to view.

To view files in the CodeCatalyst console

1. Navigate to the project where you want to view files. For more information, see Viewing a source repository (p. 91).
2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.
   
   Choose the repository where you want to view files.
3. A list of files and folders are displayed for the default branch. Files are indicated by a paper icon, while folders are indicated by a folder icon.
4. Do any of the following:
   • To view files and folders in a different branch, choose it from the list of branches.
   • To expand a folder, choose it from the list.
5. To view the contents of a specific file, choose it from the list. The contents of the file will be displayed in the branch. To view the contents of the file in a different branch, choose the branch you want from the branch selector.

   Tip
   When viewing the contents of a file, you can choose additional files to view from View files.
   To edit a file, choose Edit.

You can view multiple files in the console. You can also view files that you have cloned to your local computer using a Git client or an integrated development environment (IDE). For more information, see the documentation for your Git client or IDE.

Note
You cannot view Git submodules in the CodeCatalyst console. For more information about Git submodules, see the Git documentation.
Editing a file

You can edit individual files in the Amazon CodeCatalyst console. To edit multiple files at once, either create a Dev Environment or clone the repository and make your changes using a Git client or integrated development environment (IDE). For more information, see Dev Environments in CodeCatalyst (p. 113) or Cloning a source repository (p. 92).

To edit a file in the CodeCatalyst console

1. Navigate to the project where you want to edit a file. For more information about how to navigate to a repository, see Viewing a source repository (p. 91).
2. Choose the repository where you want to edit the file. Choose View branches and then choose the branch you want to work in. Choose the file from the list of files and folders in that branch.

   The contents of the file are displayed.
3. Choose Edit.
4. In the editor, edit the contents of the file and then choose Commit. Optionally, in Commit changes, add more information about the change in Commit message. When you are satisfied with your changes, choose Commit.

Renaming or deleting a file

You can rename or delete files in a Dev Environment, locally on your computer, or in an integrated development environment (IDE). Once you have renamed or deleted the files, commit and push those changes to the source repository. You can't rename or delete files in the Amazon CodeCatalyst console.

Working with pull requests in Amazon CodeCatalyst

A pull request is the primary way you and other project members can review, comment on, and merge code changes from one branch to another. You can use pull requests to review code changes collaboratively for minor changes or fixes, major feature additions, or new versions of your released software. If you use issues to track work on your project, you can link specific issues to pull requests to help you track what issues are being addressed by the code changes in the pull request. When you create, update, comment on, merge, or close a pull request, an email is automatically sent to the author of the pull request as well as any required or optional reviewers for the pull request.

Tip
You can configure what pull request events you that will receive emails about as part of your profile. For more information, see Managing notifications in Amazon CodeCatalyst (p. 71).

Pull requests require two branches in a source repository: a source branch that contains the code that you want reviewed, and a destination branch, where you want to merge the reviewed code. The source branch contains the AFTER commit, which is the commit that contains the changes you want to merge into the destination branch. The destination branch contains the BEFORE commit, which represents the state of the code before the pull request branch is merged into the destination branch.

Note
While you are creating a pull request, the difference displayed is the difference between the tip of the source branch and the tip of the destination branch. Once you create the pull request, the displayed difference will be between the revision of the pull request you choose and the commit that was the tip of the destination branch when you created the pull request. For
more information about differences and merge bases in Git, see git-merge-base in the Git documentation.

While a pull request is created for a specific source repository and branches, you can create, view, review, and close them as part of working with your project. You do not have to view the source repository in order to view and work with pull requests. A pull request state is set to Open when you create it. The pull request remains open until you either merge it in the CodeCatalyst console, which changes the state to Merged, or close it, which changes the state to Closed.

When your code has been reviewed, you can change the pull request state in one of several ways:

- Merge the pull request in the CodeCatalyst console. The code in the source branch of the pull request will be merged into the destination branch. The pull request status will change to Merged. It can't be changed back to Open.
- Merge the branches locally and push your changes, and then close the pull request in the CodeCatalyst console.
- Use the CodeCatalyst console to close the pull request without merging. This will change the status to Closed, and it will not merge the code from the source branch into the destination branch.

Before you create a pull request:

- Commit and push the code changes you want reviewed to a branch (the source branch).
- Set up notifications for your project, so other users can be notified about any workflows that run when you create a pull request. (This step is optional but recommended.)

**Topics**

- Creating a pull request (p. 101)
- Viewing pull requests (p. 103)
- Reviewing a pull request (p. 104)
- Updating a pull request (p. 105)
- Merging a pull request (p. 106)
- Closing a pull request (p. 108)

**Creating a pull request**

Creating pull requests helps other users see and review your code changes before you merge them into another branch. First, you create a branch for your code changes. This is referred to as the source branch for a pull request. After you commit and push changes to the repository, you can create a pull request that compares the contents of the source branch to the contents of the destination branch.

You can create a pull request in the Amazon CodeCatalyst console from a specific branch, from the pull requests page, or from the project overview. Creating a pull request from a specific branch automatically provides the repository name and source branch on the pull request creation page. When you create a pull request, you will automatically receive emails about any updates to the pull request, as well as when the pull request is merged or closed.

**Note**

While you are creating a pull request, the difference displayed is the difference between the tip of the source branch and the tip of the destination branch. Once the pull request has been created, the displayed difference will be between the revision of the pull request you choose and the commit that was the tip of the destination branch when you created the pull request. For more information about differences and merge bases in Git, see git-merge-base in the Git documentation.
To create a pull request

1. Navigate to your project.
2. Do one of the following:
   - In the navigation pane, choose Code, choose Pull requests, and then choose Create pull request.
   - On the repository home page, choose More, and then choose Create pull request.
   - On the project page, choose Create pull request.
3. In Source repository, make sure that the specified source repository is the one that contains the committed code. This option only appears if you did not create the pull request from the repository's main page.
4. In Destination branch, choose the branch to merge the code into after it is reviewed.
5. In Source branch, choose the branch that contains the committed code.
6. In Pull request title, enter a title that helps other users understand what needs to be reviewed and why.
7. (Optional) In Pull request description, provide information such as a link to issues or a description of your changes.
8. (Optional) In Issues, choose Link issues, and then either choose an issue from the list or enter its ID. To unlink an issue, choose the unlink icon.
9. (Optional) In Required reviewers, choose Add required reviewers. Choose from the list of project members to add them. Required reviewers must approve the changes before the pull request can be merged into the destination branch.
   
   **Note**
   You cannot add a reviewer as both a required reviewer and an optional reviewer. You cannot add yourself as a reviewer.
10. (Optional) In Optional reviewers, choose Add optional reviewers. Choose from the list of project members to add them. Optional reviewers do not have to approve the changes as a requirement before the pull request can be merged into the destination branch.
11. Review the differences between the branches. The difference displayed in a pull request is the changes between the revision in the source branch and the merge base, which is the head commit of the destination branch at the time the pull request was created. If no changes display, the branches might be identical, or you might have chosen the same branch for both the source and the destination.
12. When you are satisfied that the pull request contains the code and changes you want reviewed, choose Create.
   
   **Note**
   After you create the pull request, you can add comments. Comments can be added to the pull request or to individual lines in files as well as to the overall pull request. You can add links to resources, such as files, by using the @ sign followed by the name of the file.

To create a pull request from a branch

1. Navigate to the project where you want to create a pull request.
2. In the navigation pane, choose Source repositories, and then choose the repository that contains the branch where you have code changes to review.
3. Choose the drop-down arrow next to the default branch name, and then choose the branch you want from the list. To view all the branches for a repository, choose View all.
4. Choose More, and then choose Create pull request.
5. The repository and the source branch are preselected for you. In Destination branch, choose the branch where you will merge the code once it has been reviewed. In Pull request title, enter a
title that will help other project users understand what must be reviewed and why. Optionally, provide more information in **Pull request description**, such as pasting in a link to related issues in CodeCatalyst, or adding a description of the changes you made.

**Note**
Workflows that are configured to run for pull request create events will run after the pull request is created, if the destination branch for the pull request matches one of the branches specified in the workflow.

6. Review the differences between the branches. If no changes are displayed, the branches might be identical, or you might have chosen the same branch for both the source and the destination.

7. (Optional) In **Issues**, choose **Link issues**, and then either choose an issue from the list or enter its ID. To unlink an issue, choose the unlink icon.

8. (Optional) In **Required reviewers**, choose **Add required reviewers**. Choose from the list of project members to add them. Required reviewers must approve the changes before the pull request can be merged into the destination branch.

**Note**
You can't add a reviewer as both required and optional. You can't add yourself as a reviewer.

9. (Optional) In **Optional reviewers**, choose **Add optional reviewers**. Choose from the list of project members to add them. Optional reviewers do not have to approve the changes before the pull request can be merged into the destination branch.

10. When you are satisfied that the pull request contains the changes that you want reviewed and includes the required reviewers, choose **Create**.

If you have any workflows configured to run where the branch matches the destination branch in the pull request, you will see information about those workflow runs in **Overview** in the **Pull request details** area after the pull request is created. For more information, see Adding a push, pull, or schedule trigger (p. 329).

### Viewing pull requests

You can view pull requests for a project in the Amazon CodeCatalyst console. The project summary page displays all open pull requests for a project. To view all pull requests regardless of state, navigate to the pull requests page for your project.

#### To view open pull requests

1. Navigate to the project where you want to view pull requests.
2. On the project page, open pull requests are displayed, including information about who created the pull request, what repository contains the branches for the pull request, and the date the pull request was created. You can filter the open pull request view by source repository.
3. To view all pull requests, choose **View all**. You can use the selectors to choose between the options. For example, to view all pull requests, choose **Any status** and **Any author**.

   Alternatively, in the navigation pane, choose **Code**, and then choose **Pull requests**, and then use the selectors to refine your view.

4. On the **Pull requests** page, you can sort pull requests by ID, title, status, and more. To customize what information and how much information is displayed on the pull requests page, choose the gear icon.

**Note**
Not all changes in a pull request can be displayed in the console. For example, you cannot view Git submodules in the console, so you cannot view differences in a submodule in a pull request. Some differences might be too large to display. For more information, see Quotas for source repositories in CodeCatalyst (p. 110) and Viewing a file (p. 99).
Reviewing a pull request

You can use the Amazon CodeCatalyst console to collaboratively review and comment on the changes included in a pull request. You can add comments to individual lines of code in the difference between the source and destination branches, or in the difference between revisions of the pull request. You can also choose to create a Dev Environment to work on code.

**Tip**
You can configure what pull request events you that will receive emails about as part of your profile. For more information, see Managing notifications in Amazon CodeCatalyst (p. 71).

Pull requests show what the difference will be between the revision of the pull request and the commit that was the tip of the destination branch when you created the pull request. This is called the merge base. For more information about differences and merge bases in Git, see git-merge-base in the Git documentation.

**Tip**
When working in the console, particularly if you have had a pull request open for a while, consider refreshing your browser to ensure you have the latest revision available for a pull request before you start to review it.

**To review a pull request in the CodeCatalyst console**

1. Navigate to your project.
2. Navigate to the pull requests by doing one of the following:
   - If the pull request is listed on the project page, choose it from the list.
   - If the pull request is not listed on the project page, choose View all. Use the filters and sort to find the pull request, and then choose it from the list.
   - In the navigation pane, choose Code, and then choose Pull requests.
3. Choose the pull request you want to review from the list. You can filter the list of pull requests by typing part of its name in the filter bar.
4. In Overview, you can review the name and title of the pull request. You can create and view comments left on the pull request itself. You can also view the details of the pull request, including information about workflow runs, linked issues, reviewers, the author of the pull request, and feasible merge strategies.

**Note**
Comments left on specific lines of code appear in Changes.
5. (Optional) To add a comment that applies to the entire pull request, expand **Comments on pull request**, and then choose **Create comment**.

6. In **Changes**, you can see the differences between the destination branch and the most recent revision of the pull request. If there is more than one revision, you can change what revisions are compared in the difference between them. For more information about revisions, see **Revisions (p. 82)**.

7. To change the way differences are displayed, choose between **Unified** and **Split**.

8. To add a comment to a line in the pull request, go to the line you want to comment on. Choose the comment icon that appears for that line, enter a comment, and then choose **Save**.

9. To view changes between revisions in a pull request, or between its source and destination branches, choose from the available options in **Comparing**. Comments on lines in revisions are preserved in those revisions.

10. If you've configured your workflow to generate a code coverage report on pull request triggers, you can view the line and branch coverage findings in the relevant pull request. To hide code coverage findings, choose **Hide code coverage**. For more information, see **Code coverage reports (p. 147)**.

11. If you want to make code changes to the pull request, you can create a Dev Environment from the pull request. Choose **Create Dev Environment**. Optionally add a name for the Dev Environment or edit its configuration and then choose **Create**.

12. In **Reports**, you can view the quality reports in this pull request. If there is more than one revision, you can change what revisions are compared in the difference between them. You can filter the reports by name, status, workflow, action, and type.

   **Note**
   A workflow must be configured to generate reports in order for them to show up in your pull requests. For more information, see **Configuring reports (p. 149)**.

13. To view a specific report, choose it from the list. For more information, see **Testing using workflows in CodeCatalyst (p. 146)**.

14. If you are listed as a reviewer of this pull request and want to approve the changes, make sure that you are viewing the most recent revision, and then choose **Approve**.

   **Note**
   All required reviewers must approve a pull request before it can be merged.

### Updating a pull request

You can make it easier for other project members to review code by updating the pull request. You can update a pull request to change its reviewers, its links to issues, the title of the pull request, or its description. For example, you might want to change the required reviewers for a pull request to remove someone who's away on vacation, and add someone else. You can also update a pull request with further code changes by pushing commits to the source branch of an open pull request. Each push to the source branch of a pull request in the CodeCatalyst source repository creates a revision. Project members can view the differences between revisions in a pull request.

#### To update the reviewers for a pull request

1. Navigate to the project where you want to update the reviewers of a pull request.

2. On the project page, under **Open pull requests**, choose the pull request where you want to update reviewers. Alternatively, in the navigation pane, choose **Code**, choose **Pull requests**, and then choose the pull request you want to update.

3. (Optional) In **Overview**, in the **Pull request details** area, choose the plus sign to add required or optional reviewers. Choose the **X** next to a reviewer to remove them as an optional or required reviewer.
4. (Optional) In **Overview**, in the **Pull request details** area, choose **Link issues** to link an issue to the pull request, and then either choose an issue from the list or enter its ID. To unlink an issue, choose the unlink icon next to the issue you want to unlink.

**To update files and code in the source branch of a pull request**

1. To update multiple files, either [create a Dev Environment](p. 113), or clone the repository and its source branch and use a Git client or an integrated development environment (IDE) to make changes to the files in the source branch. Commit and push the changes to the source branch in the CodeCatalyst source repository to automatically update the pull request with the changes. For more information, see [Cloning a source repository](p. 92) and [Working with commits in Amazon CodeCatalyst](p. 108).

2. To update an individual file in a source branch, you can use a Git client or IDE as you would for multiple files. You can also edit it directly in the CodeCatalyst console. For more information, see [Editing a file](p. 100).

**To update the title and description of a pull request**

1. Navigate to the project where you want to update the title or description of a pull request.

2. The project page displays open pull requests, including information about who created the pull request, what repository contains the branches for the pull request, and when the pull request was created. You can filter the open pull request view by source repository. Choose the pull request that you want to change from the list.

3. To view all pull requests, choose **View all**. Alternatively, in the navigation pane, choose **Code**, and then choose **Pull requests**. Use the filter box or sort functions to find the pull request you want to change, and then choose it.

4. Choose **More**, and then choose **Edit pull request**.

5. Change the title or description, and then choose **Save**.

**Merging a pull request**

After your code has been reviewed and all required reviewers have approved it, you can merge a pull request in the CodeCatalyst console using a supported merge strategy, such as fast-forward. Not all merge strategies supported in the CodeCatalyst console are available as choices for all pull requests. CodeCatalyst evaluates the merge and only allows you to choose between merge strategies that are available in the console and capable of merging the source branch into the destination branch. You can also merge a pull request with your choice of Git merge strategies by running the `git merge` command on your local computer or a Dev Environment to merge the source branch into the destination branch. You can then push those changes in the destination branch to the source repository in CodeCatalyst.

**Note**

Merging the branch and pushing the changes in Git does not automatically close the pull request.

**Topics**

- [Merging a pull request (console)](p. 106)
- [Merging a pull request (Git)](p. 107)

**Merging a pull request (console)**

You can merge a pull request in the CodeCatalyst console if there are no merge conflicts between the source and destination branches and if all required reviewers have approved the pull request. If there are conflicts, or if the merge can’t be completed, the merge button is inactive, and a **Not mergeable** label is
displayed. In that case, you must obtain approval from any required approvers, resolve conflicts locally if necessary, and push those changes before you can merge. Merging a pull request will automatically send an email to the creator of the pull request as well as any required or optional reviewers. It will not automatically close or change the status of any issues linked to the pull request.

**Tip**
You can configure what pull request events you that will receive emails about as part of your profile. For more information, see [Managing notifications in Amazon CodeCatalyst](p. 71).

**To merge a pull request**

1. Navigate to the project where you want to merge a pull request.
2. On the project page, under **Open pull requests**, choose the pull request you want to merge. If you do not see the pull request, choose **View all pull requests** and then choose it from the list. Alternatively, in the navigation pane, choose **Code**, choose **Pull requests**, and then choose the pull request you want to merge. Choose **Merge**.
3. Choose from the available merge strategies for the pull request. Optionally, select or deselect the option to delete the source branch after merging the pull request, and then choose **Merge**.

**Note**
If the **Merge** button is inactive, or you see the **Not mergeable** label, either required reviewers have not yet approved the pull request, or the pull request can't be merged in the CodeCatalyst console. A reviewer who has not approved a pull request is indicated by a clock icon in the **Pull request details** area in **Overview**. If all required reviewers have approved the pull request but the **Merge** button is still inactive, you might have a merge conflict. You can resolve merge conflicts for the destination branch in a Dev Environment or the CodeCatalyst console and then merge the pull request, or you can resolve conflicts and merge locally, and then push the commit that contains the merge to the source branch in CodeCatalyst. For more information, see [Merging a pull request (Git)](p. 107) and your Git documentation.

**Merging a pull request (Git)**

Git supports many options for merging and managing branches. The following commands are some of the options that you can use. For more information, see the available documentation on the [Git website](https://git-scm.com). Once you have merged and pushed your changes, manually close the pull request. For more information, see [Closing a pull request](p. 108).

**Common Git commands for merging branches**

<table>
<thead>
<tr>
<th>Command Description</th>
<th>Git Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merges changes from the source branch in the local repo to the destination branch in the local repo.</td>
<td><code>git checkout destination-branch-name</code></td>
</tr>
<tr>
<td></td>
<td><code>git merge source-branch-name</code></td>
</tr>
<tr>
<td>Merges the source branch into the destination branch, specifying a fast-forward merge. This merges the branches and moves the destination branch pointer to the tip of the source branch.</td>
<td><code>git checkout destination-branch-name</code></td>
</tr>
<tr>
<td></td>
<td><code>git merge --ff-only source-branch-name</code></td>
</tr>
<tr>
<td>Merges the source branch into the destination branch, specifying a squash merge. This combines all commits from the source branch into a single merge commit in the destination branch.</td>
<td><code>git checkout destination-branch-name</code></td>
</tr>
<tr>
<td></td>
<td><code>git merge --squash source-branch-name</code></td>
</tr>
<tr>
<td>Merges the source branch into the destination branch, specifying a three-way merge. This creates a merge commit and adds the individual commits from the source branch to the destination branch.</td>
<td><code>git checkout destination-branch-name</code></td>
</tr>
<tr>
<td></td>
<td><code>git merge --no-ff source-branch-name</code></td>
</tr>
</tbody>
</table>
Closing a pull request

You can mark a pull request as **Closed**. This does not merge the pull request, but it can help you determine which pull requests require action and which pull requests are no longer relevant. We recommend closing a pull request after you merge it. Closing a pull request will automatically send an email to the creator of the pull request as well as any required or optional reviewers. It will not automatically change the status of any issues linked to the pull request.

**Note**

You cannot re-open a pull request after it has been closed.

**To close a pull request**

1. Navigate to the project where you want to close a pull request.
2. On the project page, open pull requests are displayed. Choose the pull request that you want to close.
3. Choose More, and then choose Close pull request.
4. Review the information, and then choose Close pull request.

Working with commits in Amazon CodeCatalyst

Commits are snapshots of the contents and changes to the contents of your repository. Every time a user commits and pushes a change to a branch, that information is saved. Git commit information includes the commit author, the person who committed the change, the date and time, and the changes made. Similar information is automatically included when you create or edit a file in the Amazon CodeCatalyst console, but the author name is your CodeCatalyst user name. You can also add Git tags to commits to help you identify specific commits.

In Amazon CodeCatalyst, you can:

- View a list of commits for a branch.
- View individual commits, including the changes made in a commit when compared to its parent or parents.

You can also view files and folders. For more information, see [Working with files in Amazon CodeCatalyst](p. 97).

**Topics**

- Viewing commits to a branch (p. 109)
Viewing commits to a branch

You can view the history of changes made to a branch by reviewing the branch's commits in the CodeCatalyst console. This helps you understand who made changes to the branch and when. You can also review the changes made in a specific commit.

You can also view commits by using your Git client. For more information, see your Git documentation.

To view commits (console)

1. Navigate to the project that contains the source repository where you want to view commits.

2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

   Choose the repository where you want to view commits to a branch.

3. The default branch of the repository is displayed, including information about the most recent commit to the branch. Choose Commits. Alternatively, choose More, and then choose View commits.

4. To view commits for a different branch, choose the branch selector, and then choose the name of the branch.

5. To view details about a particular commit, choose its title from Commit title. Details of the commit are displayed, including information about its parent commit and the changes made to the code by comparing the parent commit to the specified commit.

   Tip
   If a commit has more than one parent, you can choose which parent commit to view information and display changes for by choosing the drop-down icon next to the parent commit ID.

Changing how commits are displayed (CodeCatalyst console)

You can change what information is displayed in the Commits view. You can choose to hide or display columns such as author and commit ID.

To change how commits are displayed (console)

1. Navigate to the project that contains the source repository where you want to view commits.

2. Choose the name of the repository from the list of source repositories for the project. Alternatively, in the navigation pane, choose Code, and then choose Source repositories.

   Choose the repository where you want to change how you view commits.

3. The default branch of the repository is displayed, including information about the most recent commit to the branch. Choose Commits.

4. Choose the gear icon.

5. In Preferences, choose the number of commits to display, and choose whether to display information about commit author, commit date, and the commit ID.

   Note
   You can't hide the commit title in the display of information.
6. When you have made your changes, choose **Save** to save them, or **Cancel** to discard them.

## Quotas for source repositories in CodeCatalyst

The following table describes quotas and limits for source repositories in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see [Quotas for CodeCatalyst](p. 620).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Information</th>
</tr>
</thead>
</table>
| Branch names              | Any combination of allowed characters between 1 and 256 characters in length and must be unique within a repository. Branch names cannot:  
  • begin or end with a slash (/) or period (.)  
  • consist of the single character @  
  • contain two or more consecutive periods (..), forward slashes (//), or the following character combination: @{  
  • contain spaces or any of the following characters: ? ^ * \ [ ~ :  
  Branch names are references. Many of the limitations on branch names are based on the Git reference standard. For more information, see [Git Internals](http://git-scm.com/) and [git-check-ref-format](http://git-scm.com/). |
| Comments on a pull request| Maximum of 1,000 on a pull request.                                                                                                                                                                            |
| Commit message            | Maximum of 1024 characters.                                                                                                                                                                                 |
| File paths                | Any combination of allowed characters between 1 and 4,096 characters in length. File paths must be an unambiguous name that specifies the file and the exact location of the file. File paths cannot exceed 20 directories in depth. In addition, file paths cannot:  
  • contain empty strings  
  • be a relative file path  
  • include any of the following character combinations:  
    /../  
    /.../
    //  
  • end with a trailing slash or backslash  
  File names and paths must be fully qualified. The name and path to a file on your local computer must follow the standards for that operating |
<table>
<thead>
<tr>
<th>Resource</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size</td>
<td>Maximum of 6 MB for any individual file when using the CodeCatalyst console.</td>
</tr>
<tr>
<td>File size viewable in the CodeCatalyst console</td>
<td>Maximum of 6 MB for any individual file when using the CodeCatalyst console.</td>
</tr>
<tr>
<td>Git blob size</td>
<td>Maximum of 2 GB.</td>
</tr>
<tr>
<td>Note</td>
<td>There is no limit on the number or the total size of all files in a single commit, as long as the metadata does not exceed 6 MB and a single blob does not exceed 2 GB. However, as a best practice, consider making multiple smaller commits rather than one large commit.</td>
</tr>
<tr>
<td>Metadata for a commit</td>
<td>Maximum of 6 MB for the combined metadata for a commit (for example, the combination of author information, date, parent commit list, and commit messages).</td>
</tr>
<tr>
<td>Note</td>
<td>There is no limit on the number or the total size of all files in a single commit, as long as the data does not exceed 20 MB, an individual file does not exceed 6 MB, and a single blob does not exceed 2 GB.</td>
</tr>
<tr>
<td>Number of CodeCatalyst issues that can be linked to a pull request</td>
<td>50</td>
</tr>
<tr>
<td>Number of Jira issues that can be linked to a pull request</td>
<td>50</td>
</tr>
<tr>
<td>Number of open pull requests in a space</td>
<td>Maximum of 1,000 for an Amazon CodeCatalyst space.</td>
</tr>
<tr>
<td>Number of references in a single push</td>
<td>Maximum of 4,000, including create, delete, and update. There is no limit on the overall number of references in the repository.</td>
</tr>
<tr>
<td>Number of repositories in a space</td>
<td>Maximum of 5,000 for an Amazon CodeCatalyst space.</td>
</tr>
<tr>
<td>Repository descriptions</td>
<td>Any combination of characters between 0 and 1,000 characters in length. Repository descriptions are optional.</td>
</tr>
<tr>
<td>Resource</td>
<td>Information</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Repository names</td>
<td>Repository names must be unique within a project. They can contain any combination of letters, numbers, periods, underscores, and dashes between 1 and 100 characters in length. Names are case sensitive. Repository names cannot end in .git, cannot contain spaces, and cannot contain any of the following characters: ! ? @ $ % ^ &amp; * ( ) + = { } [ ]</td>
</tr>
<tr>
<td>Repository size</td>
<td>Maximum of 5 GB. Repositories are also impacted by the storage limits for your space. For more information, see Troubleshooting problems with source repositories (p. 585).</td>
</tr>
<tr>
<td>Reviewers for a pull request</td>
<td>Maximum of 100 reviewers total (optional or required) for a pull request.</td>
</tr>
</tbody>
</table>
Dev Environments in CodeCatalyst

Dev Environments are cloud-based development environments. In Amazon CodeCatalyst, you use Dev Environments to work on the code stored in the source repositories of your project. When creating a Dev Environment, you have several options:

- Create a project-specific Dev Environment in CodeCatalyst to work on code with a supported integrated development environment (IDE).
- Create an empty Dev Environment, clone code into it from a source repository, and work on that code with a supported IDE.
- Create a Dev Environment in an IDE and clone a source repository into the Dev Environment

A devfile is an open standard YAML file that standardizes your Dev Environments. In other words, this file codifies the required development tools for your Dev Environment. As a result, you can quickly set up a Dev Environment, switch between projects, and replicate the Dev Environment configuration across team members. Dev Environments minimize the time that you spend creating and maintaining a local development environment, because they use a devfile that configures all of the tools you need to code, test, and debug for a given project.

The project tools and application libraries included in your Dev Environment are defined by the devfile in the source repository of your project. If you don't have a devfile in your source repository, CodeCatalyst automatically applies a default devfile. This default devfile includes tools for the most frequently used programming languages and frameworks. If your project was created using a blueprint, a devfile is automatically created by CodeCatalyst. For more information about the devfile, see https://devfile.io.

After you've created a Dev Environment, only you can access it. In your Dev Environment, you can view and work on your source repository's code in a supported IDE.

By default, a Dev Environment is configured to have a 2-core processor, 4 GB of RAM, and 16 GB of persistent storage. If you have Space administrator permissions, you can change the billing tier for your space to use different Dev Environment configuration options and manage compute and storage limits. Here is one possible workflow for a Dev Environment:

Akua Mansa is a new developer at Example Corp. Akua joined her new team just before a new version release of the team's product. Akua's team needs her to work on a new feature for the upcoming version of the product promptly. To avoid an extensive setup process, Akua accepts an invitation to join her team's CodeCatalyst project. She then assigns herself the related issue and creates a Dev Environment from an existing branch of her team's source repository. Akua's Dev Environment opens the source repository's code in her chosen IDE. The open IDE instance is connected to her Dev Environment which contains a devfile that has been automatically identified and applied. The devfile specifies all of the tools that she needs to get started. Akua writes the code for the new product feature, commits her code changes, pushes her changes to the existing branch, and then deletes her Dev Environment when her work is complete. Akua has contributed code to her new team's source repository without a lengthy setup process.

Creating a Dev Environment

You can create a Dev Environment in multiple ways:
• Create a Dev Environment in CodeCatalyst with a CodeCatalyst source repository or a linked source repository (p. 90) from the Overview, Dev Environments or Source repositories pages
• Create an empty Dev Environment in CodeCatalyst that is not connected to a source repository from the Dev Environments page
• Create a Dev Environment in your IDE of choice and clone any source repository into the Dev Environment

You can create one Dev Environment per branch of a repository. A project can have multiple repositories. The Dev Environments you create can only be managed with your CodeCatalyst account, but you can open the Dev Environment and work in it with any of the supported IDEs. You must have the AWS Toolkit installed to use Dev Environments in your IDE. For more information, see Supported integrated development environments for Dev Environments (p. 114). By default, Dev Environments are created with a 2-core processor, 4 GB of RAM, and 16 GB of persistent storage.

Note
If you created a Dev Environment that is associated with a source repository, the Resource column always shows the branch you specified when creating this Dev Environment. This applies even if you create another branch, switch to another branch within the Dev Environment, or clone an additional repository. If you created an empty Dev Environment, the Resource column will be blank.

Supported integrated development environments for Dev Environments

You can use Dev Environments with the following supported integrated development environments (IDEs):

• AWS Cloud9
• JetBrains IDEs
  • IntelliJ IDEA Ultimate
  • GoLand
  • PyCharm Professional
• Visual Studio Code

Creating a Dev Environment in CodeCatalyst

To create a Dev Environment from a branch

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to create a Dev Environment.
3. In the navigation pane, do one of the following:
   • Choose Overview, and then navigate to the My Dev Environments section.
   • Choose Code, and then choose Dev Environments.
   • Choose Code, choose Source repositories, and choose the repository for which you want to create a Dev Environment.
5. Choose a supported IDE from the drop-down menu. See Supported integrated development environments for Dev Environments (p. 114) for more information.
6. Choose Clone a repository.
7. Do one of the following:
a. Choose the repository to clone, choose **Work in existing branch**, and then choose a branch from the **Existing branch** drop-down menu.

**Note**
If you choose a third-party repository, you must work in an existing branch.

b. Choose the repository to clone, choose **Work in new branch**, enter a branch name into the **Branch name** field, and choose a branch off of which to create the new branch from the **Create branch from** drop-down menu.

**Note**
If you create a Dev Environment from the **Source repositories** page or from a specific source repository, you do not need to choose a repository. The Dev Environment will be created from the source repository you chose from the **Source repositories** page.

9. Optionally, choose the **Dev Environment configuration** edit button to edit the Dev Environment's compute, storage, or timeout configuration.
10. Choose **Create**. While your Dev Environment is being created, the Dev Environment status column will display **Starting**, and the status column will display **Running** once the Dev Environment has been created.

**To create an empty Dev Environment**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the project where you want to create a Dev Environment.
3. In the navigation pane, do one of the following:
   - Choose **Overview**, and then navigate to the **My Dev Environments** section.
   - Choose **Code**, and then choose **Dev Environments**.
4. Choose **Create Dev Environment**.
5. Choose a supported IDE from the drop-down menu. See [Supported integrated development environments for Dev Environments](https://docs.aws.amazon.com/codecatalyst/latest/userguide/user-guide-integrations.html) for more information.
6. Choose **Create an empty Dev Environment**.
8. Optionally, choose the **Dev Environment configuration** edit button to edit the Dev Environment's compute, storage, or timeout configuration.
9. Choose **Create**. While your Dev Environment is being created, the Dev Environment status column will display **Starting**, and the status column will display **Running** once the Dev Environment has been created.

**Note**
Creating and opening a Dev Environment for the first time might take one to two minutes.

**Note**
After the Dev Environment opens in the IDE, you might need to change the directory to the source repository before you commit and push changes to your code.

---

**Stopping a Dev Environment**

The `/projects` directory of a Dev Environment stores the files that are pulled from the source repository and the devfile that is used to configure the Dev Environment. The `/home` directory, which is empty upon Dev Environment creation, stores the files you create while using your Dev Environment.
Resuming a Dev Environment

The /projects directory of a Dev Environment stores the files that are pulled from the source repository and the devfile that is used to configure the Dev Environment. The /home directory, which is empty upon Dev Environment creation, stores the files you create while using your Dev Environment.

Everything in the /projects and /home directories of a Dev Environment is stored persistently, so you can stop working in a Dev Environment if you need to switch to another Dev Environment, repository, or project.

A Dev Environment will automatically stop if it is idle for the amount of time that was selected in the Timeout fields during Dev Environment creation. You must close the AWS Cloud9 browser tab for the Dev Environment to be idle.

**Note**
The Dev Environment is still available and running even if you delete the branch with which you created the Dev Environment. If you want to resume working in a Dev Environment for which you deleted the branch, create a new branch and push your changes to it.

To resume a Dev Environment from the overview page

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to resume a Dev Environment, and navigate to the My Dev Environments section.
3. Choose Resume in (IDE).

   • For JetBrains IDEs, choose JetBrains Gateway-EAP when prompted to Choose an application to open the JetBrains-gateway link. Choose Open Link to confirm when prompted.
• For the VS Code IDE, choose VS Code when prompted to **Choose an application to open the VS Code link.** Choose **Open Link** to confirm.

**To resume a Dev Environment from the source repository**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the project where you want to resume a Dev Environment.
3. In the navigation pane, choose **Code**.
4. Choose **Source Repositories**.
5. Choose the source repository that contains the Dev Environment you want to resume.
6. Choose the branch name to view a drop-down menu of your branches, then choose your branch.
7. Choose **Resume Dev Environment**.
   - For JetBrains IDEs, choose **Open Link** to confirm when prompted to **Allow this site to open the JetBrains-gateway link with JetBrains Gateway?**
   - For the VS Code IDE, choose **Open Link** to confirm when prompted to **Allow this site to open the VS Code link with Visual Studio Code?**

**To resume a Dev Environment from the Dev Environments page**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the project where you want to resume a Dev Environment.
3. In the navigation pane, choose **Code**.
4. Choose **Dev Environments**.
5. From the **IDE** column, choose **Resume in (IDE)** for the Dev Environment.
   - For JetBrains IDEs, choose **Open Link** to confirm when prompted to **Allow this site to open the JetBrains-gateway link with JetBrains Gateway?**
   - For the VS Code IDE, choose **Open Link** to confirm when prompted to **Allow this site to open the VS Code link with Visual Studio Code?**

**Note**
Resuming a Dev Environment may take a few minutes.

**Editing a Dev Environment**

While your IDE is running, you can edit the Dev Environment. If you edit your compute or inactivity timeout, your Dev Environment will restart after you've saved your changes.

**To edit a Dev Environment**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the project where you want to edit a Dev Environment.
3. In the navigation pane, choose **Code**.
4. Choose **Dev Environments**.
5. Choose the Dev Environment you want to edit.
6. Choose **Edit**.
7. Make the changes you want to the compute or inactivity timeout.
8. Choose **Save**.
Deleting a Dev Environment

When you have finished working on the content that is stored in your Dev Environment, you can delete the Dev Environment. Create a new Dev Environment to work on new content. If you delete your Dev Environment, the persisted content will be permanently deleted. Before you delete your Dev Environment, make sure you commit and push your code changes to the Dev Environment’s original source repository. After you have deleted your Dev Environment, compute and storage billing for the Dev Environment will stop.

**Important**
Deleting a Dev Environment cannot be undone. After you delete a Dev Environment, you are no longer able to recover it.

**To delete a Dev Environment**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to delete a Dev Environment.
3. In the navigation pane, choose Code.
5. Choose the Dev Environment you want to delete.
6. Choose Delete.
7. Enter delete to confirm the Dev Environment deletion.
8. Choose Delete.

**Note**
After you delete your Dev Environment, it may take a few minutes for the storage quota to be updated. If you’ve reached the storage quota, you will be unable to create a new Dev Environment during this time.

Configuring your Dev Environment

A *devfile* is an open standard that helps you to customize your development Dev Environments across your team. A devfile is a YAML file that codifies your required development tools. By configuring a devfile, you can pre-determine the project tools and application libraries you need and Amazon CodeCatalyst installs them to your Dev Environment for you. The devfile is specific to the repository for which it was created, and you can create a separate devfile for each repository. Your Dev Environment supports commands and events, and provides a default universal devfile image.

If you create a project using the empty blueprint, you can create a devfile manually. If you create a project using a different blueprint, CodeCatalyst creates a devfile automatically. The */projects* directory of a Dev Environment stores the files that are pulled from the source repository and the devfile. The */home* directory, which is empty when you first create a Dev Environment, stores the files you create while using your Dev Environment. Everything in the */projects* and */home* directories of a Dev Environment is stored persistently.

**Note**
The */home* folder only changes if you change the name of the devfile or devfile component name. If you change the devfile or devfile component name, the contents of the */home* directory are replaced and your previous */home* directory data cannot be recovered.

If you create a Dev Environment with a source repository that does not contain a devfile in its root, or if you create a Dev Environment without a source repository, a default universal devfile is applied to the source repository automatically. The same default universal devfile image is used for all IDEs.
CodeCatalyst currently supports devfile version 2.0.0. For more information about the devfile, see Devfile schema - Version 2.0.0.

To change the configuration of a Dev Environment, you must edit the devfile. We recommend that you edit the devfile in a supported IDE and then update your Dev Environment, but you can also edit the devfile from the root of the source repository in CodeCatalyst. If you edit the devfile in a supported IDE, you must commit and push your changes to the source repository or create a pull request so a team member can review and approve the devfile edits.

- Editing the repository devfile for a Dev Environment in AWS Cloud9
- Editing the repository devfile for a Dev Environment in VS Code
- Editing the repository devfile for a Dev Environment in JetBrains

**Note**
You can only include public container images in your devfile.

**Topics**
- Editing a repository devfile for a Dev Environment (p. 119)
- Moving a repository devfile for a Dev Environment (p. 120)
- Recovery mode (p. 120)
- Devfile features supported by CodeCatalyst (p. 120)
- Example: Configuring the devfile for your Dev Environment (p. 120)
- Devfile commands (p. 121)
- Devfile events (p. 122)
- Devfile components (p. 122)
- Universal devfile images (p. 123)

## Editing a repository devfile for a Dev Environment

**To edit the repository devfile**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project that contains the source repository for which you want to edit the devfile.
3. In the navigation pane, choose Code.
5. Choose the source repository that contains the devfile that you want to edit.
6. From the list of files, choose the devfile.yaml file.
7. Choose Edit.
8. Edit the devfile.
9. Choose Commit, or create a pull request so a team member can review and approve the changes.

**Note**
If you edit your devfile, you have to restart it for the changes to take effect. This can be done by running /aws/mde/mde start --location devfile.yaml. If there's a problem starting your devfile, it will enter recovery mode.

You can review which devfile is being used by running /aws/mde/mde status. The location field has the path of the devfile relative to the environment's /projects folder.

```
Moving a repository devfile for a Dev Environment

You can move the default devfile in /projects/devfile.yaml to your source code repository. To update the location of the devfile, use following command: /aws/mde/mde start --location repository-name/devfile.yaml.

Recovery mode

If there's a problem starting your devfile, it will enter recovery mode so that you can still connect to your environment and fix your devfile. While in recovery mode, running /aws/mde/mde status won't contain the location of your devfile.

You can check the error in the logs under /aws/mde/logs, fix the devfile, and try running /aws/mde/mde start again.

Devfile features supported by CodeCatalyst

CodeCatalyst supports the following devfile features on version 2.0.0. For more information about the devfile, see Devfile schema - Version 2.0.0.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec</td>
<td>Command</td>
</tr>
<tr>
<td>postStart</td>
<td>Event</td>
</tr>
<tr>
<td>container</td>
<td>Component</td>
</tr>
<tr>
<td>args</td>
<td>Component Properties</td>
</tr>
<tr>
<td>env</td>
<td>Component Properties</td>
</tr>
<tr>
<td>mountSources</td>
<td>Component Properties</td>
</tr>
<tr>
<td>volumeMounts</td>
<td>Component Properties</td>
</tr>
</tbody>
</table>

Example: Configuring the devfile for your Dev Environment

The following is an example of a simple devfile.

```
schemaVersion: 2.0.0
metadata:
  name: al2
```
Devfile commands

Currently, CodeCatalyst only supports exec commands in your devfile. For more information, see Adding commands in the Devfile.io documentation.

The following example shows you how to specify exec commands in your devfile.

```
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
```

After you're connected to your Dev Environment, you can execute defined commands through the terminal.

```
/aws/mde/mde command <command-id>
/aws/mde/mde command executescript
```

For long running commands, you can use the streaming flag `-s` to output the execution of the command in real time.
Note
command-id must be lower case.

Exec parameters supported by CodeCatalyst

CodeCatalyst supports the following exec parameters on devfile version 2.0.0.

- commandLine
- component
- id
- workingDir

Devfile events

Currently, CodeCatalyst only supports postStart events in your devfile. For more information, see postStartObject in the Devfile.io documentation.

The following example shows you how to add postStart event bindings in your devfile.

```yaml
commands:
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"

events:
  postStart:
    - updateyum
    - executescript
```

After startup, your Dev Environment will execute the specified postStart commands in the order they are defined. If a command fails, the Dev Environment will continue running and the execution output is stored in the logs under /aws/mde/logs.

Devfile components

Currently, CodeCatalyst only supports container components in your devfile. For more information, see Adding components in the Devfile.io documentation.

The following example shows you how to add a startup command to your container in your devfile.

```yaml
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      command: ['sleep', 'infinity']
```
Note
When the container has short lived entry command, you must include command: ['sleep', 'infinity'] to keep the container running.

CodeCatalyst also supports the following properties in your container component: args, env, mountSources, and volumeMounts.

Universal devfile images

The default universal image includes the most commonly used programming languages and related tools that can be used for your IDE. If no image is specified, CodeCatalyst provides this image and contains tools that are maintained by CodeCatalyst. To remain notified of new image releases, see Universal image notifications with SNS (p. 123).

Note

Amazon CodeCatalyst supports the following devfile images.

<table>
<thead>
<tr>
<th>Image version</th>
<th>Image identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal image 1.0</td>
<td>public.ecr.aws/aws-mde/universal-image:1.0</td>
</tr>
<tr>
<td>Universal image 2.0</td>
<td>public.ecr.aws/aws-mde/universal-image:2.0</td>
</tr>
<tr>
<td>Universal image 3.0</td>
<td>public.ecr.aws/aws-mde/universal-image:3.0</td>
</tr>
</tbody>
</table>

Note
If you're using AWS Cloud9, auto-complete will not work for PHP, Ruby and CSS after upgrading to universal-image:3.0.

Topics
- Universal image notifications with SNS (p. 123)
- Universal image 1.0 runtime versions (p. 124)
- Universal image 2.0 runtime versions (p. 125)
- Universal image 3.0 runtime versions (p. 126)

Universal image notifications with SNS

CodeCatalyst provides a universal image notification service. You can use it to subscribe to an Amazon Simple Notification Service (SNS) topic that notifies you when CodeCatalyst universal image updates have been released. For more information about SNS topics, see What is Amazon Simple Notification Service?

Whenever new universal images are released, we send notifications to subscribers; this section describes how to subscribe to CodeCatalyst universal image updates.

Sample message

```json
{
    "Type": "Notification",
```

123
To subscribe to CodeCatalyst universal image updates using the Amazon SNS console

1. Open the Amazon SNS console to the Dashboard.
2. In the navigation bar, choose your AWS Region.
3. In the navigation pane, choose Subscriptions, and then choose Create subscription.
5. In Protocol, choose Email.
6. In Endpoint, provide an email address. This email address will be used to receive notifications.
7. Choose Create subscription.
8. You will receive a confirmation email with the subject line "AWS Notification - Subscription Confirmation". Open the email and choose Confirm subscription.

To unsubscribe from CodeCatalyst universal image updates using the Amazon SNS console

1. Open the Amazon SNS console to the Dashboard.
2. In the navigation bar, choose your AWS Region.
3. In the navigation pane, choose Subscriptions and then select the subscription you want to unsubscribe from.
4. Choose Actions, and then choose Delete subscriptions.
5. Choose Delete.

Universal image 1.0 runtime versions

The following table lists the available runtimes for universal-image:1.0.

**universal-image:1.0 runtime versions**

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version</th>
<th>Specific major and latest minor version</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws cli</td>
<td>2.11</td>
<td>aws-cli: 2.x</td>
</tr>
<tr>
<td>docker compose</td>
<td>2.16</td>
<td>docker-compose: 2.x</td>
</tr>
<tr>
<td>dotnet</td>
<td>6.0</td>
<td>dotnet: 6.x</td>
</tr>
<tr>
<td>Runtime name</td>
<td>Version</td>
<td>Specific major and latest minor version</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
<td>dotnet: 7.x</td>
</tr>
<tr>
<td>golang</td>
<td>1.19</td>
<td>golang: 1.x</td>
</tr>
<tr>
<td>java</td>
<td>corretto11</td>
<td>java: corretto11.x</td>
</tr>
<tr>
<td></td>
<td>corretto17</td>
<td>java: corretto17.x</td>
</tr>
<tr>
<td>nodejs</td>
<td>14.20</td>
<td>nodejs: 14.x</td>
</tr>
<tr>
<td></td>
<td>16.19</td>
<td>nodejs: 16.x</td>
</tr>
<tr>
<td>openssl</td>
<td>1.0</td>
<td>openssl: 1.x</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>7.2</td>
<td>php: 7.x</td>
</tr>
<tr>
<td>python</td>
<td>3.9</td>
<td>python: 3.x</td>
</tr>
<tr>
<td></td>
<td>3.10</td>
<td></td>
</tr>
<tr>
<td>ruby</td>
<td>3.1</td>
<td>ruby: 3.x</td>
</tr>
<tr>
<td>terraform</td>
<td>1.4</td>
<td>terraform: 1.x</td>
</tr>
</tbody>
</table>

**Universal image 2.0 runtime versions**

The following table lists the available runtimes for `universal-image:2.0`.

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version</th>
<th>Specific major and latest minor version</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws cli</td>
<td>2.11</td>
<td>aws-cli: 2.x</td>
</tr>
<tr>
<td>docker compose</td>
<td>2.17</td>
<td>docker-compose: 2.x</td>
</tr>
<tr>
<td>dotnet</td>
<td>6.0</td>
<td>dotnet: 6.x</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
<td>dotnet: 7.x</td>
</tr>
<tr>
<td>golang</td>
<td>1.20</td>
<td>golang: 1.x</td>
</tr>
<tr>
<td>java</td>
<td>corretto11</td>
<td>java: corretto11.x</td>
</tr>
<tr>
<td></td>
<td>corretto17</td>
<td>java: corretto17.x</td>
</tr>
<tr>
<td>nodejs</td>
<td>16.19</td>
<td>nodejs: 16.x</td>
</tr>
<tr>
<td>openssl</td>
<td>1.0</td>
<td>openssl: 1.x</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>7.2</td>
<td>php: 7.x</td>
</tr>
</tbody>
</table>
Amazon CodeCatalyst User Guide
Using Dev Environments with an IDE

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version</th>
<th>Specific major and latest minor version</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>3.9</td>
<td>python: 3.x</td>
</tr>
<tr>
<td></td>
<td>3.10</td>
<td></td>
</tr>
<tr>
<td>ruby</td>
<td>3.2</td>
<td>ruby: 3.x</td>
</tr>
<tr>
<td>terraform</td>
<td>1.4</td>
<td>terraform: 1.x</td>
</tr>
</tbody>
</table>

**Universal image 3.0 runtime versions**

The following table lists the available runtimes for universal-image:3.0.

<table>
<thead>
<tr>
<th>Runtime name</th>
<th>Version</th>
<th>Specific major and latest minor version</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws cli</td>
<td>2.11</td>
<td>aws-cli: 2.x</td>
</tr>
<tr>
<td>docker compose</td>
<td>2.17</td>
<td>docker-compose: 2.x</td>
</tr>
<tr>
<td>dotnet</td>
<td>6.0</td>
<td>dotnet: 6.x</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
<td>dotnet: 7.x</td>
</tr>
<tr>
<td>golang</td>
<td>1.21</td>
<td>golang: 1.x</td>
</tr>
<tr>
<td>java</td>
<td>corretto11</td>
<td>java: corretto11.x</td>
</tr>
<tr>
<td></td>
<td>corretto17</td>
<td>java: corretto17.x</td>
</tr>
<tr>
<td>nodejs</td>
<td>18.17</td>
<td>nodejs: 18.x</td>
</tr>
<tr>
<td></td>
<td>20.6</td>
<td>nodejs: 20.x</td>
</tr>
<tr>
<td>openssl</td>
<td>3.0</td>
<td>openssl: 3.x</td>
</tr>
<tr>
<td>php</td>
<td>8.2</td>
<td>php: 8.x</td>
</tr>
<tr>
<td>python</td>
<td>3.9</td>
<td>python: 3.x</td>
</tr>
<tr>
<td></td>
<td>3.11</td>
<td></td>
</tr>
<tr>
<td>ruby</td>
<td>3.2</td>
<td>ruby: 3.x</td>
</tr>
<tr>
<td>terraform</td>
<td>1.5</td>
<td>terraform: 1.x</td>
</tr>
</tbody>
</table>

**Using Dev Environments with an IDE**

You can use Dev Environments to quickly work on the code stored in the source repositories of your project. Dev Environments increase your development velocity because you can start coding immediately in a project-specific, fully functioning cloud development environment with a supported integrated development environment (IDE).
For information about working with CodeCatalyst from an IDE, see the following documentation.

- Amazon CodeCatalyst for JetBrains IDEs
- Amazon CodeCatalyst for VS Code
- Amazon CodeCatalyst for AWS Cloud9

Quotas for Dev Environments in CodeCatalyst

The following table describes quotas and limits for Dev Environments in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see Quotas for CodeCatalyst (p. 620).

<table>
<thead>
<tr>
<th>Quota Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Free tier Dev Environment hours per month</td>
<td>60</td>
</tr>
<tr>
<td>Amount of Free tier Dev Environment storage per space</td>
<td>64 GB</td>
</tr>
<tr>
<td>Amount of Free tier Dev Environment compute</td>
<td>2 vCPU, 4 GB RAM</td>
</tr>
<tr>
<td>Number of Standard tier Dev Environment hours per month</td>
<td>200</td>
</tr>
<tr>
<td>Amount of Standard tier Dev Environment storage per space</td>
<td>128 GB</td>
</tr>
</tbody>
</table>
Build, test, and deploy with workflows in CodeCatalyst

After writing your application code in a CodeCatalyst Dev Environment (p. 113) and pushing it to your CodeCatalyst source repository (p. 79), you're ready to deploy it. The way to do this automatically is through a workflow.

A workflow is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or actions, to take during a workflow run. A workflow also defines the events, or triggers, that cause the workflow to start. To set up a workflow, you create a workflow definition file using the CodeCatalyst console's visual or YAML editor.

**Tip**
For a quick look at how you might use workflows in a project, create a project with a blueprint. Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

About the workflow definition file

A workflow definition file is a YAML file that describes your workflow. The file is stored in a `~/.codecatalyst/workflows/` folder in the root of your source repository (p. 89). The file can have a `.yml` or `.yaml` extension.

The following is an example of a simple workflow definition file. We explain each line of this example in the table that follows.

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest
```

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: MyWorkflow</td>
<td>Specifies the name of the workflow. For more information about the Name property, see Top-level properties (p. 356).</td>
</tr>
<tr>
<td>SchemaVersion: 1.0</td>
<td>Specifies the workflow schema version. For more information about the SchemaVersion property, see Top-level properties (p. 356).</td>
</tr>
<tr>
<td>Line</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>RunMode: QUEUED</td>
<td>Indicates how CodeCatalyst handles multiple runs. For more information about the run mode, see Configuring queued, superseded, and parallel runs (p. 317).</td>
</tr>
<tr>
<td>Triggers:</td>
<td>Specifies the logic that will cause a workflow run to start. For more information about triggers, see Working with triggers (p. 327).</td>
</tr>
<tr>
<td>- Type: PUSH Branches: - main</td>
<td>Indicates that the workflow must start whenever you push code to the main branch of the default source repository. For more information about the workflow source, see Working with sources (p. 325).</td>
</tr>
<tr>
<td>Actions:</td>
<td>Defines the tasks to perform during a workflow run. In this example, the Actions section defines a single action called Build. For more information about actions, see Working with actions (p. 249).</td>
</tr>
<tr>
<td>Build:</td>
<td>Defines the properties for the Build action. For more information about the build action, see Building using workflows in CodeCatalyst (p. 138).</td>
</tr>
<tr>
<td>Identifier: aws/build@v1</td>
<td>Specifies the unique, hard-coded identifier for the build action.</td>
</tr>
<tr>
<td>Inputs: Sources: - WorkflowSource</td>
<td>Indicates that the build action should look in the WorkflowSource source repository to find the files it needs to complete its processing. For more information, see Working with sources (p. 325).</td>
</tr>
<tr>
<td>Configuration:</td>
<td>Contains the configuration properties that are specific to the build action.</td>
</tr>
<tr>
<td>Steps: - Run: docker build -t MyApp:latest .</td>
<td>Tells the build action to build a Docker image called MyApp and tag it with latest.</td>
</tr>
</tbody>
</table>

For a complete list of all the properties available in the workflow definition file, see the Workflow definition reference (p. 354).

**Using the CodeCatalyst console's visual and YAML editors**

To create and edit the workflow definition file, you can use your preferred editor, but we recommend using the CodeCatalyst console's visual editor or YAML editor. These editors offer helpful file validation to help ensure YAML property names, values, nesting, spacing, capitalization, and so on, are correct.
Discovering workflows

The following image shows a workflow in the visual editor. The visual editor offers you a complete user interface through which to create and configure your workflow definition file. The visual editor includes a workflow diagram (1) showing the workflow's main components, and a configuration area (2).

Alternatively, you can use the YAML editor, shown in the next image. Use the YAML editor to paste in large code blocks (from a tutorial, for example), or to add advanced properties that are not offered through the visual editor.

You can toggle from the visual editor to the YAML editor to see the effect that your configurations have on the underlying YAML code.

Discovering workflows

You can view your workflow on the **Workflows** summary page, along with other workflows you’ve set up in the same project.
The following image shows the **Workflows** summary page. It is populated with two workflows: **BuildToProd** and **UnitTests**. You can see that both have been run a few times. You can choose **Recent runs** to quickly see the run history, or choose the name of the workflow to see the workflow's YAML code and other detailed information.

**Viewing workflow run details**

You can view the details of a workflow run by choosing the run in the **Workflows** summary page. The following image shows the details of a workflow run called **Run-cc11d** that was started automatically on a commit to source. The workflow diagram indicates that an action has failed (1). You can navigate to the logs (2) to view the detailed log messages and troubleshoot issues. For more information about workflow runs, see **Working with runs (p. 314)**.

**Next steps**

To learn more about workflows concepts, see **Workflows concepts (p. 132)**.
Workflows concepts

Here are some concepts and terms to know when building, testing, or deploying your code with workflows in CodeCatalyst.

Workflows

A workflow is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or actions, to take during a workflow run. A workflow also defines the events, or triggers, that cause the workflow to start. To set up a workflow, you create a workflow definition file using the CodeCatalyst console’s visual or YAML editor.

Tip
For a quick look at how you might use workflows in a project, create a project with a blueprint. Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

For more information about workflows, see Working with workflows (p. 246).

Workflow definition files

A workflow definition file is a YAML file that describes your workflow. The file is stored in a ~/.codecatalyst/workflows/ folder in the root of your source repository (p. 89). The file can have a .yml or .yaml extension.

For more information about the workflow definition file, see Workflow definition reference (p. 354).

Actions

An action is the main building block of a workflow, and defines a logical unit of work to perform during a workflow run. Typically, a workflow includes multiple actions that run sequentially or in parallel depending on how you’ve configured them.

For more information about actions, see Working with actions (p. 249).

Action groups

An action group contains one or more actions. Grouping actions into action groups helps you keep your workflow organized, and also allows you to configure dependencies between different groups of actions.

For more information about action groups, see Grouping actions into action groups (p. 279).

Artifacts

An artifact is the output of a workflow action, and typically consists of a folder or archive of files. Artifacts are important because they allow you to share files and information between actions.

For more information about artifacts, see Working with artifacts (p. 286).

Compute

Compute refers to the computing engine (the CPU, memory, and operating system) managed and maintained by CodeCatalyst to run your workflows. You associate a compute type with either your workflow or your workflow actions (p. 249).
Environments

An environment, not to be confused with a Dev Environment, is where code is deployed to. It usually contains an instance of a running application along with its associated infrastructure. You can give your environment a name such as development, test, staging, or production. Any deployments generated by CodeCatalyst to an environment will appear on the Environments page. To set up an environment, you give it a name, such as my-production-environment, and then associate it with your AWS account.

In addition to displaying deployment information, an environment also serves as the mechanism through which to assign AWS IAM roles to workflow actions.

For more information about environments, see Working with environments (p. 307).

Reports

A report contains details about tests that occur during a workflow run. You can create reports such as a test report, a code coverage report, a software composition analysis report, and a static analysis report. You can use a report to help troubleshoot a problem during a workflow. If you have many reports from multiple workflows, you can use your reports to view trends and failure rates to help you optimize your applications and deployment configurations.

For more information about reports, see Test report types (p. 147).

Runs

A run is a single iteration of a workflow. During a run, CodeCatalyst performs the actions defined in the workflow configuration file and outputs the associated logs, artifacts, and variables.

For more information about runs, see Working with runs (p. 314).

Sources

A source, also called an input source, is a source repository that a workflow action needs access to in order to carry out its tasks. For example, a workflow action might need to access a source to obtain unit tests and run them against your application source files.

For more information about sources, see Working with sources (p. 325).

Variables

A variable is a key-value pair that contains information that you can reference in your CodeCatalyst workflow.

For more information about variables, see Working with variables (p. 337).

Workflow triggers

A workflow trigger, or simply a trigger, allows you to start a workflow run automatically when certain events occur, like a code push. You might want to configure triggers to free your software developers from having to start workflow runs manually through the CodeCatalyst console.

You can use three types of trigger:
• **Push** – A code push trigger causes a workflow run to start whenever a commit is pushed. The workflow run uses the files in the branch that your pushing to (called, the *destination* branch).

• **Pull request** – A pull request trigger causes a workflow run to start whenever a pull request is either created, revised, or closed. The workflow run uses the files in the branch that you're pulling from (called the *source* branch).

• **Schedule** – A schedule trigger causes a workflow run to start on a schedule that you define. Consider using a schedule trigger to run nightly builds of your software so that the latest build is ready for your software developers to work on the next morning.

You can use push, pull request, and schedule triggers alone or in combination in the same workflow.

For more information about triggers, see *Working with triggers (p. 327)*.

### Getting started with workflows in CodeCatalyst

In this tutorial, you'll learn how to create and configure your first workflow.

**Tip**

Prefer to start with a preconfigured workflow? See *Creating a project with a blueprint (p. 58)*, which includes instructions for setting up a project with a functioning workflow, sample application, and other resources.

**Topics**

• **Prerequisites (p. 134)**
• **Step 1: Create and configure your workflow (p. 135)**
• **Step 2: Save your workflow with a commit (p. 136)**
• **Step 3: View run results (p. 136)**
• **(Optional) Step 4: Clean up (p. 136)**

### Prerequisites

Before you begin:

• You need a CodeCatalyst **space**. For more information, see *Creating additional spaces in CodeCatalyst (p. 53)*.

• In your CodeCatalyst space, you need an empty, **Start from scratch** CodeCatalyst **project** called:

```bash
codecatalyst-project
```

For more information, see *Creating an empty project in Amazon CodeCatalyst (p. 58)*.

• In your project, you need a CodeCatalyst **repository** called:

```bash
codecatalyst-source-repository
```

For more information, see *Creating a source repository (p. 89)*.

**Note**

If you have an existing project and source repository, you can use them; however, creating new ones makes cleanup easier at the end of this tutorial.
Step 1: Create and configure your workflow

In this step, you create and configure a workflow that automatically builds and tests your source code when changes are made.

To create your workflow

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose Create workflow.

   The workflow definition file appears in the CodeCatalyst console’s YAML editor.

To configure your workflow

You can configure your workflow in the Visual editor, or the YAML editor. Let’s start with the YAML editor and then switch to the visual editor.

1. Choose + Actions to see a list of workflow actions that you can add to your workflow.
2. In the Build action, choose + to add the action’s YAML to your workflow definition file. Your workflow now looks similar to the following.

   ```yaml
   Name: Workflow_fe47
   SchemaVersion: "1.0"

   # Optional - Set automatic triggers.
   Triggers:
   - Type: Push
     Branches:
     - main

   # Required - Define action configurations.
   Actions:
   Build_f0:
     Identifier: aws/build@v1
     Inputs:
     Sources:
     - WorkflowSource # This specifies that the action requires this workflow as a source
     Outputs:
     AutoDiscoverReports:
     Enabled: true
     # Use as prefix for the report files
     ReportNamePrefix: rpt
     Configuration:
     Steps:
     - Run: echo "Hello, World!"
     - Run: echo "<xml version="1.0" encoding="UTF-8" ?>" >> report.xml
     - Run: echo "<testsuite tests="1" name="TestAgentJUnit"/>
     - Run: echo "<testcase classname="TestAgentJUnit" name="Dummy Test"/>
   ``

   The workflow copies the files in the WorkflowSource source repository to the compute machine running the Build_f0 action, prints Hello, World! to the logs, discovers test reports on the compute machine, and outputs them to the CodeCatalyst console’s Reports page.

3. Choose Visual to view the workflow definition file in the visual editor. The fields in the visual editor let you configure the YAML properties shown in the YAML editor.
Step 2: Save your workflow with a commit

In this step, you save your changes. Because workflows are stored as .yaml files in your repository, you save your changes with commits.

To commit your workflow changes

1. (Optional) Choose Validate to make sure the workflow's YAML code is valid.
2. Choose Commit.
3. In Workflow file name, enter a name for your workflow configuration file, like my-first-workflow.
4. In Commit message, enter a message to identify your commit, like create my-first-workflow.yaml.
5. In Repository, choose the repository you want to save the workflow in (codecatalyst-repository).
6. In Branch name, choose the branch you want to save the workflow in (main).
7. Choose Commit.

Your new workflow appears in the list of workflows. It might take several moments to appear.

Because workflows are saved with commits, and because the workflow has a code push trigger configured, saving the workflow starts a workflow run automatically.

Step 3: View run results

In this step, you navigate to the run that was started from your commit and view the results.

To view run results

1. Choose the name of your workflow, for example, Workflow_fe47.

   A workflow diagram showing the label of your source repository (WorkflowSource) and the build action (for example, Build_f0).

2. In the workflow run diagram, choose the build action (for example, Build_f0).
3. Review the contents of the Logs, Reports, Configuration, and Variables tabs. These tabs show you the results of your build action.

   For more information, see Viewing the results of a build action (p. 140).

(Optional) Step 4: Clean up

In this step, you clean up the resources that you created in this tutorial.

To delete resources

1. If you created a new project for this tutorial, delete it. For instructions, see Deleting a project in Amazon CodeCatalyst (p. 62). Deleting the project also deletes the source repository and workflow.
2. Delete the workflow if it is not yet deleted. For instructions, see To delete a workflow (p. 248).
3. Delete the repository if it is not yet deleted. For instruction, see Deleting a source repository (p. 93).
Viewing code quality and deployment status by commit in CodeCatalyst

At any time in the development lifecycle, it's important to know the deployment status of specific commits, such as bug fixes, new features, or other impactful changes. Consider the following scenarios in which deployment status tracking capability is helpful to development teams:

- As a developer, you've made a fix to address a bug and you want to report the status of its deployment across your team's deployment environments.
- As a release manager, you want to view a list of deployed commits to track and report their deployment status.

CodeCatalyst provides a view you can use to determine at a glance where individual commits or changes have been deployed, and to which environment. This view includes:

- A list of commits.
- The status of deployments that include the commits.
- The environments in which the commits are successfully deployed.
- The status of any tests run against the commits in your CI/CD workflow.

The following procedure details how to navigate to and use this view to track changes in your project.

**Note**

Tracking deployment status by commit is only supported with CodeCatalyst repositories (p. 79). You cannot use this feature with a GitHub repository (p. 558).

**To track deployment status by commit**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Change tracking**.
4. In the two dropdown lists at the top of the main pane, choose the source repository and branch that contain the commits whose release status you want to view.
5. Choose **View changes**.

A list of commits appears.

For each commit, you can view the following:

- Commit information such as ID, author, message, and when it was committed. For more information, see [Source repositories in CodeCatalyst](p. 79).
- The status of deployments to each environment. For more information, see [Working with environments](p. 307).
- Test and code coverage results. For more information, see [Testing using workflows in CodeCatalyst](p. 146).

**Note**

Software Composition Analysis (SCA) results are not displayed.

6. (Optional) To view more information about the changes related to a specific commit, including the latest deployment and detailed code coverage and unit test information, choose **View details** for that commit.
Building using workflows in CodeCatalyst

Using CodeCatalyst workflows (p. 128), you can build applications and other resources.

Topics
- How do I build an application? (p. 138)
- Benefits of the build action (p. 138)
- Alternatives to the build action (p. 139)
- Adding the build action (p. 139)
- Viewing the results of a build action (p. 140)
- Tutorial: Upload artifacts to Amazon S3 (p. 140)

How do I build an application?

To build an application or resource in CodeCatalyst, you first create a workflow, and then specify a build action inside it.

A build action is a workflow building block that compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.

You add a build action to your workflow using the CodeCatalyst console's visual editor or YAML editor.

The high-level steps to build an application or resource are as follows.

To build an application (high-level tasks)

1. In CodeCatalyst, you add source code for an application you want to build. For more information, see Working with source repositories in CodeCatalyst (p. 89).
2. In CodeCatalyst, you create a workflow. The workflow is where you define how to build, test, and deploy your application. For more information, see Getting started with workflows in CodeCatalyst (p. 134).
3. (Optional) In the workflow, you add a trigger that indicates the events that will cause the workflow to start automatically. For more information, see Working with triggers (p. 327).
4. In the workflow, you add a build action that compiles and packages your application or resource source code. Optionally, you can also have the build action run unit tests, generate reports, and deploy your application if you don't want to use a test or deploy action for these purposes. For more on the test and deploy actions, see Adding the build action (p. 139).
5. (Optional) In the workflow, you add a test action and a deploy action to test and deploy your application or resource. You can choose from several pre-configured actions to deploy your application to different targets, such as Amazon ECS. For more information, see Testing using workflows in CodeCatalyst (p. 146), and Deploying using workflows in CodeCatalyst (p. 163).
6. You start the workflow either manually or automatically through a trigger. The workflow runs the build, test, and deploy actions in sequence to build, test, and deploy your application and resources to the target. For more information, see Starting a workflow run (p. 314).

Benefits of the build action

Using the build action within a workflow has the following benefits:

- Fully managed – The build action eliminates the need to set up, patch, update, and manage your own build servers.
• **On demand** – The build action scales on demand to meet your build needs. You pay only for the number of build minutes you consume. For more information, see *Working with compute (p. 293)*.

• **Out of the box** – CodeCatalyst includes prepackaged runtime environment Docker images that are used to run all your workflow actions, including build actions. These images come preconfigured with useful tools for building applications such as the AWS CLI and Node.js. You can configure CodeCatalyst to use a build image that you supply from a public or private registry. For more information, see *Working with runtime environment Docker images (p. 302)*.

### Alternatives to the build action

If you’re using a build action to deploy your application, consider using a CodeCatalyst *deploy action* instead. Deploy actions perform behind-the-scenes configuration that you would otherwise have to write manually if you’re using a build action. For more information on the available deploy actions, see *List of deploy actions (p. 164)*.

You can also use AWS CodeBuild to build your applications. For more information, see *What is CodeBuild?*.

**Topics**

- Adding the build action (p. 139)
- Viewing the results of a build action (p. 140)
- Tutorial: Upload artifacts to Amazon S3 (p. 140)

### Adding the build action

Use the following procedure to add a build action to your CodeCatalyst workflow.

**Visual**

*To add a build action using the visual editor*

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose CI/CD, and then choose Workflows.
3. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
4. Choose Edit.
6. Choose Actions.
7. In Actions, choose Build.
8. In the Inputs and Configuration tabs, complete the fields according to your needs. For a description of each field, see the Build and test action reference (p. 364). This reference provides detailed information on each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.
9. (Optional) Choose Validate to validate the workflow’s YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

**YAML**

*To add a build action using the YAML editor*

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose CI/CD, and then choose Workflows.
3. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

4. Choose Edit.

5. Choose YAML.

6. Choose Actions.

7. In Actions, choose Build.

8. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the Build and test action reference (p. 364).

9. (Optional) Choose Validate to validate the workflow's YAML code before committing.

10. Choose Commit, enter a commit message, and choose Commit again.

**Build action definition**

The build action is defined as a set of YAML properties inside your workflow definition file. For information on these properties, see Build and test action reference (p. 364) in the Workflow definition reference (p. 354).

**Viewing the results of a build action**

Use the following instructions to view the results of a build action, including the generated logs, reports, and variables.

**To view the results of a build action**

1. In the navigation pane, choose CI/CD, and then choose Workflows.

2. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

3. In the workflow diagram, choose the name of your build action, for example, Build.

4. To view the logs for the build run, choose Logs. The logs for the various build phases are displayed. You can expand logs as needed.

5. To view the test reports produced by the build action, choose Reports, or in the navigation pane, choose Reports. For more information, see Test report types (p. 147).

6. To view the configuration used for the build action, choose Configuration. For more information, see Adding the build action (p. 139).

7. To view the variables used by the build action, choose Variables. For more information, see Working with variables (p. 337).

**Tutorial: Upload artifacts to Amazon S3**

In this tutorial, you learn how to upload artifacts to an Amazon S3 bucket using a CodeCatalyst workflow (p. 132) that includes a couple of build actions (p. 132). These actions run in series when the workflow starts. The first build action generates two files, Hello.txt and Goodbye.txt, and bundles them into a build artifact. The second build action uploads the artifact to Amazon S3. You'll configure the workflow to run every time you push a commit to your source repository.

**Topics**

- Prerequisites (p. 141)
- Step 1: Create an AWS role (p. 141)
- Step 2: Create an Amazon S3 bucket (p. 143)
- Step 3: Create a source repository (p. 143)
Tutorial: Upload artifacts to Amazon S3

- Step 4: Create a workflow (p. 144)
- Step 5: Verify the results (p. 146)
- Clean up (p. 146)

Prerequisites

Before you begin, you need the following:

- You need a CodeCatalyst space with a connected AWS account. For more information, see Creating additional spaces in CodeCatalyst (p. 53).
- In your space, you need an empty, Start from scratch CodeCatalyst project called:

  codecatalyst-artifact-project

  For more information, see Creating an empty project in Amazon CodeCatalyst (p. 58).
- In your project, you need a CodeCatalyst environment called:

  codecatalyst-artifact-environment

  Configure this environment as follows:
  - Choose any type, such as Development.
  - Connect your AWS account to it.

  For more information, see Working with environments (p. 307).

Step 1: Create an AWS role

In this step, you create an AWS IAM role which you will later assign to the build action in your workflow. This role grants the CodeCatalyst build action permission to access your AWS account and write to Amazon S3 where your artifact will be stored. The role is called the Build role.

  **Note**

  If you already have a build role that you created for another tutorial, you can use it for this tutorial too. Just make sure it has the permissions and trust policy shown in the following procedure.

  For more information on IAM roles, see IAM roles in the AWS AWS Identity and Access Management User Guide.

  **To create a build role**

  1. Create a policy for the role, as follows:

     a. Sign in to AWS.
     b. Open the IAM console at https://console.aws.amazon.com/iam/.
     c. In the navigation pane, choose Policies.
     d. Choose Create policy.
     e. Choose the JSON tab.
     f. Delete the existing code.
     g. Paste the following code:


```json
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:*"
    ],
    "Resource": "arn:aws:s3:::<bucket-name>*"
  }
}
```
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:ListBucket"
    ],
    "Resource": "*"
  }
]
}

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

h. Choose **Next: Tags**.
i. Choose **Next: Review**.
j. In **Name**, enter:

```
codecatalyst-s3-build-policy
```
k. Choose **Create policy**.

You have now created a permissions policy.

2. Create the build role, as follows:

a. In the navigation pane, choose **Roles**, and then choose **Create role**.
b. Choose **Custom trust policy**.
c. Delete the existing custom trust policy.
d. Add the following custom trust policy:

```
{  
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [  
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```
e. Choose **Next**.
f. In **Permissions policies**, search for codecatalyst-s3-build-policy and select its check box.
g. Choose **Next**.
h. For **Role name**, enter:
i. For Role description, enter:
   - **CodeCatalyst build role**

j. Choose Create role.

You have now created a build role with a trust policy and permissions policy.

### Step 2: Create an Amazon S3 bucket

In this step, you create an Amazon S3 bucket where the Hello.txt and Goodbye.txt artifacts will be uploaded.

**To create an Amazon S3 bucket**

1. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. In the main pane, choose Create bucket.
3. For Bucket name, enter: `codecatalyst-artifact-bucket`
4. For AWS Region, choose a Region. This tutorial assumes you chose **US West (Oregon) us-west-2**. For information about Regions supported by Amazon S3, see [Amazon Simple Storage Service endpoints and quotas](https://docs.aws.amazon.com/AmazonS3/latest/userguide/region endpoint.html) in the AWS General Reference.
5. At the bottom of the page, choose Create bucket.
6. Copy the name of the bucket you just created, for example: `codecatalyst-artifact-bucket`

You have now created a bucket called **codecatalyst-artifact-bucket** in the US West (Oregon) us-west-2 Region.

### Step 3: Create a source repository

In this step, you create a source repository in CodeCatalyst. This repository is used to store the tutorial's workflow definition file.

For more information on source repositories, see [Creating a source repository](https://docs.aws.amazon.com/CodeCatalyst/latest/UserGuide/creating-source-repository.html) (p. 89).

**To create a source repository**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your project, `codecatalyst-artifact-project`.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Create repository.
5. In Repository name, enter: `codecatalyst-artifact-source-repository`
6. Choose Create.
You have now created a repository called codecatalyst-artifact-source-repository.

**Step 4: Create a workflow**

In this step, you create a workflow that consists of the following building blocks that run sequentially:

- A trigger – This trigger starts the workflow run automatically when you push a change to your source repository. For more information on triggers, see [Working with triggers](p. 327).
- A build action called GenerateFiles – On trigger, the GenerateFiles action creates two files, Hello.txt and Goodbye.txt, and packages them into an output artifact called codecatalystArtifact.
- Another build action called Upload – On completion of the GenerateFiles action, the Upload action runs the AWS CLI command `aws s3 sync` to upload the files in the codecatalystArtifact and in your source repository to your Amazon S3 bucket. The AWS CLI comes pre-installed and pre-configured on the CodeCatalyst compute platform, so you don't need to install or configure it.

For more information on the pre-packaged software on the CodeCatalyst compute platform, see [Working with runtime environment Docker images](p. 302). For more information on the AWS CLI's `aws s3 sync` command, see `sync` in the [AWS CLI Command Reference](p. 138).

For more information on the build action, see [Building using workflows in CodeCatalyst](p. 138).

**To create a workflow**

1. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
2. Choose **Create workflow**.
3. Delete the YAML sample code.
4. Add the following YAML code:
Artifacts:
- codecatalystArtifact

Configuration:
Steps:
# Upload the output artifact to the S3 bucket.
- Run: aws s3 sync . s3://codecatalyst-artifact-bucket

In the code above, replace:

- `codecatalyst-artifact-environment` with the name of the environment you created in Prerequisites (p. 141).
- `codecatalyst-account-connection` with the name of the account connection you created in Prerequisites (p. 141).
- `codecatalyst-s3-build-role` with the name of the build role that you created in Step 1: Create an AWS role (p. 141).
- `codecatalyst-artifact-bucket` with the name of the Amazon S3 bucket you created in Step 2: Create an Amazon S3 bucket (p. 143).

For information about the properties in this file, see the Build and test action reference (p. 364).

5. (Optional) Choose Validate to make sure the YAML code is valid before committing.

6. Choose Commit.

7. On the Commit workflow dialog box, enter the following:
   a. For Workflow file name, leave the default, codecatalyst-artifact-workflow.
   b. For Commit message, enter:
      ```
      add initial workflow file
      ```
   c. For Repository, choose codecatalyst-artifact-source-repository.
   d. For Branch name, choose main.
   e. Choose Commit.

You have now created a workflow. A workflow run starts automatically because of the trigger defined at the top of the workflow. Specifically, when you committed (and pushed) the codecatalyst-artifact-workflow.yaml file to your source repository, the trigger started the workflow run.

To view the workflow run in progress

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose the workflow you just created: codecatalyst-artifact-workflow.
3. Choose GenerateFiles to see the first build action progress.
4. Choose Upload to see the second build action progress.
5. When the Upload action finishes, do the following:
   - If the workflow run succeeded, go to the next procedure.
   - If the workflow run failed, choose Logs to troubleshoot the issue.
Step 5: Verify the results

After the workflow runs, go to the Amazon S3 service and look in your `codecatalyst-artifact-bucket` bucket. It should now include the following files and folders:

```
  |   .aws/
  |   .git/
  |Goodbye.txt
  |Hello.txt
  |README.md
```

The Goodbye.txt and Hello.txt files were uploaded because they were part of the codcatalystArtifact artifact. The .aws/, .git/, and README.md files were uploaded because they were in your source repository.

Clean up

Clean up in CodeCatalyst and AWS to avoid being charged for these services.

To clean up in CodeCatalyst

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Delete the `codecatalyst-artifact-source-repository` source repository.
3. Delete the `codecatalyst-artifact-workflow` workflow.

To clean up in AWS

1. Clean up in Amazon S3, as follows:
   a. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
   b. Delete the files in the `codecatalyst-artifact-bucket` bucket.
   c. Delete the `codecatalyst-artifact-bucket` bucket.
2. Clean up in IAM, as follows:
   b. Delete the `codecatalyst-s3-build-policy`.
   c. Delete the `codecatalyst-s3-build-role`.

Testing using workflows in CodeCatalyst

In CodeCatalyst, you can run tests as part of different workflow actions, such as build and test. These workflow actions can all generate reports. A test action is a workflow action that produces test, code coverage, software composition analysis, and static analysis reports. These reports are displayed in the CodeCatalyst console.

Topics

- [Test report types](#) (p. 147)
- [Adding the test action](#) (p. 149)
- [Configuring reports](#) (p. 149)
- [Using universal-test-runner](#) (p. 158)
- [Best practices](#) (p. 160)
Test report types

The Amazon CodeCatalyst test action supports the following types of reports. For an example on how to format these reports in your YAML, see Example: Configuring reports (p. 152).

Topics

- Test reports (p. 147)
- Code coverage reports (p. 147)
- Software composition analysis reports (p. 148)
- Static analysis reports (p. 148)

Test reports

In CodeCatalyst, you can configure unit tests, integration tests, and system tests that run during builds. Then CodeCatalyst can create reports that contain the results of your tests.

You can use a test report to help troubleshoot problems with your tests. If you have many test reports from multiple builds, you can use your test reports to view failure rates to help you optimize your builds.

You can use the following test report file formats:

- Cucumber JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx, .xml)

Code coverage reports

In CodeCatalyst, you can generate code coverage reports for your tests. CodeCatalyst provides the following code coverage metrics:

Line coverage

Measures how many statements your tests cover. A statement is a single instruction, not including comments.

\[
\text{line coverage} = \frac{\text{total lines covered}}{\text{total number of lines}}
\]

Branch coverage

Measures how many branches your tests cover out of every possible branch of a control structure such as an if or case statement.

\[
\text{branch coverage} = \frac{\text{total branches covered}}{\text{total number of branches}}
\]

The following code coverage report file formats are supported:

- JaCoCo XML (.xml)
- SimpleCov JSON (generated by simplecov, not simplecov-json, .json)
• Clover XML (version 3, .xml)
• Cobertura XML (.xml)
• LCOV (.info)

Software composition analysis reports

In CodeCatalyst, you can use software composition analysis (SCA) tools to analyze components of your application and check for known security vulnerabilities. You can discover and parse SARIF reports that detail vulnerabilities with varying severities and ways to fix them. Valid severity values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

The following SCA report file formats are supported:

• SARIF (.sarif, .json)

Static analysis reports

You can use static analysis (SA) reports to identify source-level code defects. In CodeCatalyst, you can generate SA reports to help resolve issues in your code before you deploy it. These issues include bugs, security vulnerabilities, quality issues, and other vulnerabilities. Valid severity values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, and INFORMATIONAL.

CodeCatalyst provides the following SA metrics:

Bugs

Identifies a number of possible bugs found in your source code. These bugs can include issues regarding memory safety. The following is an example of a bug.

```c
// The while loop will inadvertently index into array x out-of-bounds
int x[64];
while (int n = 0; n <= 64; n++) {
    x[n] = 0;
}
```

Security vulnerabilities

Identifies a number of possible security vulnerabilities found in your source code. These security vulnerabilities can include issues such as storing your secret tokens in plaintext.

Quality issues

Identifies a number of possible quality issues found in your source code. These quality issues can include issues regarding style conventions. The following is an example of a quality issue.

```c
// The function name doesn't adhere to the style convention of camelCase
int SUBTRACT(int x, int y) {
    return x-y
}
```

Other vulnerabilities

Identifies a number of possible other vulnerabilities found in your source code.

CodeCatalyst supports the following SA report file formats:

• PyLint (.py)
Adding the test action

Use the following procedure to add a test action to your workflow.

To add a test action

- Follow the instructions in Adding the build action (p. 139). The instructions for adding a build action and test action are exactly the same.

Test action definition

The test action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see Build and test action reference (p. 364) in the Workflow definition reference (p. 354).

Configuring reports

This section describes how to configure a quality report.

Topics

- Configuring auto-discovery and manual reports (p. 149)
- Configuring success criteria for reports (p. 149)
- Example: Configuring reports (p. 152)
- SARIF support for Software Composition Analysis and Static Analysis reports (p. 153)

Configuring auto-discovery and manual reports

When auto-discovery is enabled, CodeCatalyst searches all inputs passed into the action, and all files generated by the action itself, looking for test, code coverage, software composition analysis (SCA), and static analysis (SA) reports. You can view and manipulate each of these reports in CodeCatalyst.

You can also manually configure which reports are generated. You can specify the type of report you'd like to generate as well as the file format. For more information, see Test report types (p. 147).

Configuring success criteria for reports

You can set the values that determine the success criteria for a test, code coverage, software composition analysis (SCA), or static analysis (SA) report.

Success criteria are thresholds that determine whether a report passes or fails. CodeCatalyst first generates your report, which can be a test, code coverage, SCA, or SA report, and then applies the success criteria to the generated reports. It then shows whether the success criteria were met, and to what extent. If any report does not meet the specified success criteria, the CodeCatalyst action that specified the success criteria fails.

If you do not specify success criteria, then:

- The CodeCatalyst report that is generated based on your raw reports will not display success criteria.
- Success criteria will not be used to determine whether the associated workflow action passes or fails.
To configure success criteria

1. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
2. Choose a workflow containing an action that generates a report. This is the report for which you want to apply success criteria. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
3. Choose **Edit**.
4. Choose **Visual**.
5. In the workflow diagram, choose the action that you have configured to generate CodeCatalyst reports.
6. Choose the **Outputs** tab.
7. Under **Auto-discover reports** or under **Manually configure reports**, choose **Success criteria**.

Success criteria appear. Depending on your previous selections, you may see any or all of these options:

**Pass rate**

Specify the percentage of tests in a test report that must pass for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. The pass rate criteria are applied only to test reports. For more information about test reports, see **Test reports** (p. 147).

**Line coverage**

Specify the percentage of lines in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. Line coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see **Code coverage reports** (p. 147).

**Branch coverage**

Specify the percentage of branches in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. Branch coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see **Code coverage reports** (p. 147).

**Vulnerabilities (SCA)**

Specify the maximum number and severity of vulnerabilities permitted in the SCA report for the associated CodeCatalyst report to be marked as passed. To specify vulnerabilities, you must specify:

- The minimum severity of the vulnerabilities you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

  For example, if you choose HIGH, then HIGH and CRITICAL vulnerabilities will be tallied.

- The maximum number of vulnerabilities of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Vulnerabilities criteria are applied only to SCA reports. For more information about SCA reports, see **Software composition analysis reports** (p. 148).

**Bugs**
Specify the maximum number and severity of bugs permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify bugs, you must specify:

- The minimum severity of the bugs you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

For example, if you choose HIGH, then HIGH and CRITICAL bugs will be tallied.

- The maximum number of bugs of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Bugs criteria are applied only to SA reports. For more information about SA reports, see Static analysis reports (p. 148).

Security vulnerabilities

Specify the maximum number and severity of security vulnerabilities permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify security vulnerabilities, you must specify:

- The minimum severity of the security vulnerabilities you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

For example, if you choose HIGH, then HIGH and CRITICAL security vulnerabilities will be tallied.

- The maximum number of security vulnerabilities of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Security vulnerabilities criteria are applied only to SA reports. For more information about SA reports, see Static analysis reports (p. 148).

Quality issues

Specify the maximum number and severity of quality issues permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify quality issues, you must specify:

- The minimum severity of the quality issues you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

For example, if you choose HIGH, then HIGH and CRITICAL quality issues will be tallied.

- The maximum number of quality issues of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Quality issues criteria are applied only to SA reports. For more information about SA reports, see Static analysis reports (p. 148).

8. Choose Commit.

9. Run your workflow to have CodeCatalyst apply success criteria to your raw reports, and regenerate the associated CodeCatalyst reports with success criteria information included. For more information, see Starting a workflow run (p. 314).
To configure success criteria

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose a workflow containing an action that generates a report. This is the report for which you want to apply success criteria. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
3. Choose Edit.
4. Choose YAML.
5. In the workflow diagram, choose the action that you have configured to generate CodeCatalyst reports.
6. In the details pane, choose the Outputs tab.
7. In the action, in AutoDiscoverReports section, or in the Reports section, add a SuccessCriteria property, along with PassRate, LineCoverage, BranchCoverage, Vulnerabilities, StaticAnalysisBug, StaticAnalysisSecurity, and StaticAnalysisQuality properties.

For an explanation of each of these properties, consult the Build and test action reference (p. 364).

8. Choose Commit.
9. Run your workflow to have CodeCatalyst apply success criteria to your raw reports, and regenerate the associated CodeCatalyst reports with the success criteria information included. For more information on starting a workflow, see Starting a workflow run (p. 314).

Example: Configuring reports

The following example shows how to manually configure four reports: a test report, a code coverage report, a software composition analysis report, and a static analysis report.

```
Reports:
  MyTestReport:
    Format: JUNITXML
    IncludePaths:
      - "*.xml"
    ExcludePaths:
      - report1.xml
    SuccessCriteria:
      PassRate: 90
  MyCoverageReport:
    Format: CLOVERXML
    IncludePaths:
      - output/coverage/jest/clover.xml
    SuccessCriteria:
      LineCoverage: 75
      BranchCoverage: 75
  MySCARReport:
    Format: SARIFSCA
    IncludePaths:
      - output/sca/reports.xml
    SuccessCriteria:
      Vulnerabilities:
        Number: 5
        Severity: HIGH
  MySAReport:
    Format: ESLINTJSON
    IncludePaths:
      - output/static/eslint.xml
```
SARIF support for Software Composition Analysis and Static Analysis reports

SARIF (Static Analysis Results Interchange Format) is an output file format which is available in Software Composition Analysis and Static Analysis reports in CodeCatalyst. The following example shows how to manually configure SARIF in a Static Analysis report:

```json
Reports:
  MySAReport:
    Format: SARIFSA
    IncludePaths:
      - output/sa_report.json
    SuccessCriteria:
      StaticAnalysisBug:
        Number: 25
        Severity: HIGH
      StaticAnalysisSecurity:
        Number: 5
        Severity: CRITICAL
      StaticAnalysisQuality:
        Number: 10
        Severity: INFORMATIONAL
```

CodeCatalyst supports the following SARIF properties which can be used to optimize how the analysis results will appear in your reports.

Topics

- `sarifLog object (p. 153)`
- `run object (p. 154)`
- `toolComponent object (p. 154)`
- `reportingDescriptor object (p. 154)`
- `result object (p. 156)`
- `location object (p. 157)`
- `physicalLocation object (p. 157)`
- `logicalLocation object (p. 158)`
- `fix object (p. 158)`

**sarifLog object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$schema</td>
<td>Yes</td>
<td>The URI of the SARIF JSON schema for version 2.1.0.</td>
</tr>
</tbody>
</table>
### Configuring reports

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Yes</td>
<td>CodeCatalyst only supports SARIF version 2.1.0.</td>
</tr>
<tr>
<td>runs[]</td>
<td>Yes</td>
<td>A SARIF file contains an array of one or more runs, each of which represents a single run of the analysis tool.</td>
</tr>
</tbody>
</table>

**run object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tool.driver</td>
<td>Yes</td>
<td>A toolComponent object that describes the analysis tool.</td>
</tr>
<tr>
<td>tool.name</td>
<td>No</td>
<td>A property that indicates the name of the tool used to perform analysis.</td>
</tr>
<tr>
<td>results[]</td>
<td>Yes</td>
<td>The results of the analysis tool that are displayed on CodeCatalyst.</td>
</tr>
</tbody>
</table>

**toolComponent object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Yes</td>
<td>The name of the analysis tool.</td>
</tr>
<tr>
<td>properties.artifactScanned</td>
<td>No</td>
<td>A total number of artifacts analyzed by the tool.</td>
</tr>
<tr>
<td>rules[]</td>
<td>Yes</td>
<td>An array of reportingDescriptor objects that represent rules. Based on these rules, the analysis tool finds problems in the code that is analyzed.</td>
</tr>
</tbody>
</table>

**reportingDescriptor object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>The unique identifier for the rule that is used to reference a finding. Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>name</td>
<td>No</td>
<td>The display name of the rule.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>shortDescription.text</td>
<td>No</td>
<td>A shortened description of the rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 3,000 characters</td>
</tr>
<tr>
<td>fullDescription.text</td>
<td>No</td>
<td>A complete description of the rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 3,000 characters</td>
</tr>
<tr>
<td>helpUri</td>
<td>No</td>
<td>A string that can be localized to contain the absolute URI of the primary documentation for the rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 3,000 characters</td>
</tr>
<tr>
<td>properties.unscore</td>
<td>No</td>
<td>A flag that indicates if the scan finding has been scored.</td>
</tr>
<tr>
<td>properties.score.severity</td>
<td>No</td>
<td>A fixed set of strings that specify the severity level of the finding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>properties.cvssv3_baseSeverity</td>
<td></td>
<td>A qualitative severity rating of Common Vulnerability Scoring System v3.1.</td>
</tr>
<tr>
<td>properties.cvssv3_baseScore</td>
<td>No</td>
<td>A CVSS v3 Base Score ranging from 0.0 - 10.0.</td>
</tr>
<tr>
<td>properties.cvssv2_severity</td>
<td>No</td>
<td>If CVSS v3 values are not available, CodeCatalyst searches for CVSS v2 values.</td>
</tr>
<tr>
<td>properties.cvssv2_score</td>
<td>No</td>
<td>A CVSS v2 Base Score ranging from 0.0 - 10.0.</td>
</tr>
<tr>
<td>properties.severity</td>
<td>No</td>
<td>A fixed set of strings that specify the severity level of the finding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>defaultConfiguration.level</td>
<td>No</td>
<td>The default severity of a rule.</td>
</tr>
</tbody>
</table>
## result object

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleId</td>
<td>Yes</td>
<td>The unique identifier for the rule that is used to reference a finding. Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>ruleIndex</td>
<td>Yes</td>
<td>The index of the associated rule in the tool component rules[].</td>
</tr>
<tr>
<td>message.text</td>
<td>Yes</td>
<td>A message that describes the result and displays the message for each finding. Maximum length: 3,000 characters</td>
</tr>
<tr>
<td>rank</td>
<td>No</td>
<td>A value between 0.0 to 100.0 inclusive that represents the priority or importance of the result. This scale values 0.0 being the lowest priority and 100.0 being the highest priority.</td>
</tr>
<tr>
<td>level</td>
<td>No</td>
<td>The severity of the result. Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>properties.unscore</td>
<td>No</td>
<td>A flag that indicates if the scan finding has been scored.</td>
</tr>
<tr>
<td>properties.score.severity</td>
<td>No</td>
<td>A fixed set of strings that specify the severity level of the finding. Maximum length: 1,024 characters</td>
</tr>
<tr>
<td>properties.cvssv3_baseSeverity</td>
<td>No</td>
<td>A qualitative severity rating of the finding <a href="https://cve.mitre.org/software-scoring/">Common Vulnerability Scoring System v3.1</a>.</td>
</tr>
<tr>
<td>properties.cvssv3_baseScore</td>
<td>No</td>
<td>A CVSS v3 Base Score ranging from <a href="https://cve.mitre.org/software-scoring/">0.0 - 10.0</a>.</td>
</tr>
<tr>
<td>properties.cvssv2_severity</td>
<td>No</td>
<td>If CVSS v3 values are not available, CodeCatalyst searches for CVSS v2 values.</td>
</tr>
<tr>
<td>properties.cvssv2_score</td>
<td>No</td>
<td>A CVSS v2 Base Score ranging from <a href="https://cve.mitre.org/software-scoring/">0.0 - 10.0</a>.</td>
</tr>
<tr>
<td>properties.severity</td>
<td>No</td>
<td>A fixed set of strings that specify the severity level of the finding.</td>
</tr>
</tbody>
</table>
### Configuring reports

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>locations[]</td>
<td>Yes</td>
<td>The set of locations where the result was detected. Only one location should be included unless the problem can only be corrected by making a change at every specified location. CodeCatalyst uses the first value in the location array to annotate the result. Maximum number of location objects: 10</td>
</tr>
<tr>
<td>relatedLocations[]</td>
<td>No</td>
<td>A list of additional locations references in the finding. Maximum number of location objects: 50</td>
</tr>
<tr>
<td>fixes[]</td>
<td>No</td>
<td>An array of fix objects that represent the recommendation provided by the scanning tool. CodeCatalyst uses the first recommendation in the fixes array.</td>
</tr>
</tbody>
</table>

#### location object

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>physicalLocation</td>
<td>Yes</td>
<td>Identifies the artifact and region.</td>
</tr>
<tr>
<td>logicalLocations[]</td>
<td>No</td>
<td>The set of locations described by name without reference to the artifact.</td>
</tr>
</tbody>
</table>

#### physicalLocation object

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>artifactLocation.uri</td>
<td>Yes</td>
<td>The URI indicating the location of an artifact, usually a file either in the repository or generated during a build.</td>
</tr>
<tr>
<td>fileLocation.uri</td>
<td>No</td>
<td>The fall back URI indicating the location of the file. This is used if artifactLocation.uri returns empty.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>region.startLine</td>
<td>Yes</td>
<td>The line number of the first character in the region.</td>
</tr>
<tr>
<td>region.startColumn</td>
<td>Yes</td>
<td>The column number of the first character in the region.</td>
</tr>
<tr>
<td>region.endLine</td>
<td>Yes</td>
<td>The line number of the last character in the region.</td>
</tr>
<tr>
<td>region.endColumn</td>
<td>Yes</td>
<td>The column number of the last character in the region.</td>
</tr>
</tbody>
</table>

**logicalLocation object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullyQualified Name</td>
<td>No</td>
<td>Additional information that describes the location of the result.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 1,024 characters</td>
</tr>
</tbody>
</table>

**fix object**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description.text</td>
<td>No</td>
<td>A message that displays a recommendation for each finding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum length: 3,000 characters</td>
</tr>
<tr>
<td>artifactChanges.</td>
<td>No</td>
<td>The URI indicating the location of the artifact that needs to be updated.</td>
</tr>
<tr>
<td>[0].artifactLocation.uri</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Using universal-test-runner**

Test actions integrate with the open-source command line tool `universal-test-runner`. This tool provides advanced testing features, such as retrying one or more test cases from a test report.

`universal-test-runner` uses the [Test Execution Protocol](https://docs.aws.amazon.com/codeguru/latest/using-codecatalyst/codecatalyst-user-guide.html) to run your tests for any language in a given framework. `universal-test-runner` supports the following frameworks:

- Gradle
- Jest
- Maven
- pytest
- .NET
universal-test-runner is installed only on the curated images for test actions. If you configure a test action to use a custom Docker Hub or Amazon ECR, you must manually install universal-test-runner to enable advanced testing features. To do so, install Node.js (14 or higher) on the image, then install universal-test-runner through npm using the shell command: `run: npm install -g @aws/universal-test-runner`. For more information about installing Node.js in your container through shell commands, see Installing and Updating Node Version Manager.

For more information about universal-test-runner, see What is universal-test-runner?

Visual

**To use universal-test-runner in the visual editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose CI/CD, and then choose Workflows.
3. Choose the name of your workflow.
4. Choose Edit.
6. Choose Actions.
7. In Actions, choose Test.
8. On the Configuration tab, complete the Shell commands field by updating the sample code with your choice of the supported frameworks. For example, to use a supported framework, you would use a Run command similar to the following:

   ```bash
   - Run: run-tests <framework>
   ```

   If the framework you want is not supported, consider contributing a custom adapter or runner. For a description of the Shell commands field, see Steps (p. 379).

9. (Optional) Choose Validate to validate the workflow's YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

YAML

**To use universal-test-runner in the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose CI/CD, and then choose Workflows.
3. Choose the name of your workflow.
4. Choose Edit.
5. Choose YAML.
6. Choose Actions.
7. In Actions, choose Test.
8. Modify the YAML code according to your needs. For example, to use a supported framework, you would use a Run command similar to the following.

   ```yaml
   Configuration:
   Steps:
   - Run: run-tests <framework>
   ```

   If the framework you want is not supported, consider contributing a custom adapter or runner. For a description of the Steps property, see Steps (p. 379).
9. (Optional) Choose Validate to validate the workflow's YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

Best practices

When using the testing features provided by CodeCatalyst, we recommend that you follow these best practices.

Topics

• Auto-discovery (p. 160)
• Success criteria (p. 160)
• Include/exclude paths (p. 161)

Auto-discovery

When configuring actions in CodeCatalyst, auto-discovery lets you automatically discover outputs of various tools, such as JUnit test reports, and generate relevant CodeCatalyst reports from them. Auto-discovery helps ensure that reports continue to be generated even if names or paths to discovered outputs change. When new files are added, CodeCatalyst automatically discovers them and produces relevant reports. However, if you use auto-discovery, it is important to factor in some of the following aspects of this feature:

• When you activate auto-discovery in your action, all automatically discovered reports of the same type will share the same success criteria. For example, a shared criteria such as minimum pass rate would apply to all auto-discovered test reports. If you need different criteria for reports of the same type, you must explicitly configure each of these reports.

• Auto-discovery can also find reports that are produced by your dependencies and, if success criteria are configured, might fail the action on these reports. This issue can be addressed by updating the exclude path configuration.

• Auto-discovery is not guaranteed to produce the same list of reports every time, because it scans the action at runtime. In the case where you want a particular report to always be produced, you should configure reports explicitly. For example, if tests were to stop running as part of your build, the test framework would not produce any outputs and, as a result, no test report would be produced and the action might succeed. If you want the success of your action to depend on that particular test, then you must explicitly configure that report.

Tip

When getting started on a new or existing project, use auto-discovery for the entire project directory (include **/*). This invokes report generation across all files in your project, including those within subdirectories.

For more information, see Configuring reports (p. 149).

Success criteria

You can enforce quality thresholds on your reports by configuring success criteria. For example, if two code coverage reports were auto-discovered, one with a line coverage of 80% and the other with a line coverage of 60%, you have the following options:

• Set the auto-discovery success criteria for line coverage at 80%. This would cause the first report to pass and the second report to fail, which would result in the overall action failing. To unblock the workflow, add new tests to your project until the line coverage for the second report exceeds 80%.
• Set the auto-discovery success criteria for line coverage at 60%. This would cause both reports to pass, which would result in the action succeeding. You could then work on increasing the code coverage in the second report. However, with this approach, you cannot guarantee that the coverage in the first report is not dropping below 80%.

• Explicitly configure one or both of the reports by using the visual editor or adding an explicit YAML section and path for each report. This would allow you to configure separate success criteria and custom names for each report. However, with this approach, the action could fail if the report paths change.

For more information, see Configuring success criteria for reports (p. 149).

Include/exclude paths

When reviewing action results, you can adjust the list of reports that are generated by CodeCatalyst by configuring IncludePaths and ExcludePaths.

• Use IncludePaths to specify the files and file paths you want CodeCatalyst to include when searching for reports. For example, if you specify "/test/report/*", CodeCatalyst searches the entire build image used by the action looking for the /test/report/ directory. When it finds that directory, CodeCatalyst then looks for reports in that directory.

  Note
  For manually configured reports, IncludePaths must be a glob pattern that matches a single file.

• Use ExcludePaths to specify the files and file paths you want CodeCatalyst to exclude when searching for reports. For example, if you specify "/test/reports/**/*", CodeCatalyst will not search for files in the /test/reports/ directory. To ignore all files in a directory, use the **/* glob pattern.

The following are examples of possible glob patterns.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>.</em></td>
<td>Matches all object names in the current directory that contain a dot</td>
</tr>
<tr>
<td>*.xml</td>
<td>Matches all object names in the current directory ending with .xml</td>
</tr>
<tr>
<td>*.{xml,txt}</td>
<td>Matches all object names in the current directory ending with .xml or .txt</td>
</tr>
<tr>
<td>**/<em>.</em>.xml</td>
<td>Matches object names across all directories ending with .xml</td>
</tr>
<tr>
<td>testFolder</td>
<td>Matches an object called testFolder, treating it as a file</td>
</tr>
<tr>
<td>testFolder/*</td>
<td>Matches objects in one level of the subfolder from testFolder, such as testFolder/file.xml</td>
</tr>
<tr>
<td>testFolder/**</td>
<td>Matches objects in two levels of the subfolder from testFolder, such as testFolder/reportsFolder/file.xml</td>
</tr>
<tr>
<td>testFolder/**</td>
<td>Matches subfolder testFolder as well as files below testFolder, such as testFolder/</td>
</tr>
</tbody>
</table>
Patterns and Description

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file.xml and testFolder/otherFolder/file.xml</td>
<td></td>
</tr>
</tbody>
</table>

CodeCatalyst interprets the glob patterns as follows:

- The slash (/) character separates directories in file paths.
- The asterisk (*) character matches zero or more characters of a name component without crossing folder boundaries.
- A double asterisk (**) matches zero or more characters of a name component across all directories.

**Note**
ExcludePaths takes precedence over IncludePaths. If both IncludePaths and ExcludePaths include the same folder, that folder is not scanned for reports.

**Working with tests**

This section describes how to view and configure your test actions.

**Topics**
- Viewing the results of a test action (p. 162)
- Skipping failed tests (p. 162)
- Retrying test cases (p. 163)

**Viewing the results of a test action**

Use the following instructions to view the results of a test action, including the generated logs, reports, and variables.

**To view the results of a test action**

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
3. In the workflow diagram, choose the name of your test action (for example, Test).
4. To view the logs generated by an action, choose Logs. The logs for the various action phases are displayed. You can expand or collapse the logs as needed.
5. To view the test reports produced by the test action, choose Reports. For more information, see Test report types (p. 147).
6. To view the configuration used for the test action, choose Configuration. For more information, see Adding the test action (p. 149).
7. To view the variables used by the test action, choose Variables. For more information, see Working with variables (p. 337).

**Skipping failed tests**

If your action has more than one test command, you might want to allow subsequent test commands in the action to run even if a previous command fails. For example, in the following commands, you may want test2 to run always, even if test1 fails.
Steps:
- Run: npm install
- Run: npm run test1
- Run: npm run test2

Normally, when a step returns an error, Amazon CodeCatalyst stops the workflow action and marks it as failed. You can allow the action steps to continue to run by redirecting the error output to null. You can do this by adding `2>/dev/null` to the command. With this modification, the preceding example would look like the following.

Steps:
- Run: npm install
- Run: npm run test1 2>/dev/null
- Run: npm run test2

In the second code snippet, the status of the `npm install` command will be honored, but any error returned by the `npm run test1` command will be ignored. As a result the `npm run test2` command is run. By doing this, you're able to view both reports at once regardless of whether an error occurs.

**Retrying test cases**

If your report fails because of several test cases, you can retry only those individual tests. This allows you to quickly check the quality of your test cases and determine the next steps to resolve your issues, like engaging a broken dependency, or initiating a workflow rerun. Your test action incorporates the `universal-test-runner` to retry only selected test cases, instead of the whole action. You can only retry one set of selected test cases per action at a time, and only retry five times per test report. For more information, see [Using universal-test-runner](p. 158).

**Note**
If you retry test cases on a report, it will have no effect on the status of the workflow that generated the original report.

Use the following instructions to retry the test cases in your reports.

**To retry the test cases of a report**

1. In the navigation pane, choose Reports.
2. Choose the name of your report. You can filter by the name, status, repository, branch, or type of report.
3. Under the name of the report, choose Results.
4. Select the test cases that you want to retry, choose Rerun, then choose Selected test cases.
5. When your retry is finished, choose Refresh on the banner and view the updated results.

### Deploying using workflows in CodeCatalyst

Using [CodeCatalyst workflows](p. 128), you can deploy applications and other resources to various targets such as Amazon ECS, AWS Lambda, and more.

**Topics**
- [How do I deploy an application?](p. 164)
- [List of deploy actions](p. 164)
- [Benefits of deploy actions](p. 165)
How do I deploy an application?

To deploy an application or resource through CodeCatalyst, you first create a workflow, and then specify a deploy action inside of it. A **deploy action** is a workflow building block that defines what you want to deploy, where you want to deploy it, and how you want to deploy it (for example, using a blue/green scheme). You add a deploy action to your workflow using the CodeCatalyst console's visual editor, or YAML editor.

The high-level steps to deploy an application or resource are as follows.

**To deploy an application (high-level tasks)**

1. In your CodeCatalyst project, you **add source code** for an application you want to deploy. For more information, see [Working with source repositories in CodeCatalyst](p. 89).
2. In your CodeCatalyst project, you **create a workflow**. The workflow is where you define how to build, test, and deploy your application. For more information, see [Getting started with workflows in CodeCatalyst](p. 134).
3. In the workflow, you **add a trigger**, a **build action**, and optionally, a **test action**. For more information, see [Working with triggers](p. 327), [Adding the build action](p. 139), and [Adding the test action](p. 149).
4. In the workflow, you **add a deploy action**. You can choose from several CodeCatalyst-provided deploy actions to your application to different targets, such as Amazon ECS. (You can also use a build action or a GitHub Action to deploy your application. For more information about the build action and GitHub Actions, see [Alternatives to deploy actions](p. 165).)
5. You **start the workflow** either manually or automatically through a trigger. The workflow runs the build, test, and deploy actions in sequence to deploy your application and resources to the target. For more information, see [Starting a workflow run](p. 314).

**List of deploy actions**

The following deploy actions are available:

- Deploy AWS CloudFormation stack – This action creates a CloudFormation stack in AWS based on an [AWS CloudFormation template](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/) or [AWS Serverless Application Model template](https://docs.aws.amazon.com/serverless-application-model/latest/developer-guide/) that you provide. For more information, see [Adding the “Deploy AWS CloudFormation stack” action](p. 228).
- Deploy to Amazon ECS – This action registers a [task definition](https://docs.aws.amazon.com/ecs/latest/userguide/task-definition.html) file that you provide. For more information, see [Adding the “Deploy to Amazon ECS” action](p. 230).
- Deploy to Kubernetes cluster – This action deploys an application to an Amazon Elastic Kubernetes Service cluster. For more information, see [Adding the “Deploy to Kubernetes cluster” action](p. 231).
- AWS CDK deploy – This action deploys an [AWS CDK app](https://github.com/aws-architects/aws-cdk) into AWS. For more information, see [Adding the "AWS CDK deploy" action](p. 234).
**Note**

There are other CodeCatalyst actions that can deploy resources; however, they are not considered deploy actions because their deployment information doesn’t appear on the **Environments** page. To learn more about the **Environments** page and viewing deployments, see Working with environments (p. 307) and Working with deployments (p. 238).

**Benefits of deploy actions**

Using deploy actions within a workflow has the following benefits:

- **Deployment history** – View a history of your deployments to help manage and communicate changes in your deployed software.
- **Traceability** – Track the status of your deployments through the CodeCatalyst console, and see when and where each application revision was deployed.
- **Rollbacks** – Roll back deployments automatically if there are errors. You can also configure alarms to activate deployment rollbacks.
- **Monitoring** – Watch your deployment as it progresses through the various stages of your workflow.
- **Integration with other CodeCatalyst features** – Store source code and then build, test, and deploy it, all from one application.

**Alternatives to deploy actions**

You don't have to use deploy actions, although they are recommended because they offer the benefits outlined in the preceding section. Instead, you can use the following CodeCatalyst actions (p. 249):

- A **build** action.
  
  Typically, you use build actions if you want to deploy to a target for which a corresponding deploy action does not exist, or if you want more control over the deployment procedure. For more information about using build actions to deploy resources, see Building using workflows in CodeCatalyst (p. 138).

- A **GitHub Action**.
  
  You can use a GitHub Action (p. 249) inside a CodeCatalyst workflow to deploy applications and resources (instead of a CodeCatalyst action). For information about how to use GitHub Actions inside a CodeCatalyst workflow, see Adding GitHub Actions (p. 266)

You can also use the following AWS services to deploy your application, if you don't want to use a CodeCatalyst workflow to do so:

- **AWS CodeDeploy** – see What is CodeDeploy?
- **AWS CodeBuild and AWS CodePipeline** – see What is AWS CodeBuild? and What is AWS CodePipeline?
- **AWS CloudFormation** – see What is AWS CloudFormation?

Use CodeDeploy, CodeBuild, CodePipeline, and CloudFormation services for complex, enterprise deployments.

**Tutorial: Deploy a serverless application using AWS CloudFormation**

In this tutorial, you learn how to build, test, and deploy a serverless application using a workflow and AWS CloudFormation.
The application in this tutorial is a simple web application that outputs a "Hello World" message. It consists of an AWS Lambda function and an Amazon API Gateway, and you build it using the AWS Serverless Application Model (AWS SAM), which is an extension of AWS CloudFormation.

Topics
- Prerequisites (p. 166)
- Step 1: Create a source repository (p. 143)
- Step 2: Create AWS roles (p. 167)
- Step 3: Add AWS roles to CodeCatalyst (p. 172)
- Step 4: Create an Amazon S3 bucket (p. 172)
- Step 5: Add source files (p. 173)
- Step 6: Create and run a workflow (p. 178)
- Step 7: Make a change (p. 182)
- Clean up (p. 146)

Prerequisites

Before you begin:

- You need a CodeCatalyst space with a connected AWS account. For more information, see Creating additional spaces in CodeCatalyst (p. 53).
- In your space, you need an empty, Start from scratch CodeCatalyst project called:

  codecatalyst-cfn-project

  For more information, see Creating an empty project in Amazon CodeCatalyst (p. 58).
- In your project, you need a CodeCatalyst environment called:

  codecatalyst-cfn-environment

  Configure this environment as follows:
  - Choose any type, such as Non-production.
  - Connect your AWS account to it.

  For more information, see Working with environments (p. 307).

Step 1: Create a source repository

In this step, you create a source repository in CodeCatalyst. This repository is used to store the tutorial's source files, such as the Lambda function file.

For more information about source repositories, see Creating a source repository (p. 89).

To create a source repository

1. In CodeCatalyst, in the navigation pane, choose Code, and then choose Source repositories.
2. Choose Add repository, and then choose Create repository.
3. In Repository name, enter:

   codecatalyst-cfn-source-repository
4. Choose Create.

You have now created a repository called codecatalyst-cfn-source-repository.

Step 2: Create AWS roles

In this step, you create the following AWS IAM roles:

- **Deploy role** – Grants the CodeCatalyst **Deploy AWS CloudFormation stack** action permission to access your AWS account and CloudFormation service where you'll deploy your serverless application. The **Deploy AWS CloudFormation stack** action is part of your workflow.
- **Build role** – Grants the CodeCatalyst build action permission to access your AWS account and write to Amazon S3 where your serverless application package will be stored. The build action is part of your workflow.
- **Stack role** – Grants CloudFormation permission to read and modify the resources specified in the AWS SAM template that you will provide later. Also grants permission to CloudWatch.

For more information about IAM roles, see IAM roles in the AWS Identity and Access Management User Guide.

**Note**

To save time, you can create a single role, called the CodeCatalystWorkflowDevelopmentRole-spaceName role, instead of the three roles listed previously. For more information, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions that may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern. This tutorial assumes you are creating the three roles listed previously.

**Note**

A Lambda execution role is also required, but you don't need to create it now because the sam-template.yml file creates it for you when you run the workflow in step 5.

To create a deploy role

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   b. Open the IAM console at https://console.aws.amazon.com/iam/.
   c. In the navigation pane, choose Policies.
   d. Choose Create policy.
   e. Choose the JSON tab.
   f. Delete the existing code.
   g. Paste the following code:

```json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Action": [
      "cloudformation:CreateStack",
      "cloudformation:DeleteStack",
      "cloudformation:Describe*"
    ],
```

167
Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

h. Choose Next: Tags.
i. Choose Next: Review.
j. In Name, enter:

```
codecatalyst-deploy-policy
```
k. Choose Create policy.

You have now created a permissions policy.

2. Create the deploy role, as follows:

a. In the navigation pane, choose Roles, and then choose Create role.
b. Choose Custom trust policy.
c. Delete the existing custom trust policy.
d. Add the following custom trust policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "codecatalyst-runner.amazonaws.com",
               "codecatalyst.amazonaws.com"
            ],
            "Action": "sts:AssumeRole"
         }
      }
   ]
}
```
e. Choose Next.
f. In Permissions policies, search for codecatalyst-deploy-policy and select its check box.
g. Choose Next.
h. For Role name, enter:
i. For **Role description**, enter:

```
Codecatalyst deploy role
```

j. Choose **Create role**.

You have now created a deploy role with a trust policy and permissions policy.

3. Obtain the deploy role ARN, as follows:

   a. In the navigation pane, choose **Roles**.
   b. In the search box, enter the name of the role you just created (**codecatalyst-deploy-role**).
   c. Choose the role from the list.

   The role's **Summary** page appears.
   d. At the top, copy the **ARN** value.

You have now created the deploy role with the appropriate permissions, and obtained its ARN.

**To create a build role**

1. Create a policy for the role, as follows:

   a. Sign in to AWS.
   c. In the navigation pane, choose **Policies**.
   d. Choose **Create policy**.
   e. Choose the **JSON** tab.
   f. Delete the existing code.
   g. Paste the following code:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Action": [
            "s3:PutObject",
            "iam:PassRole"
        ],
        "Resource": "*",
        "Effect": "Allow"
    }]
}
```

**Note**

The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

h. Choose **Next: Tags**.
2. Create the build role, as follows:

   a. In the navigation pane, choose Roles, and then choose Create role.
   b. Choose Custom trust policy.
   c. Delete the existing custom trust policy.
   d. Add the following custom trust policy:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": [
             "codecatalyst-runner.amazonaws.com",
             "codecatalyst.amazonaws.com"
           ],
           "Action": "sts:AssumeRole"
         }
       }
     ]
   }
   ```
   e. Choose Next.
   f. In Permissions policies, search for codecataclyst-build-policy and select its check box.
   g. Choose Next.
   h. For Role name, enter:

   ```plaintext
   codecatalyst-build-role
   ```
   i. For Role description, enter:

   ```plaintext
   CodeCatalyst build role
   ```
   j. Choose Create role.

   You have now created a build role with a trust policy and permissions policy.

3. Obtain the build role ARN, as follows:

   a. In the navigation pane, choose Roles.
   b. In the search box, enter the name of the role you just created (codecatalyst-build-role).
   c. Choose the role from the list.

   The role's Summary page appears.
   d. At the top, copy the ARN value.
You have now created the build role with the appropriate permissions, and obtained its ARN.

**To create a stack role**

1. Sign in to AWS using the account where you want to deploy your stack.
3. Create the stack role as follows:
   a. In the navigation pane, choose **Roles**.
   b. Choose **Create role**.
   c. Choose **AWS service**.
   d. In the **Use case** section, choose **CloudFormation** from the drop-down list.
   e. Select the **CloudFormation** radio button.
   f. At the bottom, choose **Next**.
   g. Using the search box, find the following permissions policies, and then select their respective check boxes.
      
      **Note**
      If you search for a policy and it doesn't appear, make sure to choose **Clear filters** and try again.

      - **CloudWatchFullAccess**
      - **AWSCloudFormationFullAccess**
      - **IAMFullAccess**
      - **AWSLambda_FullAccess**
      - **AmazonAPIGatewayAdministrator**
      - **AmazonS3FullAccess**
      - **AmazonEC2ContainerRegistryFullAccess**

      The first policy allows access to CloudWatch to enable stack rollbacks when an alarm occurs.

      The remaining policies allow AWS SAM to access the services and resources in the stack that will be deployed in this tutorial. For more information, see **Permissions** in the **AWS Serverless Application Model Developer Guide**.

   h. Choose **Next**.
   i. For **Role name**, enter:


   **codcatalyst-stack-role**

   j. Choose **Create role**.
4. Obtain the stack role's ARN, as follows:
   a. In the navigation pane, choose **Roles**.
   b. In the search box, enter the name of the role you just created (**codcatalyst-stack-role**).
   c. Choose the role from the list.
   d. In the **Summary** section, copy the **ARN** value. You need it later.

You have now created the stack role with the appropriate permissions, and you have obtained its ARN.
Step 3: Add AWS roles to CodeCatalyst

In this step, you add the build role (codecatalyst-build-role) and deploy role (codecatalyst-deploy-role) to the CodeCatalyst account connection in your space.

Note
You don't need to add the stack role (codecatalyst-stack-role) to the connection. This is because the stack role is used by CloudFormation (not CodeCatalyst), after a connection is already established between CodeCatalyst and AWS using the deploy role. Since the stack role is not used by CodeCatalyst to gain access to AWS, it does not need to be associated with an account connection.

To add build and deploy roles to your account connection

1. In CodeCatalyst, navigate to your space.
2. Choose AWS accounts. A list of account connections appears.
3. Choose the account connection that represents the AWS account where you created your build and deploy roles.
4. Choose Manage roles from AWS management console.
   
   The Add IAM role to Amazon CodeCatalyst space page appears. You might need to sign in to access the page.
5. Select Add an existing role you have created in IAM.
   
   A drop-down list appears. The list displays all IAM roles with a trust policy that includes the codecatalyst-runner.amazonaws.com and codecatalyst.amazonaws.com service principals.
6. In the drop-down list, choose codecatalyst-build-role, and choose Add role.
7. Choose Add IAM role, choose Add an existing role you have created in IAM, and in the drop-down list, choose codecatalyst-deploy-role. Choose Add role.

You have now added the build and deploy roles to your space.
8. Copy the value of the Amazon CodeCatalyst display name. You’ll need this value later, when creating your workflow.

Step 4: Create an Amazon S3 bucket

In this step, you create an Amazon S3 bucket where you store your serverless application's deployment package .zip file.

To create an Amazon S3 bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the main pane, choose Create bucket.
3. For Bucket name, enter:

   codecatalyst-cfn-s3-bucket

4. For AWS Region, choose a Region. This tutorial assumes you chose US West (Oregon) us-west-2. For information about Regions supported by Amazon S3, see Amazon Simple Storage Service endpoints and quotas in the AWS General Reference.
5. At the bottom of the page, choose Create bucket.
You have now created a bucket called `codecatalyst-cfn-s3-bucket` in the US West (Oregon) us-west-2 Region.

**Step 5: Add source files**

In this step, you add several application source files to your CodeCatalyst source repository. The `hello-world` folder contains the application files that you'll deploy. The `tests` folder contains unit tests. The folder structure is as follows:

```
- hello-world
  |-- tests
  |    |-- unit
  |    |    |-- test-handler.js
  |    |-- app.js
  |-- .npmignore
  |-- package.json
  |-- sam-template.yml
  |-- setup-sam.sh
```

**.npmignore file**

The `.npmignore` file indicates which files and folders npm should exclude from the application package. In this tutorial, npm excludes the `tests` folder because it is not part of the application.

**To add the `.npmignore` file**

2. Choose your project, `codecatalyst-cfn-project`.
3. In the navigation pane, choose `Code`, and then choose `Source repositories`.
4. From the list of source repositories, choose your repository, `codecatalyst-cfn-source-repository`.
5. In `Files`, choose `Create file`.
6. For `File name`, enter: `.npmignore`.
7. In the text box, enter the following code:

```
tests/*
```
8. Choose `Commit`, and then choose `Commit` again.

You have now created a file called `.npmignore` in the root of your repository.

**package.json file**

The `package.json` file contains important metadata about your Node project such as the project name, version number, description, dependencies, and other details that describe how to interact with and run your application.

The `package.json` in this tutorial includes a list of dependencies and a `test` script. The test script does the following:

- Using `mocha`, the test script runs the unit tests specified in `hello-world/tests/unit/` and writes the results to a `junit.xml` file using the `xunit` reporter.
To add the package.json file

1. In your repository, in Files, choose Create file.
2. For File name, enter:

   package.json

3. In the text box, enter the following code:

   ```json
   {
   "name": "hello_world",
   "version": "1.0.0",
   "description": "hello world sample for NodeJS",
   "main": "app.js",
   "author": "SAM CLI",
   "license": "MIT",
   "dependencies": {
   "axios": "^0.21.1",
   "nyc": "^15.1.0"
   },
   "scripts": {
   "test": "nyc --reporter=clover mocha hello-world/tests/unit/ --reporter xunit --reporter-option output=junit.xml"
   },
   "devDependencies": {
   "aws-sdk": "^2.815.0",
   "chai": "^4.2.0",
   "mocha": "^8.2.1"
   }
   }
   ```

4. Choose Commit, and then choose Commit again.

   You have now added a file called package.json to the root of the repository.

sam-template.yml file

The sam-template.yml file contains the instructions for deploying the Lambda function and API Gateway and configuring them together. It follows the AWS Serverless Application Model template specification, which extends the AWS CloudFormation template specification.

You use an AWS SAM template in this tutorial instead of a regular AWS CloudFormation template because AWS SAM offers a helpful AWS::Serverless::Function resource type. This type performs much behind-the-scenes configuration that you normally have to write out to use the basic CloudFormation syntax. For example, the AWS::Serverless::Function creates a Lambda function, Lambda execution role, and event source mappings that start the function. You have to code all of this if you want to write it using basic CloudFormation.

Although this tutorial uses a pre-written template, you can generate one as part of your workflow using a build action. For more information, see Adding the "Deploy AWS CloudFormation stack" action (p. 228).
To add the `sam-template.yml` file

1. In your repository, in Files, choose **Create file**.
2. For **File name**, enter:

   `sam-template.yml`

3. In the text box, enter the following code:

   ```
   AWSTemplateFormatVersion: '2010-09-09'
   Transform: AWS::Serverless-2016-10-31
   Description: >
     serverless-api
     Sample SAM Template for serverless-api

   # More info about Globals: https://github.com/awslabs/serverless-application-model/blob/master/docs/globals.rst
   Globals:
     Function:
       Timeout: 3

   Resources:
   HelloWorldFunction:
     Type: AWS::Serverless::Function # For details on this resource type, see https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md#awsserverlessfunction
     Properties:
       CodeUri: hello-world/
       Handler: app.lambdaHandler
       Runtime: nodejs12.x
       Events:
         HelloWorld:
           Type: Api # For details on this event source type, see https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md#api
           Properties:
             Path: /hello
             Method: get

   Outputs:
   # ServerlessRestApi is an implicit API created out of the events key under Serverless::Function
   # Find out about other implicit resources you can reference within AWS SAM at
   # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/generated_resources.rst#api
   HelloWorldApi:
     Description: "API Gateway endpoint URL for the Hello World function"
     Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/hello/"
   HelloWorldFunction:
     Description: "Hello World Lambda function ARN"
     Value: !GetAtt HelloWorldFunction.Arn
   HelloWorldFunctionIamRole:
     Description: "Implicit Lambda execution role created for the Hello World function"
     Value: !GetAtt HelloWorldFunctionRole.Arn
   ```

4. Choose **Commit**, and then choose **Commit again**.

   You have now added a file called `sam-template.yml` under the root folder of your repository.
**setup-sam.sh file**

The setup-sam.sh file contains the instructions for downloading and installing the AWS SAM CLI utility. The workflow uses this utility to package the hello-world source.

**To add the setup-sam.sh file**

1. In your repository, in Files, choose Create file.
2. For File name, enter:

```bash
test
```

3. In the text box, enter the following code:

```bash
#!/usr/bin/env bash
echo "Setting up sam"
yum install unzip -y
unzip -qq aws-sam-cli-linux-x86_64.zip -d sam-installation-directory
./sam-installation-directory/install; export AWS_DEFAULT_REGION=us-west-2
```

In the preceding code, replace `us-west-2` with your AWS Region.

4. Choose Commit, and then choose Commit again.

You have now added a file called setup-sam.sh to the root of the repository.

**app.js file**

The app.js contains the Lambda function code. In this tutorial, the code returns the text `hello world`.

**To add the app.js file**

1. In your repository, in Files, choose Create file.
2. For File name, enter:

```javascript
hello-world/app.js
```

3. In the text box, enter the following code:

```javascript
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;

/**
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 * @param {Object} context
 * @param {Object} event
 * @return {Object} context
 */
```

176
exports.lambdaHandler = async (event, context) => {
    try {
        // const ret = await axios(url);
        response = {
            'statusCode': 200,
            'body': JSON.stringify({
                message: 'hello world',
                // location: ret.data.trim()
            })
        }
    } catch (err) {
        console.log(err);
        return err;
    }
    return response;
};

4. Choose Commit, and then choose Commit again.

You have now created a folder called hello-world and a file called app.js.

test-handler.js file

The test-handler.js file contains unit tests for the Lambda function.

To add the test-handler.js file

1. In your repository, in Files, choose Create file.
2. For File name, enter:

   hello-world/tests/unit/test-handler.js

3. In the text box, enter the following code:

   'use strict';
   const app = require('../..../app.js');
   const chai = require('chai');
   const expect = chai.expect;
   var event, context;

   describe('Tests index', function () {
       it('verifies successful response', async () => {
           const result = await app.lambdaHandler(event, context)
           expect(result).to.be.an('object');
           expect(result.statusCode).to.equal(200);
           expect(result.body).to.be.an('string');
           let response = JSON.parse(result.body);
           expect(response).to.be.an('object');
           expect(response.message).to.be.equal('hello world');
           // expect(response.location).to.be.an('string');
       });
   });

4. Choose Commit, and then choose Commit again.
You have now added a file called test-handler.js under the hello-world/tests/unit folder.

You have now added all your source files.

Take a moment to double-check your work and make sure you placed all the files in the correct folders. The folder structure is as follows:

```
|— hello-world
 | |— tests
 | | |— unit
 | | | |— test-handler.js
 | | |— app.js
 | |— npmignore
 |— README.md
 |— package.json
 |— sam-template.yml
 |— setup-sam.sh
```

### Step 6: Create and run a workflow

In this step, you create a workflow that packages your Lambda source code and deploys it. The workflow consists of the following building blocks that run sequentially:

- **A trigger** – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see [Working with triggers](p. 327).

- **A test action (Test)** – On trigger, this action installs Node package manager (npm), and then runs the npm run test command. This command tells npm to run the test script defined in the package.json file. The test script, in turn, runs the unit tests and generates two reports: a test report (junit.xml) and a code coverage report (clover.xml). For more information, see [package.json file](p. 173).

Next, the test action transforms the XML reports into CodeCatalyst reports and displays them in the CodeCatalyst console, under the **Reports** tab of the test action.

For more information about the test action, see [Testing using workflows in CodeCatalyst](p. 146).

- **A build action (BuildBackend)** – On completion of the test action, the build action downloads and installs the AWS SAM CLI, packages the hello-world source, and copies the package to your Amazon S3 bucket, where the Lambda service expects it to be. The action also outputs a new AWS SAM template file called sam-template-packaged.yml and places it in an output artifact called buildArtifact.

For more information about the build action, see [Building using workflows in CodeCatalyst](p. 138).

- **A deploy action (DeployCloudFormationStack)** – On completion of the build action, the deploy action looks for the output artifact generated by the build action (buildArtifact), finds the AWS SAM template inside of it, and then runs the template. The AWS SAM template creates a stack that deploys the serverless application.

To create a workflow

1. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
2. Choose **Create workflow**.
3. For **Source repository**, choose codecatalyst-cfn-source-repository.
4. For **Branch**, choose main.
5. Choose **Create**.
6. Delete the YAML sample code.
7. Add the following YAML code:

```yaml
Name: codecatalyst-cfn-workflow
SchemaVersion: 1.0

Triggers:
- Type: PUSH
  Branches:
    - main

Actions:
  Test:
    Identifier: aws/managed-test@v1
    Inputs:
      Sources:
        - WorkflowSource
    Outputs:
      Reports:
        CoverageReport:
          Format: CLOVERXML
          IncludePaths:
            - "coverage/**"
        TestReport:
          Format: JUNITXML
          IncludePaths:
            - junit.xml

Configuration:
  Steps:
    - Run: npm install
    - Run: npm run test

BuildBackend:
  Identifier: aws/build@v1
  DependsOn:
    - Test
  Environment:
    Name: codecatalyst-cfn-environment
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-build-role
  Inputs:
    Sources:
      - WorkflowSource
  Configuration:
    Steps:
      - Run: . ./setup-sam.sh
        - Run: sam package --template-file sam-template.yml --s3-bucket codecatalyst-cfn-s3-bucket --output-template-file sam-template-packaged.yml --region us-west-2
    Outputs:
      Artifacts:
        - Name: buildArtifact
          Files:
            - "**/*"

DeployCloudFormationStack:
  Identifier: aws/cfn-deploy@v1
  DependsOn:
    - BuildBackend
  Environment:
    Name: codecatalyst-cfn-environment
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-deploy-role
  Inputs:
    Artifacts:
```

179
- buildArtifact
  Sources: []
  Configuration:
  name: codecatalyst-cfn-stack
  region: us-west-2
  role-arn: arn:aws:iam::111122223333:role/StackRole
  template: ./sam-template-packaged.yml
  capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND

In the preceding code, replace:

- Both instances of `codecatalyst-cfn-environment` with the name of your environment.
- Both instances of `codecatalyst-account-connection` with the display name of your account connection. The display name might be a number. For more information, see [Step 3: Add AWS roles to CodeCatalyst](p. 172).
- `codecatalyst-build-role` with the name of the build role that you created in [Step 2: Create AWS roles](p. 167).
- `codecatalyst-cfn-s3-bucket` with the name of the Amazon S3 bucket you created in [Step 4: Create an Amazon S3 bucket](p. 172).
- Both instances of `us-west-2` with the Region where your Amazon S3 bucket resides (first instance) and where your stack will be deployed (second instance). These Regions can be different. This tutorial assumes that both Regions are set to `us-west-2`. For details about Regions supported by Amazon S3 and AWS CloudFormation, see [Service endpoints and quotas in the AWS General Reference](p. 167).
- `codecatalyst-deploy-role` with the name of the deploy role that you created in [Step 2: Create AWS roles](p. 167).
- `codecatalyst-cfn-environment` with the name of the environment that you created in [Prerequisites](p. 166).
- `arn:aws:iam::111122223333:role/StackRole` with the Amazon Resource Name (ARN) of the stack role that you created in [Step 2: Create AWS roles](p. 167).

**Note**
If you decided not to create build, deploy, and stack roles, replace `codecatalyst-build-role, codecatalyst-deploy-role, and arn:aws:iam::111122223333:role/StackRole` with the name or ARN of the CodeCatalystWorkflowDevelopmentRole-`spaceName` role. For more information about this role, see [Step 2: Create AWS roles](p. 167).

For information about the properties in the code shown previously, see the "Deploy AWS CloudFormation stack" action reference (p. 411).

8. (Optional) Choose **Validate** to make sure the YAML code is valid before committing.

9. Choose **Commit**.

10. On the **Commit workflow** dialog box, enter the following:

    a. For **Workflow file name**, keep the default, `codecatalyst-cfn-workflow`.
    b. For **Commit message**, enter:

```
add initial workflow file
```

c. For **Repository**, choose `codecatalyst-cfn-source-repository`.

d. For **Branch name**, choose `main`.

e. Choose **Commit**.
You have now created a workflow. A workflow run starts automatically because of the trigger defined at the top of the workflow. Specifically, when you committed (and pushed) the `codecatalyst-cfn-workflow.yaml` file to your source repository, the trigger started the workflow run.

**To view the workflow run in progress**

1. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
2. Choose the workflow you just created: `codecatalyst-cfn-workflow`.
3. Choose the **Runs** tab.
4. In the **Run ID** column, choose the run ID.
5. Choose **Test** to see the tests progress.
6. Choose **BuildBackend** to see the build progress.
7. Choose **DeployCloudFormationStack** to see the deployment progress.

   For more information about viewing run details, see [Viewing workflow run status and details](p. 315).

8. When the **DeployCloudFormationStack** action finishes, do the following:
   - If the workflow run succeeded, go to the next procedure.
   - If the workflow run failed on the **Test** or **BuildBackend** action, choose **Logs** to troubleshoot the issue.
   - If the workflow run failed on the **DeployCloudFormationStack** action, choose the deploy action, and then choose the **Summary** tab. Scroll to the **CloudFormation events** section to view the detailed error message. If a rollback occurred, delete the `codecatalyst-cfn-stack` stack through the AWS CloudFormation console in AWS before re-running the workflow.

**To verify the deployment**

1. After a successful deployment, choose **Variables (7)** from the horizontal menu bar near the top. (Do not choose **Variables** in the pane on the right.)
2. Next to **HelloWorldApi**, paste the `https://` URL into a browser.

   A **hello world** JSON message from the Lambda function is displayed, indicating that the workflow deployed and configured the Lambda function and API Gateway successfully.

   **Tip**
   
   You can have CodeCatalyst display this URL in the workflow diagram with a few small configurations. For more information, see [Surfacing the URL of the deployed application](p. 242).

**To verify unit test results and code coverage**

1. In the workflow diagram, choose **Test**, and then choose **Reports**.
2. Choose **TestReport** to view the unit test results, or choose **CoverageReport** to view the code coverage details of the files being tested, in this case, `app.js` and `test-handler.js`.

**To verify deployed resources**

1. Sign in to the AWS Management Console and open the API Gateway console at [https://console.aws.amazon.com/apigateway/](https://console.aws.amazon.com/apigateway/).
2. Observe the `codecatalyst-cfn-stack` API that the AWS SAM template created. The API name comes from the Configuration/name value in the workflow definition file (`codecatalyst-cfn-workflow.yaml`).


4. In the navigation pane, choose **Functions**.

5. Choose your Lambda function, `codecatalyst-cfn-stack-HelloWorldFunction-string`.

6. You can see how the API Gateway is a trigger for the function. This integration was automatically configured by the AWS SAM `AWS::Serverless::Function` resource type.

---

**Step 7: Make a change**

In this step, you make a change to your Lambda source code and commit it. This commit starts a new workflow run. This run deploys the new Lambda function in a blue-green scheme that uses the default traffic shifting configuration specified in the Lambda console.

**To make a change to your Lambda source**

1. In CodeCatalyst, navigate to your project.
2. In the navigation pane, choose **Code**, and then choose **Source repositories**.
3. Choose your source repository `codecatalyst-cfn-source-repository`.
4. Change the application file:
   a. Choose the `hello-world` folder.
   b. Choose the `app.js` file.
   c. Choose **Edit**.
   d. On line 23, change `hello world` to **Tutorial complete!**.
   e. Choose **Commit**, and then choose **Commit** again.

   The commit causes a workflow run to start. This run will fail because you haven't updated the unit tests to reflect the name change.

5. Update the unit tests:
   a. Choose `hello-world\tests\unit\test-handler.js`.
   b. Choose **Edit**.
   c. On line 19, change `hello world` to **Tutorial complete!**.
   d. Choose **Commit**, and then choose **Commit** again.

   The commit causes another workflow run to start. This run will succeed.

6. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
7. Choose `codecatalyst-cfn-workflow`, and then choose **Runs**.
8. Choose the run ID of the latest run. It should still be in progress.
9. Choose **Test**, **BuildBackend**, and **DeployCloudFormationStack** to see the workflow run progress.
10. When the workflow finishes, choose **Variables (7) near the top.**
11. Next to **HelloWorldApi**, paste the `https://` URL into a browser.

   A Tutorial complete! message appears in the browser, indicating that your new application was deployed successfully.

---

**Clean up**

Clean up the files and services used in this tutorial to avoid being charged for them.
To clean up in the CodeCatalyst console

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Delete codecatalyst-cfn-workflow.
3. Delete codecatalyst-cfn-environment.
4. Delete codecatalyst-cfn-source-repository.
5. Delete codecatalyst-cfn-project.

To clean up in the AWS Management Console

1. Clean up in CloudFormation, as follows:
   b. Delete the codecatalyst-cfn-stack.
      Deleting the stack removes all tutorial resources from the API Gateway and Lambda services.
2. Clean up in Amazon S3, as follows:
   a. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
   b. Choose the codecatalyst-cfn-s3-bucket.
   c. Delete the bucket contents.
   d. Delete the bucket.
3. Clean up in IAM, as follows:
   b. Delete the codecatalyst-deploy-policy.
   c. Delete the codecatalyst-build-policy.
   d. Delete the codecatalyst-stack-policy.
   e. Delete the codecatalyst-deploy-role.
   f. Delete the codecatalyst-build-role.
   g. Delete the codecatalyst-stack-role.

In this tutorial, you learned how to deploy a serverless application as a CloudFormation stack using a CodeCatalyst workflow and a Deploy AWS CloudFormation stack action.

Tutorial: Deploy an application to Amazon ECS

In this tutorial, you learn how to deploy a serverless application into Amazon Elastic Container Service (Amazon ECS) using a workflow, Amazon ECS, and a few other AWS services. The deployed application is a simple Hello World website built on an Apache web server Docker image. The tutorial walks you through the required preparation work such as setting up a cluster, and then describes how to create a workflow to build and deploy the application.

Tip

Instead of working your way through this tutorial, you can use a blueprint that does a complete Amazon ECS setup for you. You'll need to use either the Node.js API with AWS Fargate or Java API with AWS Fargate blueprint. For more information, see Creating a project with a blueprint (p. 58).

Topics

* Prerequisites (p. 184)
* Step 1: Set up an AWS user and AWS CloudShell (p. 184)
• **Step 2:** Deploy a placeholder application into Amazon ECS (p. 187)
• **Step 3:** Create an Amazon ECR image repository (p. 190)
• **Step 4:** Create AWS roles (p. 191)
• **Step 5:** Add AWS roles to CodeCatalyst (p. 199)
• **Step 6:** Create a source repository (p. 199)
• **Step 7:** Add source files (p. 200)
• **Step 8:** Create and run a workflow (p. 202)
• **Step 9:** Make a change to your source files (p. 205)
• **Clean up** (p. 206)

**Prerequisites**

Before you begin:

• You need a CodeCatalyst **space** with a connected AWS account. For more information, see **Creating additional spaces in CodeCatalyst** (p. 53).
• In your space, you need an empty, **Start from scratch** CodeCatalyst **project** called:

  codecatalyst-ecs-project

  For more information, see **Creating an empty project in Amazon CodeCatalyst** (p. 58).
• In your project, you need a CodeCatalyst **environment** called:

  codecatalyst-ecs-environment

  Configure this environment as follows:
  • Choose any type, such as **Non-production**.
  • Connect your AWS account to it.

  For more information, see **Working with environments** (p. 307).

**Step 1: Set up an AWS user and AWS CloudShell**

The first step in this tutorial is to create a user in AWS IAM Identity Center, and launch an AWS CloudShell instance as this user. For the duration of this tutorial, CloudShell is your development computer and is where you configure AWS resources and services. Delete this user after completing the tutorial.

  **Note**
  Do not use your root user for this tutorial. You must create a separate user or else you may experience problems when performing actions in the AWS Command Line Interface (CLI) later on.

  For more information about IAM Identity Center users and CloudShell, see the **AWS IAM Identity Center User Guide** and **AWS CloudShell User Guide**.

  **To create an IAM Identity Center user**

  1. Sign in to the AWS Management Console and open the AWS IAM Identity Center console at [https://console.aws.amazon.com/singlesignon/](https://console.aws.amazon.com/singlesignon/).
**Note**
Make sure you sign in using the AWS account that is connected to your CodeCatalyst space. You can verify which account is connected by navigating to your space and choosing the **AWS accounts** tab. For more information, see *Creating additional spaces in CodeCatalyst* (p. 53).

2. In the navigation pane, choose **Users**, and then choose **Add user**.
3. In **Username**, enter:

   CodeCatalystECSUser

4. Under **Password**, choose **Generate a one-time password that you can share with this user**.
5. In **Email address** and **Confirm email address**, enter an email address that doesn't already exist in IAM Identity Center.
6. In **First name** and **Last name**, enter:

   CodeCatalystECSUser

7. In **Display name**, keep the automatically generated name:

   CodeCatalystECSUser CodeCatalystECSUser

8. Choose **Next**.
9. On the **Add user to groups** page, choose **Next**.
10. On the **Review and add user** page, review the information and choose **Add user**.

   A **One-time password** dialog box appears.
11. Choose **Copy** and then paste the sign-in information, including the AWS access portal URL and the one-time password.
12. Choose **Close**.

**To create a permission set**

You'll assign this permission set to CodeCatalystECSUser later.

1. In the navigation pane, choose **Permission sets**, and then choose **Create permission set**.
2. Choose **Predefined permission set** and then select **AdministratorAccess**. This policy provides full permissions to all AWS services.
3. Choose **Next**.
4. In **Permission set name**, enter:

   CodeCatalystECSPermissionSet

5. Choose **Next**.
6. On the **Review and create** page, review the information and choose **Create**.

**To assign the permission set to CodeCatalystECSUser**

1. In the navigation pane, choose **AWS accounts**, and then select the check box next to the AWS account that you're currently signed in to.
2. Choose **Assign users or groups**.
3. Choose the **Users** tab.
4. Select the check box next to CodeCatalystECSUser.
5. Choose Next.
6. Select the check box next to CodeCatalystECSPermissionSet.
7. Choose Next.
8. Review the information and choose Submit.

You have now assigned CodeCatalystECSUser and CodeCatalystECSPermissionSet to your AWS account, binding them together.

**To sign out and sign back in as CodeCatalystECSUser**

1. Before you sign out, make sure you have the AWS access portal URL and the username and one-time password for CodeCatalystECSUser. You should have copied this information to a text editor earlier.

   **Note**
   If you do not have this information, go to the CodeCatalystECSUser details page in IAM Identity Center, choose Reset password, Generate a one-time password [...], and Reset password again to display the information on the screen.

2. Sign out of AWS.
3. Paste the AWS access portal URL into your browser’s address bar.
4. Sign in with the username and one-time password for CodeCatalystECSUser.
5. In New password, enter a password, and choose Set new password.

   An AWS account box appears on the screen.
6. Choose AWS account, and then choose the name of the AWS account to which you assigned the CodeCatalystECSUser user and permission set.
7. Next to the CodeCatalystECSPermissionSet, choose Management console.

   The AWS Management Console appears. You are now signed in as CodeCatalystECSUser with the appropriate permissions.

**To launch an AWS CloudShell instance**

1. As CodeCatalystECSUser, in the top navigation bar, choose the AWS icon.

   ![AWS icon](image)

   The main page of the AWS Management Console appears.
2. In the top navigation bar, choose the AWS CloudShell icon.

   ![CloudShell icon](image)

   CloudShell opens. Wait while the CloudShell environment is created.

   **Note**
   If you don’t see the CloudShell icon, make sure that you’re in a Region supported by CloudShell. This tutorial assumes you are in the US West (Oregon) Region.

**To verify that the AWS CLI is installed**

1. In the CloudShell terminal, enter:

   ```shell
codecatalyst --version
   ```
2. Check that a version appears.

The AWS CLI is already configured for the current user, CodeCatalystECSUser, so there is no need to configure AWS CLI keys and credentials, as is normally the case.

**Step 2: Deploy a placeholder application into Amazon ECS**

In this section, you manually deploy a placeholder application into Amazon ECS. This placeholder application will be replaced by the Hello World application deployed by your workflow. The placeholder application is Apache Web Server.

For more information about Amazon ECS, see the *Amazon Elastic Container Service Developer Guide*.

Complete the following series of procedures to deploy the placeholder application.

**To create the task execution role**

This role grants Amazon ECS and AWS Fargate (Fargate) permission to make API calls on your behalf.

1. Create a trust policy:
   a. In AWS CloudShell, enter the following command:

   ```bash
   cat > codecatalyst-ecs-trust-policy.json
   ```

   A blinking prompt appears in the CloudShell terminal.

   b. Enter the following code at the prompt:

   ```json
   {
   "Version": "2012-10-17",
   "Statement": [
   {
   "Sid": "",
   "Effect": "Allow",
   "Principal": {
   "Service": "ecs-tasks.amazonaws.com"
   },
   "Action": "sts:AssumeRole"
   }
   ]
   }
   ```

   c. Place your cursor after the last curly bracket (}).

   d. Press **Enter** and then **Ctrl+d** to save the file and exit `cat`.

2. Create a task execution role:

   ```bash
   aws iam create-role \
   --role-name codecatalyst-ecs-task-execution-role \
   --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
   ```

3. Attach the AWS managed `AmazonECSTaskExecutionRolePolicy` policy to the role:

   ```bash
   aws iam attach-role-policy \
   --role-name codecatalyst-ecs-task-execution-role \
   --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
   ```

4. Display the role's details:
aws iam get-role \\
   --role-name codecatalyst-ecs-task-execution-role

5. Note the role's "Arn": value, for example, arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role. You will need this Amazon Resource Name (ARN) later.

To create an Amazon ECS cluster

This cluster will contain the Apache placeholder application, and later, the Hello World application.

1. As CodeCatalystECSUser, in AWS CloudShell, create an empty cluster:

   aws ecs create-cluster --cluster-name codecatalyst-ecs-cluster

2. (Optional) Verify that the cluster was created successfully:

   aws ecs list-clusters

   The ARN of the codecatalyst-ecs-cluster cluster should appear in the list, indicating a successful creation.

To create a task definition file

The task definition file indicates to run the Apache 2.4 Web server Docker image (httpd:2.4) which is pulled from DockerHub.

1. As CodeCatalystECSUser, in AWS CloudShell, create a task definition file:

   cat > taskdef.json

2. Paste the following code at the prompt:

   ```json
   {
   "executionRoleArn": "$arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role$",
   "containerDefinitions": [
   {
   "name": "codecatalyst-ecs-container",
   "image": "httpd:2.4",
   "essential": true,
   "portMappings": [
   {
   "hostPort": 80,
   "protocol": "tcp",
   "containerPort": 80
   }
   ]
   },
   "requiresCompatibilities": [
   "FARGATE"
   ],
   "cpu": "256",
   "family": "codecatalyst-ecs-task-def",
   "memory": "512",
   "networkMode": "awsvpc"
   }
   ```
In the preceding code, replace `arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role` with the ARN of the task execution role that you noted in To create the task execution role (p. 187).

3. Place your cursor after the last curly bracket `}`.
4. Press `Enter` and then `Ctrl+d` to save the file and exit `cat`.

To register the task definition file with Amazon ECS

1. As CodeCatalystECSUser, in AWS CloudShell, register the task definition:

   ```bash
   aws ecs register-task-definition --cli-input-json file://taskdef.json
   ```

2. (Optional) Verify that the task definition was registered:

   ```bash
   aws ecs list-task-definitions
   ```

   The `codecatalyst-ecs-task-def` task definition should appear in the list.

To create the Amazon ECS service

The Amazon ECS service runs the tasks (and associated Docker containers) of the Apache placeholder application, and later, the Hello World application.

1. As CodeCatalystECSUser, switch to the Amazon Elastic Container Service console if you haven't done so already.
2. Choose the cluster you created earlier, `codecatalyst-ecs-cluster`.
3. In the Services tab, choose Create.
4. In the Create page, do the following:
   a. Keep all default settings except for those listed next.
   b. For Launch type, choose FARGATE.
   c. Under Task definition, in the Family drop-down list, choose:

```
codecatalyst-ecs-task-def
```

d. For Service name, enter:

```
codecatalyst-ecs-service
```

e. For Desired tasks, enter:

```
3
```

In this tutorial, each task launches a single Docker container.

f. Expand the Networking section.

g. For VPC, choose any VPC.

h. For Subnets, choose any subnet.

Note

Only specify one subnet. That's all that is needed for this tutorial.
Note
If you don’t have a VPC and subnet, create them. See Create a VPC, and Create a subnet in your VPC in the Amazon VPC User Guide.

i. For Security group, choose Create a new security group, and then do the following:

   i. For Security group name, enter:

   codecatalyst-ecs-security-group

   ii. For Security group description, enter:

   CodeCatalyst ECS security group

   iii. Choose Add rule. For Type, choose HTTP, and for Source, choose Anywhere.

   j. At the bottom, choose Create.

   k. Wait while the service is created. This may take a few minutes.

5. Choose the Tasks tab, and then choose the refresh button. Verify that all three tasks have their Last Status column set to Running.

(Optional) To verify that your Apache placeholder application is running

1. In the Tasks tab, choose any one of the three tasks.

2. In the Public IP field, choose open address.

   An It Works! page appears. This indicates that the Amazon ECS service successfully started a task that launched a Docker container with the Apache image.

   At this point in the tutorial, you have manually deployed an Amazon ECS cluster, service, and task definition, as well as an Apache placeholder application. With all these items in place, you are now ready to create a workflow that will replace the Apache placeholder application with the tutorial’s Hello World application.

Step 3: Create an Amazon ECR image repository

In this section, you create a private image repository in Amazon Elastic Container Registry (Amazon ECR). This repository stores the tutorial’s Docker image that will replace the Apache placeholder image you deployed previously.

For more information about Amazon ECR, see the Amazon Elastic Container Registry User Guide.

To create an image repository in Amazon ECR

1. As CodeCatalystECSUser, in AWS CloudShell, create an empty repository in Amazon ECR:

   ```bash
   aws ecr create-repository --repository-name codecatalyst-ecs-image-repo
   ```

2. Display the Amazon ECR repository’s details:

   ```bash
   aws ecr describe-repositories \
   --repository-names codecatalyst-ecs-image-repo
   ```

3. Note the “repositoryUri”: value, for example, 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo.

   You need it later when adding the repository to your workflow.
Step 4: Create AWS roles

In this section, you create AWS IAM roles that your CodeCatalyst workflow will need in order to function. These roles are:

- **Build role** – Grants the CodeCatalyst build action (in the workflow) permission to access your AWS account and write to Amazon ECR and Amazon EC2.
- **Deploy role** – Grants the CodeCatalyst Deploy to ECS action (in the workflow) permission to access your AWS account, Amazon ECS, and a few other AWS services.

For more information about IAM roles, see IAM roles in the AWS Identity and Access Management User Guide.

**Note**

To save time, you can create a single role, called the CodeCatalystWorkflowDevelopmentRole-`spaceName` role, instead of the two roles listed previously. For more information, see Creating the CodeCatalystWorkflowDevelopmentRole-`spaceName` role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-`spaceName` role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern. This tutorial assumes you are creating the two roles listed previously.

To create the build and deploy roles, you can use either the AWS Management Console or the AWS CLI.

**AWS Management Console**

To create the build and deploy roles, complete the following series of procedures.

**To create a build role**

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   b. Open the IAM console at https://console.aws.amazon.com/iam/.
   c. In the navigation pane, choose Policies.
   d. Choose Create policy.
   e. Choose the JSON tab.
   f. Delete the existing code.
   g. Paste the following code:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "ecr:*",
           "ec2:*"
         ],
         "Resource": "*"
       }
     ]
   }
   ```
Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

h. Choose Next: Tags.

i. Choose Next: Review.

j. In Name, enter:

   codecatalyst-ecs-build-policy

k. Choose Create policy.

   You have now created a permissions policy.

2. Create the build role, as follows:

   a. In the navigation pane, choose Roles, and then choose Create role.

   b. Choose Custom trust policy.

   c. Delete the existing custom trust policy.

   d. Add the following custom trust policy:

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": [
             "codecatalyst-runner.amazonaws.com",
             "codecatalyst.amazonaws.com"
           ],
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   e. Choose Next.

   f. In Permissions policies, search for codecatalyst-ecs-build-policy, select its check box.

   g. Choose Next.

   h. For Role name, enter:

      codecatalyst-ecs-build-role

   i. For Role description, enter:

      CodeCatalyst ECS build role

   j. Choose Create role.

   You have now created a build role with a permissions policy and a trust policy.
3. Obtain the build role ARN, as follows:
   a. In the navigation pane, choose Roles.
   b. In the search box, enter the name of the role you just created (codecatalyst-ecs-build-role).
   c. Choose the role from the list.
      The role's Summary page appears.
   d. At the top, copy the ARN value. You need it later.

To create a deploy role

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   b. Open the IAM console at https://console.aws.amazon.com/iam/.
   c. In the navigation pane, choose Policies.
   d. Choose Create Policy.
   e. Choose the JSON tab.
   f. Delete the existing code.
   g. Paste the following code:

```json
{
   "Version": "2012-10-17",
   "Statement": [{
      "Action": ["ecs:DescribeServices",
                  "ecs:CreateTaskSet",
                  "ecs:DeleteTaskSet",
                  "ecs:ListClusters",
                  "ecs:RegisterTaskDefinition",
                  "ecs:UpdateServicePrimaryTaskSet",
                  "ecs:UpdateService",
                  "elasticloadbalancing:DescribeTargetGroups",
                  "elasticloadbalancing:DescribeListeners",
                  "elasticloadbalancing:ModifyListener",
                  "elasticloadbalancing:DescribeRules",
                  "elasticloadbalancing:ModifyRule",
                  "lambda:InvokeFunction",
                  "lambda:ListFunctions",
                  "cloudwatch:DescribeAlarms",
                  "sns:Publish",
                  "sns:ListTopics",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "codedeploy:CreateApplication",
                  "codedeploy:CreateDeployment",
                  "codedeploy:CreateDeploymentGroup",
                  "codedeploy:GetApplication",
                  "codedeploy:GetDeployment",
                  "codedeploy:GetDeploymentGroup",
                  "codedeploy:ListApplications",
                  "codedeploy:ListDeploymentGroups",
                  "codedeploy:ListDeployments",
                  "codedeploy:StopDeployment",
                  "codedeploy:GetDeploymentTarget",
                  "codedeploy:ListDeploymentTargets",
                  "codedeploy:GetDeploymentConfig",
                  "codedeploy:GetApplicationRevision",
                  "lambda:UpdateFunctionVersion"
    ],
   },
   {
      "Action": ["ecs:UpdateService"]
    }
  ]
}
```
"codedeploy:RegisterApplicationRevision",
"codedeploy:BatchGetApplicationRevisions",
"codedeploy:BatchGetDeploymentGroups",
"codedeploy:BatchGetDeployments",
"codedeploy:BatchGetApplications",
"codedeploy:ListApplicationRevisions",
"codedeploy:ListDeploymentConfigs",
"codedeploy:ContinueDeployment"
],
"Resource":"*",
"Effect":"Allow"
},
{
"Action":
["iam:PassRole"
],
"Effect":"Allow",
"Resource":"*",
"Condition":{
"StringLike":{"iam:PassedToService":[
  "ecs-tasks.amazonaws.com",
  "codedeploy.amazonaws.com"
]}
}
}

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement. You can then scope down the policy with the resource name after it is available.

"Resource": "*

---

h. Choose **Next: Tags**.

i. Choose **Next: Review**.

j. In **Name**, enter:

```
codcatalyst-ecs-deploy-policy
```

k. Choose **Create policy**.

You have now created a permissions policy.

2. Create the deploy role, as follows:

a. In the navigation pane, choose **Roles**, and then choose **Create role**.

b. Choose **Custom trust policy**.

c. Delete the existing custom trust policy.

d. Add the following custom trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ]
      }
    }
  ]
}
```
e. Choose **Next**.
f. In **Permissions policies**, search for `codecatalyst-ecs-deploy-policy`, select its check box.
g. Choose **Next**.
h. For **Role name**, enter:

```
codecatalyst-ecs-deploy-role
```

i. For **Role description**, enter:

```
CodeCatalyst ECS deploy role
```

j. Choose **Create role**.

You have now created a deploy role with a trust policy.

3. Obtain the deploy role ARN, as follows:
   a. In the navigation pane, choose **Roles**.
   b. In the search box, enter the name of the role you just created (`codecatalyst-ecs-deploy-role`).
   c. Choose the role from the list.
      The role's **Summary** page appears.
   d. At the top, copy the **ARN** value. You need it later.

**AWS CLI**

To create the build and deploy roles, complete the following series of procedures.

**To create a trust policy for both roles**

As **CodeCatalystECSUser**, in AWS CloudShell, create a trust policy file:

1. Create the file:

```
cat > codecatalyst-ecs-trust-policy.json
```

2. At the terminal prompt, paste the following code:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ],
                "Action": "sts:AssumeRole"
            }
        }
    ]
}
```
3. Place your cursor after the last curly bracket (}).
4. Press Enter and then Ctrl+d to save the file and exit cat.

To create the build policy and build role
1. Create the build policy:
   a. As CodeCatalystECSUser, in AWS CloudShell, create a build policy file:

   ```bash
   cat > codecatalyst-ecs-build-policy.json
   ```

   b. At the prompt, enter the following code:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "ecr:*",
           "ec2:*
         ],
         "Resource": "*
       }
     ]
   }
   ```

   c. Place your cursor after the last curly bracket (}).
   d. Press Enter and then Ctrl+d to save the file and exit cat.
2. Add the build policy to AWS:

   ```bash
   aws iam create-policy \
   --policy-name codecatalyst-ecs-build-policy \
   --policy-document file://codecatalyst-ecs-build-policy.json
   ```

3. In the command output, note the "arn": value, for example, `arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy`. You need this ARN later.
4. Create the build role and attach the trust policy to it:

   ```bash
   aws iam create-role \
   --role-name codecatalyst-ecs-build-role \
   --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
   ```

5. Attach the build policy to the build role:

   ```bash
   aws iam attach-role-policy \
   --role-name codecatalyst-ecs-build-role \
   --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy
   ```

   Where `arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy` is replaced with the ARN of the build policy you noted earlier.
6. Display the build role’s details:
aws iam get-role \
   --role-name codecatalyst-ecs-build-role

7. Note the role's "Arn": value, for example, arn:aws:iam::111122223333:role/ codecatalyst-ecs-build-role. You need this ARN later.

To create the deploy policy and deploy role

1. Create a deploy policy:
   a. In AWS CloudShell, create a deploy policy file:

```
cat > codecatalyst-ecs-deploy-policy.json
```
   b. At the prompt, enter the following code:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {"Action": ["ecs:DescribeServices",
                  "ecs:CreateTaskSet",
                  "ecs:DeleteTaskSet",
                  "ecs:ListClusters",
                  "ecs:RegisterTaskDefinition",
                  "ecs:UpdateServicePrimaryTaskSet",
                  "ecs:UpdateService",
                  "elasticloadbalancing:DescribeTargetGroups",
                  "elasticloadbalancing:DescribeListeners",
                  "elasticloadbalancing:DescribeRules",
                  "elasticloadbalancing:ModifyListener",
                  "lambda:InvokeFunction",
                  "lambda:ListFunctions",
                  "cloudwatch:DescribeAlarms",
                  "sns:Publish",
                  "sns:ListTopics",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "codedeploy:CreateApplication",
                  "codedeploy:CreateDeployment",
                  "codedeploy:CreateDeploymentGroup",
                  "codedeploy:GetApplication",
                  "codedeploy:GetDeployment",
                  "codedeploy:GetDeploymentGroup",
                  "codedeploy:ListApplications",
                  "codedeploy:ListDeploymentGroups",
                  "codedeploy:ListDeployments",
                  "codedeploy:StopDeployment",
                  "codedeploy:GetDeploymentTarget",
                  "codedeploy:ListDeploymentTargets",
                  "codedeploy:GetDeploymentConfig",
                  "codedeploy:GetApplicationRevision",
                  "codedeploy:RegisterApplicationRevision",
                  "codedeploy:BatchGetApplicationRevisions",
                  "codedeploy:BatchGetDeploymentGroups",
                  "codedeploy:BatchGetDeployments",
                  "codedeploy:BatchGetApplications",
                  "codedeploy:ListApplicationRevisions",
                  "codedeploy:ListDeploymentConfigs",
                  "codedeploy:ContinueDeployment"
   ]
}
```
Note
The first time the role is used to run workflow actions, use the wildcard in the
resource policy statement and then scope down the policy with the resource name
after it is available.

"Resource": "*

c. Place your cursor after the last curly bracket {).
d. Press Enter and then Ctrl+d to save the file and exit cat.

2. Add the deploy policy to AWS:

```bash
aws iam create-policy \
--policy-name codecatalyst-ecs-deploy-policy \
--policy-document file://codecatalyst-ecs-deploy-policy.json
```

3. In the command output, note the deploy policy's "arn": value, for example,
this ARN later.

4. Create the deploy role and attach the trust policy to it:

```bash
aws iam create-role \
--role-name codecatalyst-ecs-deploy-role \
--assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. Attach the deploy policy to the deploy role, where arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy is replaced with the ARN of the deploy policy you noted earlier.

```bash
aws iam attach-role-policy \
--role-name codecatalyst-ecs-deploy-role \
--policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy
```

6. Display the deploy role's details:

```bash
aws iam get-role \
--role-name codecatalyst-ecs-deploy-role
```

7. Note the role's "Arn": value, for example, arn:aws:iam::111122223333:role/codecatalyst-ecs-deploy-role. You need this ARN later.
Step 5: Add AWS roles to CodeCatalyst

In this step, you add the build role (`codecatalyst-ecs-build-role`) and deploy role (`codecatalyst-ecs-deploy-role`) to the CodeCatalyst account connection in your space.

To add build and deploy roles to your account connection

1. In CodeCatalyst, navigate to your space.
2. Choose AWS accounts. A list of account connections appears.
3. Choose the account connection that represents the AWS account where you created your build and deploy roles.
4. Choose Manage roles from AWS management console. The Add IAM role to Amazon CodeCatalyst space page appears. You might need to sign in to access the page.
5. Select Add an existing role you have created in IAM.

A drop-down list appears. The list displays all IAM roles with a trust policy that includes the `codecatalyst-runner.amazonaws.com` and `codecatalyst.amazonaws.com` service principals.
6. In the drop-down list, choose `codecatalyst-ecs-build-role`, and choose Add role.

Note
If you see The security token included in the request is invalid, it might be because you do not have the right permissions. To fix this issue, sign out of AWS as sign back in with the AWS account that you used when you created your CodeCatalyst space.
7. Choose Add IAM role, choose Add an existing role you have created in IAM, and in the drop-down list, choose `codecatalyst-ecs-deploy-role`. Choose Add role.

You have now added the build and deploy roles to your space.
8. Copy the value of the Amazon CodeCatalyst display name. You’ll need this value later, when creating your workflow.

Step 6: Create a source repository

In this step, you create a source repository in CodeCatalyst. This repository stores the tutorial's source files, such as the task definition file.

For more information about source repositories, see Creating a source repository (p. 89).

To create a source repository

2. Navigate to your project, `codecatalyst-ecs-project`.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Create repository.
5. In Repository name, enter:

   `codecatalyst-ecs-source-repository`

6. Choose Create.
Step 7: Add source files

In this section, you add the Hello World source files to your CodeCatalyst repository, codecatalyst-ecs-source-repository. They consist of:

- An `index.html` file – Displays a Hello World message in the browser.
- A Dockerfile – Describes the base image to use for your Docker image and the Docker commands to apply to it.
- A `taskdef.json` file – Defines the Docker image to use when launching tasks into your cluster.

The folder structure is as follows:

```
.
 | — public-html
 |   | — index.html
 | — Dockerfile
 | — taskdef.json
```

**Note**
The following instructions show you how to add the files using the CodeCatalyst console but you can use Git if you prefer. For details, see Cloning a source repository (p. 92).

**Topics**
- `index.html` (p. 200)
- `Dockerfile` (p. 201)
- `taskdef.json` (p. 201)

**index.html**
The `index.html` file displays a Hello World message in the browser.

**To add the `index.html` file**

1. In the CodeCatalyst console, go to your source repository, codecatalyst-ecs-source-repository.
2. In Files, choose Create file.
3. For File name, enter:

   `public-html/index.html`

   **Important**
   Make sure to include the `public-html/` prefix to create a folder of the same name. The `index.html` is expected to be in this folder.

4. In the text box, enter the following code:

   ```html
   <html>
   <head>
   <title>Hello World</title>
   <style>
   body {
   background-color: black;
   text-align: center;
   color: white;
   ```
5. Choose **Commit**, and then choose **Commit** again.

The `index.html` is added to your repository in a `public-html` folder.

**Dockerfile**

The Dockerfile describes the base Docker image to use and the Docker commands to apply to it. For more information about the Dockerfile, see the [Dockerfile Reference](#).

The Dockerfile specified here indicates to use the Apache 2.4 base image (`httpd`). It also includes instructions for copying a source file called `index.html` to a folder on the Apache server that serves webpages. The `EXPOSE` instruction in the Dockerfile tells Docker that the container is listening on port 80.

**To add the Dockerfile**

1. In your source repository, choose **Create file**.
2. For **File name**, enter:

```
Dockerfile
```

Do not include a file extension.

**Important**

The Dockerfile must reside in your repository's root folder. The workflow's Docker build command expects it to be there.

3. In the text box, enter the following code:

```python
FROM httpd:2.4
EXPOSE 80
```

4. Choose **Commit**, and then choose **Commit** again.

The Dockerfile is added to your repository.

**taskdef.json**

The `taskdef.json` file that you add in this step is the same as the one you already specified in **Step 2: Deploy a placeholder application into Amazon ECS (p. 187)** with the following difference:

Instead of specifying a hardcoded Docker image name in the `image:` field (`httpd:2.4`), the task definition here uses a couple of variables to denote the image: `$REPOSITORY_URI` and `$IMAGE_TAG`. These variables will be replaced with real values generated by the workflow's build action when you run the workflow in a later step.

For details on the task definition parameters, see [Task definition parameters](#) in the [Amazon Elastic Container Service Developer Guide](#).
To add the taskdef.json file

1. In your source repository, choose Create file.
2. For File name, enter:

   taskdef.json

3. In the text box, enter the following code:

   ```json
   {
       "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role",
       "containerDefinitions": [
         {
           "name": "codecatalyst-ecs-container",
           # The $REPOSITORY_URI and $IMAGE_TAG variables will be replaced
           # by the workflow at build time (see the build action in the
           # workflow)
           "image": $REPOSITORY_URI:$IMAGE_TAG,
           "essential": true,
           "portMappings": [
             {
               "hostPort": 80,
               "protocol": "tcp",
               "containerPort": 80
             }
           ]
         }
       ],
       "requiresCompatibilities": ["FARGATE"],
       "networkMode": "awsvpc",
       "cpu": "256",
       "memory": "512",
       "family": "codecatalyst-ecs-task-def"
   }
   ```

   In the preceding code, replace

   `arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role`

   with the ARN of the task execution role that you noted in To create the task execution role (p. 187).

4. Choose Commit, and then choose Commit again.

   The taskdef.json file is added to your repository.

Step 8: Create and run a workflow

In this step, you create a workflow that takes your source files, builds them into a Docker image, and then deploys the image to your Amazon ECS cluster. This deployment replaces the existing Apache placeholder application.

The workflow consists of the following building blocks that run sequentially:

- A trigger – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see Working with triggers (p. 327).
- A build action (BuildBackend) – On trigger, the action builds the Docker image using the Dockerfile and pushes the image to Amazon ECR. The build action also updates the taskdef.json with the
correct image field value, and then creates an output artifact of this file. This artifact is used as the input for the deploy action, which is next.

For more information about the build action, see Building using workflows in CodeCatalyst (p. 138).

- A deploy action (DeployToECS) – On completion of the build action, the deploy action looks for the output artifact generated by the build action (TaskDefArtifact), finds the taskdef.json inside of it, and registers it with your Amazon ECS service. The service then follows the instructions in the taskdef.json file to run three Amazon ECS tasks—and associated Hello World Docker containers—inside your Amazon ECS cluster.

To create a workflow

1. In the CodeCatalyst console, in the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose Create workflow.
3. For Source repository, choose codecatalyst-ecs-source-repository.
4. For Branch, choose main.
5. Choose Create.
6. Delete the YAML sample code.
7. Add the following YAML code:

```yaml
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
- Type: PUSH
  Branches:
    - main

Actions:
  BuildBackend:
    Identifier: aws/build@v1
  Environment:
    Name: codecatalyst-ecs-environment
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-ecs-build-role

Inputs:
  Sources:
    - WorkflowSource
  Variables:
    - Name: REPOSITORY_URI
      Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo
    - Name: IMAGE_TAG
      Value: ${WorkflowSource.CommitId}

Configuration:
  Steps:
    #pre_build:
      - Run: echo Logging in to Amazon ECR...
      - Run: aws --version
      - Run: aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
    #build:
      - Run: echo Build started on 'date'
      - Run: echo Building the Docker image...
      - Run: docker build -t $REPOSITORY_URI:latest .
      - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
    #post_build:
      - Run: echo Build completed on 'date'
      - Run: echo Pushing the Docker images...
      - Run: docker push $REPOSITORY_URI:latest
```
- Run: docker push $REPOSITORY_URI:$IMAGE_TAG
  # Replace the variables in taskdef.json
- Run: find taskdef.json -type f | xargs sed -i "s|$REPOSITORY_URI|\$REPOSITORY_URI|g"
- Run: find taskdef.json -type f | xargs sed -i "s|\$IMAGE_TAG|\$IMAGE_TAG|g"
- Run: cat taskdef.json
  # The output artifact will be a zip file that contains a task definition file.

Outputs:
  Artifacts:
    - Name: TaskDefArtifact
      Files:
        - taskdef.json

DeployToECS:
  DependsOn:
    - BuildBackend
  Identifier: aws/ecs-deploy@v1
  Environment:
    Name: codecatalyst-ecs-environment
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-ecs-deploy-role
  Inputs:
    Sources: []
    Artifacts:
      - TaskDefArtifact
  Configuration:
    region: us-west-2
    cluster: codecatalyst-ecs-cluster
    service: codecatalyst-ecs-service
    task-definition: taskdef.json

In the preceding code, replace:

- Both instances of **codecatalyst-ecs-environment** with the name of the environment you created in [Prerequisites](p. 184).

- Both instances of **codecatalyst-account-connection** with the display name of your account connection. The display name might be a number. For more information, see [Step 5: Add AWS roles to CodeCatalyst](p. 199).

- **codecatalyst-ecs-build-role** with the name of the build role you created in [Step 4: Create AWS roles](p. 191).

- `111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo` (in the **Value:** property) with the URI of the Amazon ECR repository you created in [Step 3: Create an Amazon ECR image repository](p. 190).

- `111122223333.dkr.ecr.us-west-2.amazonaws.com` (in the Run: `aws ecr` command) with the URI of the Amazon ECR repository without the image suffix (`/codecatalyst-ecs-image-repo`).

- **codecatalyst-ecs-deploy-role** with the name of the deploy role you created in [Step 4: Create AWS roles](p. 191).

- Both instances of **us-west-2** with your AWS Region code. For a list of Region codes, see [Regional endpoints](in the AWS General Reference).

**Note**

If you decided not to create build and deploy roles, replace **codecatalyst-ecs-build-role** and **codecatalyst-ecs-deploy-role** with the name of the CodeCatalystWorkFlowDevelopmentRole-*spaceName* role. For more information about this role, see [Step 4: Create AWS roles](p. 191).
Tip
Instead of using the `find` and `sed` commands shown in the previous workflow code to update the repository and image name, you can use the **Render Amazon ECS task definition** action for this purpose. For more information, see Adding the "Render Amazon ECS task definition" action (p. 260).

8. (Optional) Choose **Validate** to make sure that the YAML code is valid before committing.

9. Choose **Commit**.

10. In the **Commit workflow** dialog box, enter the following:

    a. For **Commit message**, remove the text and enter:

    ```yaml
    Add first workflow
    ```

    b. For **Repository**, choose codecatalyst-ecs-source-repository.

    c. For **Branch name**, choose main.

    d. Choose **Commit**.

You have now created a workflow. A workflow run starts automatically because of the trigger defined at the top of the workflow. Specifically, when you committed (and pushed) the **workflow.yaml** file to your source repository, the trigger started the workflow run.

**To view the workflow run progress**

1. In the navigation pane of the CodeCatalyst console, choose **CI/CD**, and then choose **Workflows**.

2. Choose the workflow you just created, codecatalyst-ecs-workflow.

3. Choose **BuildBackend** to see the build progress.

4. Choose **DeployToECS** to see the deployment progress.

   For more information about viewing run details, see Viewing workflow run status and details (p. 315).

**To verify the deployment**

1. Open the Amazon ECS classic console at [https://console.aws.amazon.com/ecs/](https://console.aws.amazon.com/ecs/).

2. Choose your cluster, codecatalyst-ecs-cluster.

3. Choose the **Tasks** tab.

4. Choose any one of the three tasks.

5. In the **Public IP** field, choose open address.

   A "Hello World" page appears in the browser, indicating that the Amazon ECS service successfully deployed your application.

**Step 9: Make a change to your source files**

In this section, you make a change to the **index.html** file in your source repository. This change causes the workflow to build a new Docker image, tag it with a commit ID, push it to Amazon ECR, and deploy it to Amazon ECS.
**To change the index.html**

1. In the CodeCatalyst console, in the navigation pane, choose **Code**, then choose **Source repositories**, and then choose your repository, `codecatalyst-ecs-source-repository`.
2. Choose **public-html**, and then choose **index.html**.
   
   The contents of `index.html` appear.
3. Choose **Edit**.
4. On line 14, change the `Hello World` text to `Tutorial complete!`.
5. Choose **Commit**, and then choose **Commit** again.
   
   The commit causes a new workflow run to start.
6. (Optional) Go to your source repository's main page, choose **View commits**, and then note the commit ID for the `index.html` change.
7. Watch the deployment progress:
   
   a. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
   b. Choose `codecatalyst-ecs-workflow` to view the latest run.
   c. Choose **BuildBackend**, and **DeployToECS** to see the workflow run progress.
8. Verify that your application was updated, as follows:
   
   a. Open the Amazon ECS classic console at https://console.aws.amazon.com/ecs/.
   b. Choose your cluster, `codecatalyst-ecs-cluster`.
   c. Choose the **Tasks** tab.
   d. Choose any one of the three tasks.
   e. In the **Public IP** field, choose **open address**.

   A **Tutorial complete!** page appears.
9. (Optional) In AWS, switch to the Amazon ECR console and verify that the new Docker image was tagged with the commit ID from step 6.

**Clean up**

Clean up the files and services used in this tutorial to avoid being charged for them.

In the AWS Management Console, clean up in this order:

1. In Amazon ECS, do the following:
   
   a. Delete `codecatalyst-ecs-service`.
   b. Delete `codecatalyst-ecs-cluster`.
   c. Deregister `codecatalyst-ecs-task-definition`.
2. In Amazon ECR, delete `codecatalyst-ecs-image-repo`.
3. In Amazon EC2, delete `codecatalyst-ecs-security-group`.
4. In IAM Identity Center, delete:
   
   a. `CodeCatalystECSUser`
   b. `CodeCatalystECSPermissionSet`

   In the CodeCatalyst console, clean up as follows:
1. Delete `codecatalyst-ecs-workflow`. 

---

206
2. Delete codecatalyst-ecs-environment.
3. Delete codecatalyst-ecs-source-repository.
4. Delete codecatalyst-ecs-project.

In this tutorial, you learned how to deploy an application to an Amazon ECS service using a CodeCatalyst workflow and a **Deploy to Amazon ECS** action.

**Tutorial: Deploy an application to Amazon EKS**

In this tutorial, you learn how to deploy a containerized application into Amazon Elastic Kubernetes Service using an Amazon CodeCatalyst workflow, Amazon EKS, and a few other AWS services. The deployed application is a simple 'Hello, World!' website built on an Apache web server Docker image. The tutorial walks you through the required preparation work such as setting up a development machine and an Amazon EKS cluster, and then describes how to create a workflow to build the application and deploy it into the cluster.

After the initial deployment is complete, the tutorial instructs you to make a change to your application source. This change causes a new Docker image to be built and pushed to your Docker image repository with new revision information. The new revision of the Docker image is then deployed into Amazon EKS.

**Tip**
Instead of working your way through this tutorial, you can use a blueprint that does a complete Amazon EKS setup for you. You'll need to use the **EKS App Deployment** blueprint. For more information, see [Creating a project with a blueprint](p. 58).

**Topics**
- Prerequisites (p. 207)
- Step 1: Set up your development machine (p. 208)
- Step 2: Create an Amazon EKS cluster (p. 212)
- Step 3: Create an Amazon ECR image repository (p. 213)
- Step 4: Add source files (p. 214)
- Step 5: Create AWS roles (p. 217)
- Step 6: Add AWS roles to CodeCatalyst (p. 220)
- Step 7: Update the ConfigMap (p. 221)
- Step 8: Create and run a workflow (p. 223)
- Step 9: Make a change to your source files (p. 226)
- Clean up (p. 227)

**Prerequisites**

Before you begin this tutorial:

- You need an Amazon CodeCatalyst space with a connected AWS account. For more information, see [Creating additional spaces in CodeCatalyst](p. 53).
- In your space, you need an empty, **Start from scratch** CodeCatalyst project called:

```
codecatalyst-eks-project
```

For more information, see [Creating an empty project in Amazon CodeCatalyst](p. 58).
• In your project, you need an empty CodeCatalyst source repository called:

```
codecatalyst-eks-source-repository
```

For more information, see Source repositories in CodeCatalyst (p. 79).

• In your project, you need a CodeCatalyst CI/CD environment (not a Dev Environment) called:

```
codecatalyst-eks-environment
```

Configure this environment as follows:

• Choose any type, such as Non-production.
• Connect your AWS account to it.

For more information, see Working with environments (p. 307).

**Step 1: Set up your development machine**

The first step in this tutorial is to configure a development machine with a few tools that you'll use throughout this tutorial. These tools are:

• the eksctl utility – for cluster creation
• the kubectl utility – a prerequisite for eksctl
• the AWS CLI – also a prerequisite for eksctl

You can install these tools on your existing development machine if you have one, or you can use a CodeCatalyst Dev Environment, which is Cloud-based. The benefit of a CodeCatalyst Dev Environment is that it's easy to spin up and take down, and is integrated with other CodeCatalyst services, allowing you to work through this tutorial in fewer steps.

This tutorial assumes you'll be using a CodeCatalyst Dev Environment.

The following instructions describe a quick way to launch a CodeCatalyst Dev Environment and configure it with the required tools, but if you want detailed instructions, see:

• Creating a Dev Environment (p. 113) in this guide.
• Installing kubectl in the Amazon EKS User Guide.
• Installing or upgrading eksctl in the Amazon EKS User Guide.
• Installing or updating the latest version of the AWS CLI in the AWS Command Line Interface User Guide.

**To launch a Dev Environment**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project, codecatalyst-eks-project.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose the name of your source repository, codecatalyst-eks-source-repository.
5. Near the top choose Create Dev Environment, and then choose AWS Cloud9 (in browser).
6. Make sure that Work in existing branch and main are selected, and then choose Create.

    Your Dev Environment launches in a new browser tab, and your repository (codecatalyst-eks-source-repository) is cloned into it.
To install and configure kubectl

1. In the Dev Environment terminal, enter:

   curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl

2. Enter:

   chmod +x ./kubectl

3. Enter:

   mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin

4. Enter:

   echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc

5. Enter:

   kubectl version --short --client

6. Check that a version appears.

   You have now installed kubectl.

To install and configure eksctl

**Note**

eksctl is not strictly required because you can use kubectl instead. However, eksctl has the benefit of automating much of the cluster configuration, and is therefore the tool recommended for this tutorial.

1. In the Dev Environment terminal, enter:

   curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

2. Enter:

   sudo cp /tmp/eksctl /usr/bin

3. Enter:

   eksctl version

4. Check that a version appears.

   You have now installed eksctl.

To verify that the AWS CLI is installed

1. In the Dev Environment terminal, enter:

   aws --version
2. Check that a version appears to verify that the AWS CLI is installed.

Complete the remaining procedures to configure the AWS CLI with the necessary permissions to access AWS.

**To configure the AWS CLI**

You must configure the AWS CLI with access keys and a session token to give it access to AWS services. The following instructions provide a quick way to configure the keys and token, but if you want detailed instructions, see Configuring the AWS CLI in the AWS Command Line Interface User Guide.

1. Create an IAM Identity Center user, as follows:
   a. Sign in to the AWS Management Console and open the AWS IAM Identity Center console at https://console.aws.amazon.com/singlesignon/.
      (You might need to choose Enable if you’ve never signed in to IAM Identity Center before.)

      **Note**
      Make sure you sign in using the AWS account that is connected to your CodeCatalyst space. You can verify which account is connected by navigating to your space and choosing the AWS accounts tab. For more information, see Creating additional spaces in CodeCatalyst (p. 53).

   b. In the navigation pane, choose Users, and then choose Add user.

   c. In Username, enter:

      **codecatalyst-eks-user**

   d. Under Password, choose Generate a one-time password that you can share with this user.

   e. In Email address and Confirm email address, enter an email address that doesn't already exist in IAM Identity Center.

   f. In First name, enter:

      **codecatalyst-eks-user**

   g. In Last name, enter:

      **codecatalyst-eks-user**

   h. In Display name, keep:

      **codecatalyst-eks-user codecatalyst-eks-user**

   i. Choose Next.

   j. On the Add user to groups page, choose Next.

   k. On the Review and add user page, review the information and choose Add user.

      A One-time password dialog box appears.

   l. Choose Copy and then paste the sign-in information to a text file. The sign-in information consists of the AWS access portal URL, a user name, and a one-time password.

   m. Choose Close.

2. Create a permission set, as follows:
   a. In the navigation pane, choose Permission sets, and then choose Create permission set.
b. Choose **Predefined permission set** and then select **AdministratorAccess**. This policy provides full permissions to all AWS services.

c. Choose **Next**.

d. In **Permission set name**, remove AdministratorAccess and enter:

```
codcatalyst-eks-permission-set
```

e. Choose **Next**.

f. On the **Review and create** page, review the information and choose **Create**.

3. Assign the permission set to **codecatalyst-eks-user**, as follows:

a. In the navigation pane, choose **AWS accounts**, and then select the check box next to the AWS account that you're currently signed in to.

b. Choose **Assign users or groups**.

c. Choose the **Users** tab.

d. Select the check box next to **codecatalyst-eks-user**.

e. Choose **Next**.

f. Select the check box next to **codecatalyst-eks-permission-set**.

g. Choose **Next**.

h. Review the information and choose **Submit**.

You have now assigned **codecatalyst-eks-user** and **codecatalyst-eks-permission-set** to your AWS account, binding them together.

4. Obtain **codecatalyst-eks-user**'s access keys and session token, as follows:

a. Make sure you have the AWS access portal URL and the username and one-time password for **codecatalyst-eks-user**. You should have copied this information to a text editor earlier.

   **Note**
   If you do not have this information, go to the **codecatalyst-eks-user** details page in IAM Identity Center, choose **Reset password**, **Generate a one-time password [...]**, and **Reset password** again to display the information on the screen.

b. Sign out of AWS.

c. Paste the AWS access portal URL into your browser's address bar.

d. Sign in with:

   - **Username**:

   ```
codcatalyst-eks-user
```

   - **Password**:

   ```
one-time-password
```

e. In **Set new password**, enter a new password and choose **Set new password**.

   An **AWS account** box appears on the screen.

f. Choose **AWS account**, and then choose the name of the AWS account to which you assigned the **codecatalyst-eks-user** user and permission set.

g. Next to **codecatalyst-eks-permission-set**, choose **Command line or programmatic access**.

h. Copy the commands in the middle of the page. They look similar to the following:

```bash
export AWS_ACCESS_KEY_ID="AKIAI0SFODNN7EXAMPLE"
```
export AWS_SECRET_ACCESS_KEY="wJalrXUttnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_SESSION_TOKEN="session-token"

...where session-token is a long random string.

5. Add the access keys and session token to the AWS CLI, as follows:
   b. At the terminal prompt, paste the commands you copied. Press Enter.

You have now configured the AWS CLI with access keys and a session token. You can now use
AWS CLI to complete the tasks required by this tutorial.

Important
    If at any time during this tutorial you see messages similar to:
    Unable to locate credentials. You can configure credentials by running "aws configure".
    Or:
    ExpiredToken: The security token included in the request is expired
    ...it's because your AWS CLI session has expired. In this case, do not run the aws configure command. Instead, use the instructions in step 4 of this procedure that starts with Obtain codecatalyst-eks-user's access key and session token to refresh your session.

Step 2: Create an Amazon EKS cluster

In this section, you create a cluster in Amazon EKS. The instructions below describe a quick way to create the cluster using eksctl, but if you want detailed instructions, see:

- Getting started with eksctl in the Amazon EKS User Guide
  or
- Getting started with the console and AWS CLI in the Amazon EKS User Guide (this topic provides kubectl instructions for creating the cluster)

Note
    Private clusters are not supported by the CodeCatalyst integration with Amazon EKS.

Before you begin

Make sure you have completed the following tasks on your development machine:

- Installed the eksctl utility.
- Installed the kubectl utility.
- Installed the AWS CLI and configured it with access keys and a session token.

For information on how to complete these tasks, see Step 1: Set up your development machine (p. 208).

To create a cluster

Important
    Do not use the Amazon EKS service's user interface to create the cluster because the cluster won't be configured correctly. Use the eksctl utility, as described in the following steps.
1. Go to your Dev Environment.

2. Create a cluster and nodes:

   ```bash
   eksctl create cluster --name codecatalyst-eks-cluster --region us-west-2
   ```

   Where:
   - `codecatalyst-eks-cluster` is replaced with the name you want to give your cluster.
   - `us-west-2` is replaced with your Region.

   After 10-20 minutes, a message similar to the following appears:

   EKS cluster "codecatalyst-eks-cluster" in "us-west-2" region is ready

   **Note**
   You will see multiple waiting for CloudFormation stack messages while AWS creates your cluster. This is expected.

3. Verify that your cluster was created successfully:

   ```bash
   kubectl cluster-info
   ```

   You will see a message similar to the following, indicating a successful cluster creation:

   Kubernetes master is running at https://long-string.gr7.us-west-2.eks.amazonaws.com
   CoreDNS is running at https://long-string.gr7.us-west-2.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

**Step 3: Create an Amazon ECR image repository**

In this section, you create a private image repository in Amazon Elastic Container Registry (Amazon ECR). This repository stores the Docker image for the tutorial.

For more information about Amazon ECR, see the *Amazon Elastic Container Registry User Guide*.

**To create an image repository in Amazon ECR**

1. Go to your Dev Environment.

2. Create an empty repository in Amazon ECR:

   ```bash
   aws ecr create-repository --repository-name codecatalyst-eks-image-repo
   ```

   Replace `codecatalyst-eks-image-repo` with the name you want to give the Amazon ECR repository.

   This tutorial assumes you named your repository codecatalyst-eks-image-repo.

3. Display the Amazon ECR repository's details:

   ```bash
   aws ecr describe-repositories --repository-names codecatalyst-eks-image-repo
   ```

   4. Note the “repositoryUri”: value, for example, 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo.
Step 4: Add source files

In this section, you add application source files to your source repository (codecatalyst-eks-source-repository). They consist of:

- An `index.html` file – Displays a 'Hello, World!' message in the browser.
- A `Dockerfile` – Describes the base image to use for your Docker image and the Docker commands to apply to it.
- A `deployment.yaml` file – The Kubernetes manifest that defines the Kubernetes service and deployment.

The folder structure is as follows:

```
|-- codecatalyst-eks-source-repository
    |-- Kubernetes
        |-- deployment.yaml
    |-- public-html
        |-- index.html
    |-- Dockerfile
```

Topics

- `index.html` (p. 214)
- `Dockerfile` (p. 215)
- `deployment.yaml` (p. 215)

index.html

The `index.html` file displays a 'Hello, World!' message in the browser.

To add the `index.html` file

1. Go to your Dev Environment.
2. In `codecatalyst-eks-source-repository`, create a folder called `public-html`.
3. In `/public-html`, create a file called `index.html` with the following contents:

   ```html
   <html>
   <head>
   <title>Hello World</title>
   <style>
   body {
   background-color: black;
   text-align: center;
   color: white;
   font-family: Arial, Helvetica, sans-serif;
   }
   </style>
   </head>
   <body>
   <h1>Hello, World!</h1>
   </body>
   </html>
   ```
4. At the terminal prompt, enter:
cd /projects/codecatalyst-eks-source-repository

5. Add, commit, and push:

```bash
    git add .
    git commit -m "add public-html/index.html"
    git push
```

The index.html is added to your repository in a public-html folder.

**Dockerfile**

The Dockerfile describes the base Docker image to use and the Docker commands to apply to it. For more information about the Dockerfile, see the [Dockerfile Reference](#).

The Dockerfile specified here indicates to use the Apache 2.4 base image (`httpd`). It also includes instructions for copying a source file called `index.html` to a folder on the Apache server that serves webpages. The EXPOSE instruction in the Dockerfile tells Docker that the container is listening on port 80.

**To add the Dockerfile**

1. In codecatalyst-eks-source-repository, create a file called Dockerfile with the following contents:

```bash
    FROM httpd:2.4
    EXPOSE 80
```

Do not include a file extension.

**Important**
The Dockerfile must reside in your repository's root folder. The workflow's Docker build command expects it to be there.

2. Add, commit, and push:

```bash
    git add .
    git commit -m "add Dockerfile"
    git push
```

The Dockerfile is added to your repository.

**deployment.yaml**

In this section, you add a deployment.yaml file to your repository. The deployment.yaml file is a Kubernetes manifest that defines two Kubernetes resource types or *kinds* to run: a 'service' and a 'deployment'.

- The 'service' deploys a load balancer into Amazon EC2. The load balancer provides you with an Internet-facing public URL and standard port (port 80) that you can use to browse to the 'Hello, World!' application.
- The 'deployment' deploys three pods, and each pod will contain a Docker container with the 'Hello, World!' application. The three pods are deployed onto the nodes that were created when you created the cluster.
The manifest in this tutorial is short; however, a manifest can include any number of Kubernetes resource types, such as pods, jobs, ingresses, and network policies. Further, you can use multiple manifest files if your deployment is complex.

**To add a deployment.yaml file**

1. In `codecatalyst-eks-source-repository`, create a folder called `Kubernetes`.
2. In `/Kubernetes`, create a file called `deployment.yaml` with the following contents:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
    app: my-app
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: codecatalyst-eks-container
        # The $REPOSITORY_URI and $IMAGE_TAG placeholders will be replaced by actual values supplied by the build action in your workflow
        image: $REPOSITORY_URI:$IMAGE_TAG
        ports:
        - containerPort: 80
```

3. Add, commit, and push:

```bash
git add .
git commit -m "add Kubernetes/deployment.yaml"
git push
```

The `deployment.yaml` file is added to your repository in a folder called `Kubernetes`.

You have now added all your source files.

Take a moment to double-check your work and make sure you placed all the files in the correct folders. The folder structure is as follows:
Step 5: Create AWS roles

In this section, you create AWS IAM roles that your CodeCatalyst workflow will need in order to function. These roles are:

- **Build role** – Grants the CodeCatalyst build action (in the workflow) permission to access your AWS account and write to Amazon ECR and Amazon EC2.
- **Deploy role** – Grants the CodeCatalyst **Deploy to Kubernetes cluster** action (in the workflow) permission to access your AWS account and Amazon EKS.

For more information about IAM roles, see [IAM roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html) in the *AWS Identity and Access Management User Guide*.

**Note**

To save time, you can create a single role, called the `CodeCatalystWorkflowDevelopmentRole-spaceName` role, instead of the two roles listed previously. For more information, see [Creating the `CodeCatalystWorkflowDevelopmentRole-spaceName` role for your account and space](https://docs.aws.amazon.com/codecatalyst/latest/userguide/tutorial-deploy-to-amazon-eks.html#tutorial-deploy-to-amazon-eks-1) (p. 517). Understand that the `CodeCatalystWorkflowDevelopmentRole-spaceName` role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern. This tutorial assumes you are creating the two roles listed previously.

To create the build and deploy roles, complete the following series of procedures.

1. To create a trust policy for both roles

   1. Go to your Dev Environment.
   2. In the `Cloud9-long-string` directory, create a file called `codecatalyst-eks-trust-policy.json` with the following contents:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "codecatalyst-runner.amazonaws.com",
               "codecatalyst.amazonaws.com"
            ]
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```
2. To create the build policy for the build role

- In the Cloud9-\textit{long-string} directory, create a file called codecatalyst-eks-build-policy.json with the following contents:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

\textbf{Note}  
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

3. To create the deploy policy for the deploy role

- In the Cloud9-\textit{long-string} directory, create a file called codecatalyst-eks-deploy-policy.json with the following contents:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

\textbf{Note}  
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

You have now added three policy documents to your Dev Environment. Your directory structure now looks like this:

|-- Cloud9-\textit{long-string}
4. To add the build policy to AWS

1. In the Dev Environment terminal, enter:

   ```bash
cd /projects
   ``

2. Enter:

   ```bash
   aws iam create-policy \
   --policy-name codecatalyst-eks-build-policy \
   --policy-document file://codecatalyst-eks-build-policy.json
   ``

3. Press Enter.

4. In the command output, note the "arn": value, for example, 
   ```bash
   arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy
   ```
   You need this ARN later.

5. To add the deploy policy to AWS

1. Enter:

   ```bash
   aws iam create-policy \
   --policy-name codecatalyst-eks-deploy-policy \
   --policy-document file://codecatalyst-eks-deploy-policy.json
   ``

2. Press Enter.

3. In the command output, note the deploy policy's "arn": value, for example, 
   ```bash
   arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy
   ```
   You need this ARN later.

6. To create the build role

1. Enter:

   ```bash
   aws iam create-role \
   --role-name codecatalyst-eks-build-role \
   --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
   ``

2. Press Enter.

3. Enter:

   ```bash
   aws iam attach-role-policy \
   --role-name codecatalyst-eks-build-role \
   --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy
   ``

   Where `arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy` is replaced with the ARN of the build policy you noted earlier.
4. Press Enter.
5. At the terminal prompt, enter:

   ```bash
   aws iam get-role --role-name codecatalyst-eks-build-role
   ```

6. Press Enter.
7. Note the role's "Arn": value, for example, `arn:aws:iam::111122223333:role/codecatalyst-eks-build-role`. You need this ARN later.

### 7. To create the deploy role

1. Enter:

   ```bash
   aws iam create-role --role-name codecatalyst-eks-deploy-role --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
   ```

2. Press Enter.
3. Enter:

   ```bash
   aws iam attach-role-policy --role-name codecatalyst-eks-deploy-role --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy
   ```

   Where `arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy` is replaced with the ARN of the deploy policy you noted earlier.

4. Press Enter.
5. Enter:

   ```bash
   aws iam get-role --role-name codecatalyst-eks-deploy-role
   ```

6. Press Enter.
7. Note the role's "Arn": value, for example, `arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role`. You need this ARN later.

You have now created build and deploy roles and noted their ARNs.

### Step 6: Add AWS roles to CodeCatalyst

In this step, you add the build role (`codecatalyst-eks-build-role`) and deploy role (`codecatalyst-eks-deploy-role`) to the AWS account that you connected to your space. This makes the roles available for use in your workflow.

#### To add build and deploy roles to your AWS account

1. In the CodeCatalyst console, navigate to your space.
2. At the top, choose **Settings**.
3. In the navigation pane, choose **AWS accounts**. A list of accounts appears.
4. In the **Amazon CodeCatalyst display name** column, copy the display name of the AWS account where you created your build and deploy roles. (It might be a number.) You'll need this value later, when creating your workflow.
5. Choose the display name.
6. Choose **Manage roles from AWS management console.**

   The *Add IAM role to Amazon CodeCatalyst space* page appears. You might need to sign in to access the page.

7. Select **Add an existing role you have created in IAM.**

   A drop-down list appears. The list displays the build and deploy roles, and any other IAM roles with a trust policy that includes the `codecatalyst-runner.amazonaws.com` and `codecatalyst.amazonaws.com` service principals.

8. From the drop-down list, add:

   • `codecatalyst-eks-build-role`
   • `codecatalyst-eks-deploy-role`

   **Note**

   If you see *The security token included in the request is invalid*, it might be because you do not have the right permissions. To fix this issue, sign out of AWS as sign back in with the AWS account that you used when you created your CodeCatalyst space.

9. Return to the CodeCatalyst console and refresh the page.

   The build and deploy roles should now appear under **IAM roles**.

   These roles are now available for use in CodeCatalyst workflows.

**Step 7: Update the ConfigMap**

You must add the deploy role that you created in Step 5: Create AWS roles (p. 217) to the Kubernetes ConfigMap file to give the **Deploy to Kubernetes cluster** action (in your workflow) the ability to access and interact with your cluster. You can use eksctl or kubectl to perform this task.

**To configure the Kubernetes ConfigMap file using eksctl**

- In the Dev Environment terminal, enter:

  ```bash
  ```

  Where:

  • `codecatalyst-eks-cluster` is replaced with the cluster name of the Amazon EKS cluster.
  • `arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role` is replaced with the ARN of the deploy role that you created in Step 5: Create AWS roles (p. 217).
  • `codecatalyst-eks-deploy-role` (next to --username) is replaced with the name of the deploy role that you created in Step 5: Create AWS roles (p. 217).

  **Note**

  If you decided not to create a deploy role, replace `codecatalyst-eks-deploy-role` with the name of the CodeCatalystWorkflowDevelopmentRole*spaceName* role. For more information about this role, see Step 5: Create AWS roles (p. 217).

  • `us-west-2` is replaced with your Region.

  For details on this command, see Manage IAM users and roles.
A message similar to the following appears:

```
2023-06-09 00:58:29 [#]  checking arn arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role against entries in the auth ConfigMap
2023-06-09 00:58:29 [#]  adding identity "arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role" to auth ConfigMap
```

To configure the Kubernetes ConfigMap file using kubectl

1. In the Dev Environment terminal, enter:

   ```
kubectl edit configmap -n kube-system aws-auth
   
   The ConfigMap file appears on the screen.
   
   2. Add the text in red italics:
   ```

   ```
   # Please edit the object below. Lines beginning with a '#' will be ignored,
   # and an empty file will abort the edit. If an error occurs while saving this file will
   # be
   # reopened with the relevant failures.
   #
   apiVersion: v1
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: arn:aws:iam::111122223333:role/eksctl-codecatalyst-eks-cluster-n-
      NodeInstanceRole-16BC456ME6YR5
      username: system:node:{{EC2PrivateDNSName}}
  - groups:
    - system:masters
    rolearn: arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role
      username: codecatalyst-eks-deploy-role
  mapUsers: |
  
  kind: ConfigMap
metadata:
  creationTimestamp: "2023-06-08T19:04:39Z"
  managedFields:
```

Where:

- `arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role` is replaced with the ARN of the deploy role that you created in Step 5: Create AWS roles (p. 217).
- `codecatalyst-eks-deploy-role` (next to `username:`) is replaced with the name of the deploy role that you created in Step 5: Create AWS roles (p. 217).

**Note**

If you decided not to create a deploy role, replace `codecatalyst-eks-deploy-role` with the name of the CodeCatalystWorkflowDevelopmentRole-`spaceName` role. For more information about this role, see Step 5: Create AWS roles (p. 217).

For details, see Enabling IAM principal access to your cluster in the Amazon EKS User Guide.
You have now given the deploy role, and by extension the **Deploy to Amazon EKS** action, `system:masters` permissions to your Kubernetes cluster.

**Step 8: Create and run a workflow**

In this step, you create a workflow that takes your source files, builds them into a Docker image, and then deploys the image into tree pods in your Amazon EKS cluster.

The workflow consists of the following building blocks that run sequentially:

- A trigger – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see [Working with triggers](#).
- A build action (BuildBackend) – On trigger, the action builds the Docker image using the Dockerfile and pushes the image to Amazon ECR. The build action also updates the `$REPOSITORY_URI` and `$IMAGE_TAG` variables in the deployment.yaml file with the correct values, and then creates an output artifact of this file and any others in the Kubernetes folder. In this tutorial, the only file in the Kubernetes folder is deployment.yaml but you could include more files. The artifact is used as the input for the deploy action, which is next.

  For more information about the build action, see [Building using workflows in CodeCatalyst](#).

- A deploy action (DeployToEKS) – On completion of the build action, the deploy action looks for the output artifact generated by the build action (Manifests), and finds the deployment.yaml file inside of it. The action then follows the instructions in the deployment.yaml file to run three pods—each containing a single 'Hello, World!' Docker container—inside your Amazon EKS cluster.

To create a workflow

1. Go to the CodeCatalyst console.
2. Navigate to your project (codecatalyst-eks-project).
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose Create workflow.
5. For Source repository, choose codecatalyst-eks-source-repository.
6. For Branch, choose main.
7. Choose Create.
8. Delete the YAML sample code.
9. Add the following YAML code to create a new workflow definition file:

   ```yaml
   Name: codecatalyst-eks-workflow
   SchemaVersion: 1.0

   Triggers:
   - Type: PUSH
     Branches:
     - main
   Actions:
   BuildBackend:
     Identifier: aws/build@v1
   Environment:
     - Name: codecatalyst-eks-environment
   Connections:
     - Name: codecatalyst-account-connection
   Role: codecatalyst-eks-build-role
   Inputs:
   ```

   For more information about the workflow definition file, see [Workflow definition reference](#).
Sources:
- WorkflowSource

Variables:
- Name: REPOSITORY_URI
  Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo
- Name: IMAGE_TAG
  Value: ${WorkflowSource.CommitId}

Configuration:
Steps:
  #pre_build:
  - Run: echo Logging in to Amazon ECR...
  - Run: aws --version
  - Run: aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
  #build:
  - Run: echo Build started on `date`
  - Run: echo Building the Docker image...
  - Run: docker build -t $REPOSITORY_URI:latest .
  - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  #post_build:
  - Run: echo Build completed on `date`
  - Run: echo Pushing the Docker images...
     - Run: docker push $REPOSITORY_URI:latest
     - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
  # Replace the variables in deployment.yaml
  - Run: find Kubernetes/ -type f | xargs sed -i "s|\$REPOSITORY_URI|\$REPOSITORY_URI\|g"
  - Run: find Kubernetes/ -type f | xargs sed -i "s|\$IMAGE_TAG|\$IMAGE_TAG\|g"

In the preceding code, replace:

- Both instances of ${codecatlyst-eks-environment} with the name of the environment you created in Prerequisites (p. 207).
- Both instances of ${codecatlyst-account-connection} with the display name of your account connection. The display name might be a number. For more information, see Step 6: Add AWS roles to CodeCatalyst (p. 220).
• `codecatalyst-eks-build-role` with the name of the build role you created in Step 5: Create AWS roles (p. 217).

• `111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo` (in the Value: property) with the URI of the Amazon ECR repository you created in Step 3: Create an Amazon ECR image repository (p. 213).

• `111122223333.dkr.ecr.us-west-2.amazonaws.com` (in the Run: `aws ecr` command) with the URI of the Amazon ECR repository without the image suffix (`/codecatalyst-eks-image-repo`).

• `codecatalyst-eks-deploy-role` with the name of the deploy role you created in Step 5: Create AWS roles (p. 217).

• Both instances of `us-west-2` with your AWS Region code. For a list of Region codes, see Regional endpoints in the AWS General Reference.

Note
If you decided not to create build and deploy roles, replace `codecatalyst-eks-build-role` and `codecatalyst-eks-deploy-role` with the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role. For more information about this role, see Step 5: Create AWS roles (p. 217).

10. (Optional) Choose Validate to make sure that the YAML code is valid before committing.

11. Choose Commit.

12. In the Commit workflow dialog box, enter the following:

   a. For Commit message, remove the text and enter:

   ```
   Add first workflow
   ```

   b. For Repository, choose codecatalyst-eks-source-repository.
   
   c. For Branch name, choose main.
   
   d. Choose Commit.

   You have now created a workflow. A workflow run starts automatically because of the trigger defined at the top of the workflow. Specifically, when you committed (and pushed) the workflow.yaml file to your source repository, the trigger started the workflow run.

To view the workflow run progress

1. In the navigation pane of the CodeCatalyst console, choose CI/CD, and then choose Workflows.
2. Choose the workflow you just created, codecatalyst-eks-workflow.
3. Choose BuildBackend to see the build progress.
4. Choose DeployToEKS to see the deployment progress.

   For more information about viewing run details, see Viewing workflow run status and details (p. 315).

To verify the deployment

1. Open the Amazon EC2 console at `https://console.aws.amazon.com/ec2/`.
2. On the left, near the bottom, choose Load Balancers.
3. Select the load balancer that was created as part of your Kubernetes deployment. If you're not sure which load balancer to choose, look for the following tags under the Tags tab:
• kubernetes.io/service-name
• kubernetes.io/cluster/ekstutorialcluster
4. With the correct load balancer selected, choose the Description tab.
5. Copy and paste the DNS name value into your browser's address bar.

The 'Hello, World!' webpage appears in your browser, indicating that you successfully deployed your application.

Step 9: Make a change to your source files

In this section, you make a change to the index.html file in your source repository. This change causes the workflow to build a new Docker image, tag it with a commit ID, push it to Amazon ECR, and deploy it to Amazon ECS.

To change the index.html
1. Go to your Dev Environment.
2. At the terminal prompt, change to your source repository:
   cd /projects/codecatalyst-eks-source-repository
3. Pull the latest workflow changes:
   git pull
5. On line 14, change the Hello, World! text to Tutorial complete!.
6. Add, commit, and push:
   git add .
   git commit -m "update index.html title"
   git push

   A workflow run starts automatically.
7. (Optional) Enter:
   git show HEAD

   Note the commit ID for the index.html change. This commit ID will be tagged to the Docker image that will be deployed by the workflow run that you just started.
8. Watch the deployment progress:
   a. In the CodeCatalyst console, in the navigation pane, choose CI/CD, and then choose Workflows.
   b. Choose codecatalyst-eks-workflow to view the latest run.
   c. Choose BuildBackend, and DeployToEKS to see the workflow run progress.
9. Verify that your application was updated, as follows:
   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. On the left, near the bottom, choose Load Balancers.
   c. Select the load balancer that was created as part of your Kubernetes deployment.
   d. Copy and paste the DNS name value into your browser's address bar.
The 'Tutorial Complete!' webpage appears in your browser, indicating that you successfully deployed a new revision of your application.

10. (Optional) In AWS, switch to the Amazon ECR console and verify that the new Docker image was tagged with the commit ID from step 7 of this procedure.

**Clean up**

You should clean up your environment so that you're not charged unnecessarily for the storage and compute resources used by this tutorial.

**To clean up**

1. Delete your cluster:
   - In the Dev Environment terminal, enter:
     ```
     eksctl delete cluster --region=us-west-2 --name=codecatalyst-eks-cluster
     ```
   Where:
   - `us-west-2` is replaced with your Region.
   - `codecatalyst-eks-cluster` is replaced with the name of the cluster you created.

   After 5-10 minutes, the cluster and associated resources are deleted, including but not limited to AWS CloudFormation stacks, nodes groups (in Amazon EC2), and load balancers.

   **Important**
   If the `eksctl delete cluster` command doesn't work, you may need to refresh your AWS credentials or your `kubectl` credentials. If you're not sure which credentials to refresh, refresh the AWS credentials first. To refresh your AWS credentials, see *How do I fix "Unable to locate credentials" and "ExpiredToken" errors? (p. 589)*. To refresh your `kubectl` credentials, see *How do I fix "Unable to connect to the server" errors? (p. 590)*.

2. In the AWS console, clean up as follows:
   1. In Amazon ECR, delete `codecatalyst-eks-image-repo`.
   2. In IAM Identity Center, delete:
      a. `codecatalyst-eks-user`
      b. `codecatalyst-eks-permission-set`
   3. In IAM, delete:
      - `codecatalyst-eks-build-role`
      - `codecatalyst-eks-deploy-role`
      - `codecatalyst-eks-build-policy`
      - `codecatalyst-eks-deploy-policy`

3. In the CodeCatalyst console, clean up as follows:
   1. Delete `codecatalyst-eks-workflow`.
   2. Delete `codecatalyst-eks-environment`.
   3. Delete `codecatalyst-eks-source-repository`.
   5. Delete `codecatalyst-eks-project`. 
In this tutorial, you learned how to deploy an application to an Amazon EKS service using a CodeCatalyst workflow and a **Deploy to Kubernetes cluster** action.

### Adding the "Deploy AWS CloudFormation stack" action

**Tip**  
For a tutorial that shows you how to use the **Deploy AWS CloudFormation stack** action, see [Tutorial: Deploy a serverless application using AWS CloudFormation](p. 165).

This section describes how to add the **Deploy AWS CloudFormation stack** action to your workflow. The action creates a CloudFormation stack in AWS based on a template that you provide. The template can be a:

- AWS CloudFormation template – For more information, see [Working with AWS CloudFormation templates](#).
- AWS SAM template – For more information, see [AWS Serverless Application Model (AWS SAM) specification](#).

**Note**  
To use a AWS SAM template, you must first package your AWS SAM application using the `sam package` operation. For a tutorial that shows you how to do this packaging automatically as part of a Amazon CodeCatalyst workflow, see [Tutorial: Deploy a serverless application using AWS CloudFormation](p. 165).

If the stack already exists, the action runs the CloudFormation `CreateChangeSet` operation, and then the `ExecuteChangeSet` operation. The action then waits for the changes to be deployed and marks itself as either succeeded for failed, depending on the results.

Use the **Deploy AWS CloudFormation stack** action if you already have an AWS CloudFormation or AWS SAM template that contains resources you’d like to deploy, or you plan on generating one automatically as part of a workflow [build action (p. 139)](#) using tools like AWS SAM and [AWS Cloud Development Kit (AWS CDK)](#).

There are no restrictions on the template you can use—whatever you can author in CloudFormation or AWS SAM you can use with the **Deploy AWS CloudFormation stack** action.

### Visual

**To add the "Deploy AWS CloudFormation stack" action using the visual editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **Deploy AWS CloudFormation stack** action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
Adding the "Deploy AWS CloudFormation stack" action

Or

- Choose **Deploy AWS CloudFormation stack**. The action details dialog box appears. On this
dialog box:
  - (Optional) Choose **Download** to view the action's source code (p.____).
  - Choose **Add to workflow** to add the action to the workflow diagram and open its
    configuration pane.

10. In the **Inputs** and **Configuration** tabs, complete the fields according to your needs.
    For a description of each field, see the "Deploy AWS CloudFormation stack" action
    reference (p. 411). This reference provides detailed information about each field (and
    corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

YAML

To add the "Deploy AWS CloudFormation stack" action using the YAML editor

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name
   where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **Deploy AWS CloudFormation stack** action, and do one of the following:

   - Choose the plus sign (+) to add the action to the workflow diagram and open its
     configuration pane.

   Or

   - Choose **Deploy AWS CloudFormation stack**. The action details dialog box appears. On this
dialog box:
     - (Optional) Choose **Download** to view the action's source code (p.____).
     - Choose **Add to workflow** to add the action to the workflow diagram and open its
       configuration pane.

10. Modify the properties in the YAML code according to your needs. An explanation of
    each available property is provided in the "Deploy AWS CloudFormation stack" action
    reference (p. 411).
11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
12. Choose **Commit**, enter a commit message, and choose **Commit** again.

Variables produced by the "Deploy AWS CloudFormation stack" action

When the **Deploy AWS CloudFormation stack** action runs, it produces variables which you can
use in subsequent workflow actions. For details, see "Deploy AWS CloudFormation stack" action
variables (p. 349) in the List of predefined variables (p. 346).
"Deploy AWS CloudFormation stack" action definition

The **Deploy AWS CloudFormation stack** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "Deploy AWS CloudFormation stack" action reference (p. 411) in the Workflow definition reference (p. 354).

Adding the "Deploy to Amazon ECS" action

Tip
For a tutorial that shows you how to use the **Deploy to Amazon ECS** action, see Tutorial: Deploy an application to Amazon ECS (p. 183).

Tip
For a working example of the **Deploy to Amazon ECS** action, create a project with either the Node.js API with AWS Fargate or Java API with AWS Fargate blueprint. For more information, see Creating a project with a blueprint (p. 58).

This section describes how to add the **Deploy to Amazon ECS** action to your workflow. This action registers a **task definition** file that you provide. Upon registration, the task definition is then instantiated by your **Amazon ECS service** running in your **Amazon ECS cluster**. "Instantiating a task definition" is equivalent to deploying an application into Amazon ECS.

To use this action, you must have an existing Amazon ECS cluster, service, and task definition file ready.

For more information about Amazon ECS, see the Amazon Elastic Container Service Developer Guide.

Visual

To add the "Deploy to Amazon ECS" action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. At the top-left, choose + **Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **Deploy to Amazon ECS** action, and do one of the following:

   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.

   Or

   - Choose **Deploy to Amazon ECS**. The action details dialog box appears. On this dialog box:

       - (Optional) Choose **Download** to view the action’s source code (p. _____).

       - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. In the **Inputs** and **Configuration** tabs, complete the fields according to your needs. For a description of each field, see the "Deploy to Amazon ECS" action reference (p. 424). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow’s YAML code before committing.
12. Choose Commit, enter a commit message, and choose Commit again.

YAML

To add the "Deploy to Amazon ECS" action using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the Deploy to Amazon ECS action, and do one of the following:
   • Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   Or
   • Choose Deploy to Amazon ECS. The action details dialog box appears. On this dialog box:
     • (Optional) Choose Download to view the action's source code.
     • Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.
10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "Deploy to Amazon ECS" action reference (p. 424).
11. (Optional) Choose Validate to validate the workflow's YAML code before committing.
12. Choose Commit, enter a commit message, and choose Commit again.

Variables produced by the "Deploy to Amazon ECS" action

When the Deploy to Amazon ECS action runs, it produces variables which you can use in subsequent workflow actions. For details, see "Deploy to Amazon ECS" action variables (p. 350) in the List of predefined variables (p. 346).

"Deploy to Amazon ECS" action definition

The Deploy to Amazon ECS action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "Deploy to Amazon ECS" action reference (p. 424) in the Workflow definition reference (p. 354).

Adding the "Deploy to Kubernetes cluster" action

Tip
For a tutorial that shows you how to use the Deploy to Kubernetes cluster action, see Tutorial: Deploy an application to Amazon EKS (p. 207).

This section describes how to add the Deploy to Kubernetes cluster action to your workflow. This action deploys your application to a Kubernetes cluster that you have set up in Amazon Elastic Kubernetes
Adding the "Deploy to Kubernetes cluster" action

Service (EKS) using one or more Kubernetes manifest files that you provide. For a sample manifest, see deployment.yaml (p. 215) in Tutorial: Deploy an application to Amazon EKS (p. 207).

For more information about Kubernetes, see the Kubernetes Documentation.

For more information about Amazon EKS, see What is Amazon EKS? in the Amazon EKS User Guide.

How it works

The Deploy to Kubernetes cluster works as follows:

1. At runtime, the action installs the Kubernetes kubectl utility to the CodeCatalyst compute machine where the action is running. The action configures kubectl to point to the Amazon EKS cluster you provided when you configured the action. The kubectl utility is necessary to run the kubectl apply command, next.

2. The action runs the kubectl apply -f my-manifest.yaml command, which carries out the instructions in my-manifest.yaml to deploy your application as a set of containers and pods into the configured cluster. For more information on this command, see the kubectl apply topic in the Kubernetes Reference Documentation.

Prerequisites

To use this action, you must have the following prepared:

Tip
To set up these prerequisites quickly, follow the instructions in Tutorial: Deploy an application to Amazon EKS (p. 207).

• A Kubernetes cluster in Amazon EKS. For information about clusters, see Amazon EKS clusters in the Amazon EKS User Guide.

• At least one Dockerfile that describes how to assemble your application into a Docker image. For more information about Dockerfiles, see the Dockerfile reference.

• At least one Kubernetes manifest file, which is called a configuration file or configuration in the Kubernetes documentation. For more information, see Managing resources in the Kubernetes documentation.

• An IAM role that gives the Deploy to Kubernetes cluster action the ability to access and interact with your Amazon EKS cluster. For more information, see the Role (p. 436) topic in the "Deploy to Kubernetes cluster" action reference (p. 432).

After creating this role, you must add it to:

• Your Kubernetes ConfigMap file. To learn how to add a role to a ConfigMap file, see Enabling IAM principal access to your cluster in the Amazon EKS User Guide.

• CodeCatalyst. To learn how to add an IAM role to CodeCatalyst, see Adding IAM roles to account connections (p. 40).

• A CodeCatalyst space, project, and environment. The space and environment must both be connected to the AWS account into which you will be deploying your application. For more information, see Creating additional spaces in CodeCatalyst (p. 53), Creating an empty project in Amazon CodeCatalyst (p. 58), and Working with environments (p. 307).

• A source repository supported by CodeCatalyst. The repository stores your application source files, Dockerfiles, and Kubernetes manifests. For more information, see Source repositories in CodeCatalyst (p. 79).

Adding the "Deploy to cluster" action

Use the following instructions to add the Deploy to cluster action to your workflow.
Visual

To add the "Deploy to Kubernetes cluster" action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the Deploy to Kubernetes cluster action, and do one of the following:
   • Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.

   Or
   • Choose Deploy to Kubernetes cluster. The action details dialog box appears. On this dialog box:
     • (Optional) Choose Download to view the action’s source code (p. 432).
     • Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.
10. In the Inputs and Configuration tabs, complete the fields according to your needs. For a description of each field, see the "Deploy to Kubernetes cluster" action reference (p. 432). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.
11. (Optional) Choose Validate to validate the workflow's YAML code before committing.
12. Choose Commit, enter a commit message, and choose Commit again.

YAML

To add the "Deploy to Kubernetes cluster" action using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the Deploy to Kubernetes cluster action, and do one of the following:
   • Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.

   Or
   • Choose Deploy to Kubernetes cluster. The action details dialog box appears. On this dialog box:
Adding the "AWS CDK deploy" action

This section describes how to add the AWS CDK deploy action to your workflow. The AWS CDK deploy action synthesizes and deploys your AWS Cloud Development Kit (AWS CDK) app into AWS. If your app already exists in AWS, the action updates it if necessary.

For general information about writing apps using the AWS CDK, see What is the AWS CDK? in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

Topics

- When to use this action (p. 234)
- How it works (p. 234)
- How many stacks can the action deploy? (p. 235)
- Prerequisites (p. 235)
- Example workflow (p. 235)
- Adding the "AWS CDK deploy" action (p. 236)
- Variables produced by the "AWS CDK deploy" action (p. 237)
- "AWS CDK deploy" action definition (p. 238)

When to use this action

Use this action if you have developed an app using the AWS CDK, and you now want to deploy it automatically as part of automated continuous integration and delivery (CI/CD) workflow. For example, you might want to deploy your AWS CDK app automatically whenever someone merges a pull request related to your AWS CDK app source.

How it works

The AWS CDK deploy works as follows:

1. At runtime, the action installs the AWS CDK Toolkit to the CodeCatalyst build image (p. 302). The toolkit is necessary to run the cdk deploy command, next.
2. The action runs the `cdk deploy` command, which synthesizes and deploys your AWS CDK app into AWS. For more information about this command, see the AWS CDK Toolkit (cdk command) topic in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

How many stacks can the action deploy?

The AWS CDK deploy can deploy a single stack only. If your AWS CDK app consists of multiple stacks, you must create a parent stack with nested stacks, and deploy the parent using this action.

Prerequisites

Before you can use the AWS CDK deploy action, complete the following tasks:

1. **Have an AWS CDK app ready.** You can write your AWS CDK app using AWS CDK v1 or v2, in any programming language supported by the AWS CDK. Make sure your AWS CDK app files are available in:
   - A CodeCatalyst source repository (p. 79), or
   - A CodeCatalyst output artifact (p. 286) generated by another workflow action

2. **Bootstrap your AWS environment.** To bootstrap, you can:
   - Use one of the methods described in How to bootstrap in the AWS Cloud Development Kit (AWS CDK) Developer Guide.
   - Use the AWS CDK bootstrap action. You can add this action in the same workflow as your AWS CDK deploy, or in a different one. Just make sure the bootstrap action runs at least once prior to running the AWS CDK deploy action so that the necessary resources are in place. For more information about the AWS CDK bootstrap action, see Adding the "AWS CDK bootstrap" action (p. 254).

   For more information about bootstrapping, see Bootstrapping in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

Example workflow

The following example workflow includes the AWS CDK deploy action, along with the AWS CDK bootstrap action. The workflow consists of the following building blocks that run sequentially:

   **Note**
   The following workflow example is for illustrative purposes, and will not work without additional configuration. It is meant to give you an example of what a workflow might look like when it is configured with the AWS CDK deploy action.

   - **A trigger** – This trigger starts the workflow run automatically when you push a change to your source repository. This repository contains your AWS CDK app. For more information about triggers, see Working with triggers (p. 327).
   - **An AWS CDK bootstrap action** (CDKBootstrap) – On trigger, the action deploys the CDKToolkit bootstrap stack into AWS. If the CDKToolkit stack already exists in the environment, it will be upgraded if necessary; otherwise, nothing happens, and the action is marked as succeeded.
   - **An AWS CDK deploy action** (AWS CDKDeploy) – On completion of the AWS CDK bootstrap action, the AWS CDK deploy action synthesizes your AWS CDK app code into an AWS CloudFormation template and deploys the stack defined in the template into AWS.

Name: codecatalyst-cdk-deploy-workflow
SchemaVersion: 1.0
Triggers:
Adding the "AWS CDK deploy" action

Use the following instructions to add the AWS CDK deploy action to your workflow.

**Prerequisites**

Before you begin, make sure you have completed the tasks described in Prerequisites (p. 235).

**Visual**

**To add the "AWS CDK deploy" action using the visual editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the AWS CDK deploy action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Or
Adding the "AWS CDK deploy" action

• Choose **AWS CDK deploy**. The action details dialog box appears. On this dialog box:
  • (Optional) Choose **Download** to view the action's source code (p. 592).
  • Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. In the **Inputs** and **Configuration** tabs, complete the fields according to your needs. For a description of each field, see the "AWS CDK deploy" action reference (p. 393). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and then choose **Commit** again.

  **Note**
  If your **AWS CDK deploy** action fails with an npm `install` error, see [How do I fix "npm install" errors? (p. 592)](https://codecatalyst.aws/) for information about how to fix the error.

YAML

**To add the "AWS CDK deploy" action using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **AWS CDK deploy** action, and do one of the following:

   • Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   
   Or
   
   • Choose **AWS CDK deploy**. The action details dialog box appears. On this dialog box:
     • (Optional) Choose **Download** to view the action's source code (p. 592).
     • Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "AWS CDK deploy" action reference (p. 393).

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and then choose **Commit** again.

  **Note**
  If your **AWS CDK deploy** action fails with an npm `install` error, see [How do I fix "npm install" errors? (p. 592)](https://codecatalyst.aws/) for information about how to fix the error.

**Variables produced by the "AWS CDK deploy" action**

When the **AWS CDK deploy** action runs, it produces variables which you can use in subsequent workflow actions. For details, see "AWS CDK deploy" action variables (p. 348) in the List of predefined variables (p. 346).
"AWS CDK deploy" action definition

The **AWS CDK deploy** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "**AWS CDK deploy** action reference (p. 393)" in the **Workflow definition reference (p. 354)**.

Working with deployments

This section describes how to manage, monitor, and configure deployments in Amazon CodeCatalyst.

Topics

- Deploying an application or resource (p. 238)
- Rerunning a deployment (p. 238)
- Viewing deployment status, commits, and pull requests (p. 238)
- Viewing the deployment logs (p. 239)
- Configuring rollbacks (p. 240)
- Surfacing the URL of the deployed application (p. 242)
- Removing a deployment target (p. 245)

Deploying an application or resource

To deploy an application or resource using Amazon CodeCatalyst, you must use a CodeCatalyst workflow. This workflow must include at least one action that can deploy something. For example, if you wanted to deploy an AWS CloudFormation stack, you would create a workflow that included the **Deploy AWS CloudFormation stack** action.

For more information about workflows, see **Build, test, and deploy with workflows in CodeCatalyst (p. 128)**. For more information about actions, see **Working with actions (p. 249)**.

Rerunning a deployment

To rerun a deployment in Amazon CodeCatalyst, you rerun its associated workflow. For instructions, see **Starting a workflow run (p. 314)**.

Viewing deployment status, commits, and pull requests

You can view the following information about a deployment in Amazon CodeCatalyst:

- Deployment activity, including the deployment status, start time, end time, history, and duration of events.
- Stack name, AWS Region, last update time, and associated workflows.
- Commits and pull requests.
- Action-specific information, for example, CloudFormation events and outputs.

You can view deployment information starting from a **workflow (p. 128)**, an **environment (p. 307)**, or a workflow **action (p. 132)**.

To view deployment information starting from a workflow

- Go to the workflow run that deployed your application. For instructions, see **Viewing workflow run status and details (p. 315)**.
To view deployment information starting from an environment

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Environments.
4. Choose the environment where your stack was deployed, for example, Production.
5. Choose Deployment activity to view the deployment history of your stacks, the status of the deployments (for example, SUCCEEDED or FAILED), and other deployment-related information.
6. Choose Deployment target to view information about the stacks, clusters, or other targets deployed into the environment. You can view information such as the stack name, Region, provider, and identifier.

To view deployment information starting from an action

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. In the workflow diagram, choose the workflow action that deployed your application. For example, you might choose DeployCloudFormationStack.
6. Review the contents in the right pane for action-specific deployment information.

Viewing the deployment logs

You can view logs related to specific deploy actions to troubleshoot problems in Amazon CodeCatalyst.

You can view logs starting from a workflow (p. 128), or an environment (p. 307).

To view the logs of a deploy action starting from a workflow

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Runs.
6. Choose the workflow run that deployed your application.
7. In the workflow diagram, choose the action whose logs you want to view.
8. Choose the Logs tab and expand the sections to reveal the log messages.
9. To view more logs, choose the Summary tab, and then choose View in CloudFormation (if it's available) to view more logs there. You may need to sign in to AWS.

To view the logs of a deploy action starting from an environment

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Environments.
4. Choose the environment into which your application was deployed.
5. In Deployment activity, find the Workflow Run ID column, and choose the workflow run that deployed your stack.
6. In the workflow diagram, choose the action whose logs you want to view.
7. Choose the Logs tab and expand the sections to reveal the log messages.
8. To view more logs, choose the Summary tab, and then choose View in CloudFormation (if it's available) to view more logs there. You may need to sign in to AWS.

### Configuring rollbacks

By default, if the Deploy AWS CloudFormation stack action fails, it will cause AWS CloudFormation to roll back the stack to the last known stable state. You can change the behavior so that rollbacks occur not only when the action fails, but also when a specified Amazon CloudWatch alarm occurs. For more information about CloudWatch alarms, see Using Amazon CloudWatch alarms in the Amazon CloudWatch User Guide.

You can also change the default behavior so that CloudFormation does not roll back the stack when the action fails.

Use the following instructions to configure rollbacks.

**Note**

You cannot start a rollback manually.

**Visual**

### Before you begin

1. Make sure you have a workflow (p. 246) that includes a functioning Deploy AWS CloudFormation stack action. For more information, see Adding the "Deploy AWS CloudFormation stack" action (p. 228).

2. In the role specified in the Stack role - optional field of the Deploy AWS CloudFormation stack action, make sure to include the CloudWatchFullAccess permission. For information about creating this role with the appropriate permissions, see Step 2: Create AWS roles (p. 167).

### To configure rollback alarms for the "Deploy AWS CloudFormation stack" action

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of a workflow that includes the Deploy AWS CloudFormation stack action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. Choose your Deploy AWS CloudFormation stack action.
8. In the details pane, choose Configuration.
9. At the bottom, expand Advanced.
10. Under Monitor alarm ARNs, choose Add alarm.
11. Enter information into the following fields.
   - Alarm ARN
Specify the Amazon Resource Name (ARN) of an Amazon CloudWatch alarm to use as a rollback trigger. For example, arn:aws:cloudwatch::123456789012:alarm/MyAlarm. You can have a maximum of five rollback triggers.

**Note**
If you specify a CloudWatch alarm ARN, you'll also need to configure additional permissions to enable the action to access CloudWatch. For more information, see Configuring rollbacks (p. 240).

- **Monitoring time**

Specify an amount of time, from 0 to 180 minutes, during which CloudFormation monitors the specified alarms. Monitoring begins after all the stack resources have been deployed. If the alarm occurs within the specified monitoring time, then the deployment fails, and CloudFormation rolls back the entire stack operation.

Default: 0. CloudFormation only monitors alarms while the stack resources are being deployed, not after.

**YAML**

**To configure rollback triggers for the "Deploy AWS CloudFormation stack" action**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of a workflow that includes the Deploy AWS CloudFormation stack action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. Add the monitor-alarm-arns and monitor-timeout-in-minutes properties in the YAML code to add rollback triggers. For an explanation of each property, see "Deploy AWS CloudFormation stack" action reference (p. 411).
8. In the role specified in the role-arn property of the Deploy AWS CloudFormation stack action, make sure to include the CloudWatchFullAccess permission. For information about creating this role with the appropriate permissions, see Step 2: Create AWS roles (p. 167).

**Visual**

**To turn off rollbacks for the "Deploy AWS CloudFormation stack" action**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of a workflow that includes the Deploy AWS CloudFormation stack action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. Choose your Deploy AWS CloudFormation stack action.
8. In the details pane, choose **Configuration**.
9. At the bottom, expand **Advanced**.
10. Turn on **Disable rollback**.

YAML

**To turn off rollbacks for the "Deploy AWS CloudFormation stack" action**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of a workflow that includes the **Deploy AWS CloudFormation stack** action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. Add the `disable-rollback: 1` property in the YAML code to stop rollbacks. For an explanation of this property, see **"Deploy AWS CloudFormation stack" action reference (p. 411)**.

**Surfacing the URL of the deployed application**

If your workflow deploys an application, you can configure Amazon CodeCatalyst to display the application's URL as a clickable link. This link appears in the CodeCatalyst console, inside the action that deployed it. The following workflow diagram shows the **View App** URL appearing at the bottom of an action.

By making this URL clickable in the CodeCatalyst console, you can quickly verify your application deployment.

**Note**
The app URL is not supported with the **Deploy to Amazon ECS** action.

To enable this feature, add an output variable to your action with a name that contains `appurl` or `endpointurl`. You can use a name with or without a joining dash (-), underscore (_), or space ( ). The string is case-insensitive. Set the variable's value to the `http` or `https` URL of your deployed application.
Note
If you're updating an existing output variable to include the app url, or endpoint url string, update all references to this variable to use the new variable name.

For detailed steps, see one of the following procedures:

- **To display the app URL in the "AWS CDK deploy" action (p. 243)**
- **To display the app URL in the "Deploy AWS CloudFormation stack" action (p. 243)**
- **To display the app URL in all other actions (p. 244)**

When you've finished configuring the URL, verify that it appears as expected by following these instructions:

- **To verify that the application URL was added (p. 245)**

**To display the app URL in the "AWS CDK deploy" action**

1. If you're using the **AWS CDK deploy** action, add a CfnOutput construct (which is a key-value pair) in your AWS CDK application code:
   - The key name must contain appurl, or endpointurl, with or without a joining dash (-), underscore (_), or space (_). The string is case-insensitive.
   - The value must be the http or https URL of your deployed application.

   For example, your AWS CDK code might look like this:

   ```typescript
   import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
   import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
   import * as s3 from 'aws-cdk-lib/aws-s3';
   import { Construct } from 'constructs';
   import * as cdk from 'aws-cdk-lib';
   export class HelloCdkStack extends Stack {
     constructor(scope: Construct, id: string, props?: StackProps) {
       super(scope, id, props);
       const bucket = new s3.Bucket(this, 'my-bucket', {
         removalPolicy: RemovalPolicy.DESTROY,  
       });
       new CfnOutput(this, 'APP-URL', {
         value: https://mycompany.myapp.com,
         description: 'The URL of the deployed application',
         exportName: 'myApp',
       });
       ...
     }
   }
   ```

   For more information about the CfnOutput construct, see **interface CfnOutputProps** in the **AWS Cloud Development Kit (AWS CDK) API Reference**.

2. Save and commit your code.

3. Proceed to **To verify that the application URL was added (p. 245)**.

**To display the app URL in the "Deploy AWS CloudFormation stack" action**

1. If you're using the **Deploy AWS CloudFormation stack** action, add an output to the Outputs section in your CloudFormation template or AWS SAM template with these characteristics:
• The key (also called the logical ID) must contain `appurl`, or `endpointurl`, with or without a joining dash (-), underscore (_), or space ( ). The string is case-insensitive.

• The value must be the http or https URL of your deployed application.

For example, your CloudFormation template might look like this:

```
"Outputs" : {
    "APP-URL" : {
        "Description" : "The URL of the deployed app",
        "Value" : "https://mycompany.myapp.com",
        "Export" : {
            "Name" : "My App"
        }
    }
}
```

For more information about CloudFormation outputs, see Outputs in the AWS CloudFormation User Guide.

2. Save and commit your code.
3. Proceed to To verify that the application URL was added (p. 245).

To display the app URL in all other actions

If you're using another action to deploy your application, such as the build action or GitHub Actions, do the following to have the app URL displayed.

1. Define an environment variable in the Inputs or Steps section of the action in the workflow definition file. The variable must have these characteristics:

   • The name must contain `appurl`, or `endpointurl`, with or without a joining dash (-), underscore (_), or space ( ). The string is case-insensitive.

   • The value must be the http or https URL of your deployed application.

For example, a build action might look like this:

```
Build-action:
  Identifier: aws/build@v1
  Inputs:
    Variables:
      - Name: APP-URL
        Value: https://mycompany.myapp.com
```

...or this:

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: APP-URL=https://mycompany.myapp.com
```

For more information about defining environment variables, see Defining a variable (p. 338).

2. Export the variable.
For example, your build action might look like this:

```yaml
Build-action:
  ...
Outputs:
  Variables:
    - APP-URL
```

For information about exporting variables, see [Exporting a variable so that other actions can use it](#) (p. 341).

3. (Optional) Choose Validate to validate the workflow's YAML code before committing.
4. Choose Commit, enter a commit message, and choose Commit again.
5. Proceed to To verify that the application URL was added (p. 245).

### To verify that the application URL was added

- Start a workflow run, if it hasn't started automatically. The new run should have the app URL displayed as a clickable link in its workflow diagram. For more information about starting runs, see [Starting a workflow run](#) (p. 314).

### Removing a deployment target

You can remove a deployment target such as an Amazon ECS cluster or AWS CloudFormation stack from the **Environments** page in the CodeCatalyst console.

**Important**

When you remove a deployment target, it is removed from the CodeCatalyst console, but remains available in the AWS service that hosts it (if it still exists).

Consider removing a deployment target if the target has become stale in CodeCatalyst. Targets might become stale if:

- You deleted the workflow that deployed to the target.
- You changed the stack or cluster that you're deploying to.
- You deleted the stack or cluster from the CloudFormation or Amazon ECS service in the AWS console.

### To remove a deployment target

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Environments**.
4. Choose the name of the environment that contains the deployment target you want to remove. For information about environments, see [Working with environments](#) (p. 307).
5. Choose the **Deployment targets** tab.
6. Choose the radio button next to the deployment target you want to remove.
7. Choose **Remove**.

The target is removed from the page.
Working with workflows

A workflow is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or actions, to take during a workflow run. A workflow also defines the events, or triggers, that cause the workflow to start. To set up a workflow, you create a workflow definition file using the CodeCatalyst console's visual or YAML editor.

Tip
For a quick look at how you might use workflows in a project, create a project with a blueprint. Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

Topics
- Creating, editing, and deleting a workflow (p. 246)
- Working with actions (p. 249)
- Working with artifacts (p. 286)
- Working with compute and runtime environment Docker images (p. 293)
- Working with environments (p. 307)
- Working with file caching (p. 311)
- Working with runs (p. 314)
- Working with secrets (p. 322)
- Working with sources (p. 325)
- Working with triggers (p. 327)
- Working with variables (p. 337)

Creating, editing, and deleting a workflow

Use the following procedures to create, edit, and delete a workflow.

Visual

To create a workflow using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose Create workflow.

The Create workflow dialog box appears.

5. In the Source repository field, choose a source repository where the workflow definition file will reside. The file will be stored in the ~/.codecatalyst/workflows/ folder in the chosen repository. If no source repository exists, create one (p. 89).
6. In the Branch field, choose a branch where the workflow definition file will reside.
7. Choose Create.

Amazon CodeCatalyst saves the repository and branch information in memory, but the workflow is not yet committed.
9. Build the workflow:
a. (Optional) In the workflow diagram, choose the **Source** and **Triggers** box. A **Triggers** pane appears. Choose **Add trigger** to add a trigger. For more information, see [Adding a push, pull, or schedule trigger](#p.329).

b. Choose **+ Actions** (top-left). The **Actions** catalog appears.

c. Choose the plus sign (+) inside an action to add it to the workflow. Use the pane on the right to configure the action. For more information, see [Adding an action](#p.250).

d. (Optional) Choose **Workflow properties** (top-right). A **Workflow properties** pane appears. Configure the workflow name run mode, and compute. For more information, see [Configuring queued, superseded, and parallel runs](#p.317) and [Working with compute and runtime environment Docker images](#p.293).

10. (Optional) Choose **Validate** to validate the workflow’s YAML code before committing.

11. Choose **Commit**, and on the **Commit workflow** dialog box, do the following:

   a. For **Workflow file name**, leave the default name or enter your own.

   b. For **Commit message**, leave the default message or enter your own.

   c. For **Repository** and **Branch**, choose the source repository and branch for the workflow definition file. These fields should be set to the repository and branch that you specified earlier in the **Create workflow** dialog box. You can change the repository and branch now, if you’d like.

      **Note**
      After committing your workflow definition file, it cannot be associated with another repository or branch, so make sure to choose them carefully.

   d. Choose **Commit** to commit the workflow definition file.

**YAML**

**To create a workflow using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose **Create workflow**.

   The **Create workflow** dialog box appears.

5. In the **Source repository** field, choose a source repository where the workflow definition file will reside. The file will be stored in the ~/.codecatalyst/workflows/ folder in the chosen repository. If no source repository exists, create one ([p. 89](#p.89)).

6. In the **Branch** field, choose a branch where the workflow definition file will reside.
7. Choose **Create**.

   Amazon CodeCatalyst saves the repository and branch information in memory, but the workflow is not yet committed.

8. Choose **YAML**.

9. Build the workflow:

   a. (Optional) Add a trigger to the YAML code. For more information, see [Adding a push, pull, or schedule trigger](#p.329).

   b. Choose **+ Actions** (top-left). The **Actions** catalog appears.

   c. Choose the plus sign (+) inside an action to add it to the workflow. Use the pane on the right to configure the action. For more information, see [Adding an action](#p.250).
d. (Optional) Choose **Workflow properties** (top-right). A **Workflow properties** pane appears. Configure the workflow name, run mode, and compute. For more information, see *Configuring queued, superseded, and parallel runs (p. 317)* and *Working with compute and runtime environment Docker images (p. 293).*

10. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

11. Choose **Commit**, and on the **Commit workflow** dialog box, do the following:

   a. For **Workflow file name**, leave the default name or enter your own.

   b. For **Commit message**, leave the default message or enter your own.

   c. For **Repository** and **Branch**, choose the source repository and branch for the workflow definition file. These fields should be set to the repository and branch that you specified earlier in the **Create workflow** dialog box. You can change the repository and branch now, if you'd like.

   **Note**
   After committing your workflow definition file, it cannot be associated with another repository or branch, so make sure to choose them carefully.

   d. Choose **Commit** to commit the workflow definition file.

---

**To edit a workflow**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).

2. Choose your project.

3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.

4. Choose the name of the workflow that you want to edit. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Choose **Edit**.

6. Edit the workflow. Choose **Visual** to use the visual editor, or **YAML** to use the YAML editor. You can:

   a. Edit triggers. For more information, see *Working with triggers (p. 327).*

   b. Edit actions. For more information, see *Working with actions (p. 249).*

7. Edit additional workflow properties. Choose **Workflow properties** (top-right) to change the workflow name, run mode, or compute. For more information, see *Configuring queued, superseded, and parallel runs (p. 317)* and *Working with compute and runtime environment Docker images (p. 293).*

---

**To delete a workflow**

Deleting a workflow deletes the workflow definition file and associated runs. Any AWS services deployed from the deleted workflow must be manually removed or charges will continue to incur for those services.

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).

2. Choose your project.

3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.

4. Choose the name of the workflow that you want to delete. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Choose **Delete**.

6. Choose **Confirm**.
Working with actions

An action is the main building block of a workflow, and defines a logical unit of work to perform during a workflow run. Typically, a workflow includes multiple actions that run sequentially or in parallel depending on how you've configured them.

Topics
• Action types (p. 249)
• Adding, configuring, and removing an action (p. 250)
• Developing a custom action (p. 278)
• Grouping actions into action groups (p. 279)
• Configuring actions to depend on other actions (p. 280)
• Viewing an action's source code (p. 283)
• Working with action versions (p. 284)

Action types

Within an Amazon CodeCatalyst workflow, you can use the following types of actions.

CodeCatalyst actions

A CodeCatalyst action is an action that is authored, maintained, and fully supported by the CodeCatalyst development team.

There are CodeCatalyst actions for building, testing, and deploying applications, as well as for performing miscellaneous tasks, such as invoking an AWS Lambda function.

Documentation for CodeCatalyst actions is available in this guide, and in each action's readme.

For information about the available CodeCatalyst actions, and how to add one to a workflow, see Adding an action (p. 250).

CodeCatalyst Labs actions

A CodeCatalyst Labs action is an action that is part of Amazon CodeCatalyst Labs, a proving ground for experimental applications. CodeCatalyst Labs actions have been developed to showcase integrations with AWS services.

Documentation for CodeCatalyst Labs actions is available in each action's readme.

For information about adding a CodeCatalyst Labs action to a workflow and viewing its readme, see Adding an action (p. 250).

GitHub Actions

A GitHub Action is a lot like a CodeCatalyst action (p. 249), except that it was developed for use with GitHub workflows. For details about GitHub Actions, see the GitHub Actions documentation.

You can use a GitHub Action alongside native CodeCatalyst actions in a CodeCatalyst workflow.

For your convenience, the CodeCatalyst console provides access to several popular GitHub Actions. You can also use any GitHub Action listed in the GitHub Marketplace (subject to a few limitations).

Documentation for GitHub Actions is available in each action's readme.
For more information, see Adding GitHub Actions (p. 266).

Third-party actions

A third-party action is an action that is authored by a third-party vendor, and made available in the CodeCatalyst console. An example of a third-party action is the Mend SCA action, authored by Mend.

Documentation for third-party actions is available in each action's readme. Additional documentation might also be provided by the third-party vendor.

For information about adding a third-party action to a workflow and viewing its readme, see Adding an action (p. 250).

Adding, configuring, and removing an action

Choose one of the following topics to learn how to add the action to a workflow and how to remove it.

Topics
  • Adding an action (p. 250)
  • Adding the "Amazon S3 publish" action (p. 251)
  • Adding the "AWS CDK bootstrap" action (p. 254)
  • Adding the "AWS Lambda invoke" action (p. 256)
  • Adding the "Render Amazon ECS task definition" action (p. 260)
  • Adding GitHub Actions (p. 266)
  • Removing an action (p. 278)

Adding an action

Use the following instructions to add an action to a workflow and then configure it.

To add and configure an action

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. At the top-left, choose + Actions, The Actions catalog appears.
7. In the drop-down list, do one of the following:
   • Choose Amazon CodeCatalyst to view CodeCatalyst (p. 249), CodeCatalyst Labs (p. 249), or third-party (p. 250) actions.
     • CodeCatalyst actions have a by AWS label.
     • CodeCatalyst Labs actions have a by CodeCatalyst Labs label.
     • Third-party actions have a by vendor label, where vendor is the name of the third-party vendor.
   • Choose GitHub to view a curated list of GitHub Actions (p. 273).
8. In the action catalog, search for an action, and then do one of the following:
   • Choose the plus sign (+) to add the action to your workflow.
   • Choose the action's name to view its readme.
9. Configure the action. Choose Visual to use the visual editor, or YAML to use the YAML editor. For detailed instructions, see the following links.

For instructions on adding CodeCatalyst actions (p. 249), see:

- Adding the build action (p. 139)
- Adding the test action (p. 149)
- Adding the "Deploy AWS CloudFormation stack" action (p. 228)
- Adding the "Deploy to Amazon ECS" action (p. 230)
- Adding the "AWS CDK deploy" action (p. 234)
- Adding the "Amazon S3 publish" action (p. 251)
- Adding the "AWS CDK bootstrap" action (p. 254)
- Adding the "AWS Lambda invoke" action (p. 256)
- Adding the "Render Amazon ECS task definition" action (p. 260)

For instructions on adding CodeCatalyst Labs actions (p. 249), see:

- The action's readme. You can find the readme by choosing the action's name in the action catalog.

For instructions on adding GitHub Actions (p. 249), see:

- Adding GitHub Actions (p. 266)

For instructions on adding third-party actions (p. 250), see:

- The action's readme. You can find the readme by choosing the action's name in the action catalog.

10. (Optional) Choose Validate to make sure the YAML code is valid.

11. Choose Commit to commit your changes.

Adding the "Amazon S3 publish" action

This section describes how to add the Amazon S3 publish action to your workflow. The Amazon S3 publish action copies files from a source directory to an Amazon S3 bucket. The source directory can reside in:

- A source repository (p. 79), or
- An output artifact (p. 286) generated by another workflow action

Topics

- When to use this action (p. 251)
- Example workflow (p. 252)
- Adding the "Amazon S3 publish" action (p. 253)
- Variables produced by the "Amazon S3 publish" action (p. 254)
- "Amazon S3 publish" action definition (p. 254)

When to use this action

Use this action if:

- You have a workflow that generates files that you want to store in Amazon S3.
For example, you might have a workflow that builds a static website that you want to host in Amazon S3. In this case, your workflow would include a build action (p. 139) to build the site's HTML and supporting files, and an Amazon S3 publish action to copy the files to Amazon S3.

- You have a source repository that contains files that you want to store in Amazon S3.

For example, you might have a source repository with application source files that you want to archive on a nightly basis to Amazon S3.

**Example workflow**

The following example workflow includes the Amazon S3 publish action, along with a build action. The workflow builds a static documentation website and then publishes it to Amazon S3, where it is hosted. The workflow consists of the following building blocks that run sequentially:

- **A trigger** – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see Working with triggers (p. 327).
- **A build action (BuildDocs)** – On trigger, the action builds a static documentation website (mkdocs build) and adds the associated HTML files and supporting metadata to an artifact called MyDocsSite. For more information about the build action, see Building using workflows in CodeCatalyst (p. 138).
- **An Amazon S3 publish action (PublishToS3)** – On completion of the build action, this action copies the site in the MyDocsSite artifact to Amazon S3 for hosting.

```
Name: codecatalyst-s3-publish-workflow
SchemaVersion: 1.0

Triggers:
- Type: PUSH
  Branches:
  - main

Actions:
BuildDocs:
  Identifier: aws/build@v1
  Inputs:
    Sources:
      - WorkflowSource
  Configuration:
    Steps:
      - Run: echo BuildDocs started on `date`
      - Run: pip install --upgrade pip
      - Run: pip install mkdocs
      - Run: mkdocs build
      - Run: echo BuildDocs completed on `date`
  Outputs:
    Artifacts:
      - Name: MyDocsSite
        Files:
        - "site/**/*"

PublishToS3:
  Identifier: aws/s3-publish@v1
  Environment:
    Name: codecatalyst-s3-publish-environment
  Connections:
    - Name: codecatalyst-account-connection
```
Role: codecatalyst-s3-publish-build-role

Inputs:
- WorkflowSource
- MyDocsSite

Artifacts:
- MyDocsSite

Configuration:
- DestinationBucketName: my-bucket
- SourcePath: /artifacts/PublishToS3/MyDocSite/site
- TargetPath: my/docs/site

Adding the "Amazon S3 publish" action

Use the following instructions to add the Amazon S3 publish action to your workflow.

Visual

To add the "Amazon S3 publish" action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the Amazon S3 publish action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Or
   - Choose Amazon S3 publish. The action details dialog box appears. On this dialog box:
     - (Optional) Choose View source to view the action's source code (p. 379).
     - Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.
10. In the Inputs, Configuration, and Outputs tabs, complete the fields according to your needs. For a description of each field, see the "Amazon S3 publish" action reference (p. 379). This reference provides detailed information on each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.
11. (Optional) Choose Validate to validate the workflow's YAML code before committing.
12. Choose Commit, enter a commit message, and then choose Commit again.

YAML

To add the "Amazon S3 publish" action using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose + **Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **Amazon S3 publish** action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   Or
   - Choose **Amazon S3 publish**. The action details dialog box appears. On this dialog box:
     - (Optional) Choose **View source** to view the action’s source code (p. 379).
     - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.
10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "Amazon S3 publish" action reference (p. 379).
11. (Optional) Choose **Validate** to validate the workflow’s YAML code before committing.
12. Choose **Commit**, enter a commit message, and then choose **Commit** again.

**Variables produced by the "Amazon S3 publish" action**

When the **Amazon S3 publish** action runs, it produces variables that you can use in subsequent workflow actions. For details, see "Amazon S3 publish" action variables (p. 347) in the List of predefined variables (p. 346).

"Amazon S3 publish" action definition

The **Amazon S3 publish** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "Amazon S3 publish" action reference (p. 379) in the Workflow definition reference (p. 354).

**Adding the "AWS CDK bootstrap" action**

This section describes how to add the **AWS CDK bootstrap** action to your workflow. The **AWS CDK bootstrap** action provisions a bootstrap stack in your AWS environment using the modern template. If a bootstrap stack already exists, the action updates it if necessary. Having a bootstrap stack present in AWS is a prerequisite for deploying an AWS CDK app.

For more information about bootstrapping, see Bootstrapping in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

**Topics**

- When to use this action (p. 254)
- Prerequisites (p. 255)
- Example workflow (p. 255)
- Adding the "AWS CDK bootstrap" action (p. 255)
- Variables produced by the "AWS CDK bootstrap" action (p. 256)
- "AWS CDK bootstrap" action definition (p. 256)

**When to use this action**

Use this action if you have a workflow that deploys an AWS CDK app, and you want to deploy (and update, if needed) the bootstrap stack at the same time. In this case, you would add the **AWS CDK bootstrap** action to the same workflow as the one that deploys your AWS CDK app.
Do not use this action if either of the following applies:

- You already deployed a bootstrap stack using another mechanism, and you want to keep it intact (no updates).
- You want to use a custom bootstrap template, which is not supported with the AWS CDK bootstrap action.

**Prerequisites**

Before you can use the AWS CDK bootstrap action, make sure you have an AWS CDK app ready. The bootstrap action will synthesize the AWS CDK app before bootstrapping. You can write your app in any programming language supported by the AWS CDK.

Make sure your AWS CDK app files are available in:

- A CodeCatalyst source repository (p. 79), or
- A CodeCatalyst output artifact (p. 286) generated by another workflow action

**Example workflow**

Refer to the Example workflow (p. 235) in the Adding the "AWS CDK deploy" action (p. 234) for a workflow that includes the AWS CDK bootstrap action.

**Adding the "AWS CDK bootstrap" action**

Use the following instructions to add the AWS CDK bootstrap action to your workflow.

**Prerequisites**

Before you begin, make sure you have completed the tasks described in Prerequisites (p. 255).

**Visual**

**To add the "AWS CDK bootstrap" action using the visual editor**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the AWS CDK bootstrap action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Or
   - Choose AWS CDK bootstrap. The action details dialog box appears. On this dialog box:
     - (Optional) Choose View source to view the action's source code (p. ).
     - Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.
10. In the **Inputs**, **Configuration**, and **Outputs** tabs, complete the fields according to your needs. For a description of each field, see the "**AWS CDK bootstrap** action reference (p. 386). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and then choose **Commit** again.

**Note**

If your **AWS CDK bootstrap** action fails with an npm install error, see **How do I fix "npm install" errors? (p. 592)** for information about how to fix the error.

---

**YAML**

**To add the "AWS CDK bootstrap" action using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **AWS CDK bootstrap** action, and choose **+** to add it to the workflow diagram and open its configuration pane.
10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "**AWS CDK bootstrap** action reference (p. 386).

11. (Optional) Choose **Validate** to validate the workflow’s YAML code before committing.

12. Choose **Commit**, enter a commit message, and then choose **Commit** again.

**Note**

If your **AWS CDK bootstrap** action fails with an npm install error, see **How do I fix "npm install" errors? (p. 592)** for information about how to fix the error.

---

**Variables produced by the "AWS CDK bootstrap" action**

When the **AWS CDK bootstrap** action runs, it produces variables that you can use in subsequent workflow actions. For details, see "**AWS CDK bootstrap** action variables (p. 347)" in the List of predefined variables (p. 346).

"**AWS CDK bootstrap** action definition"

The **AWS CDK bootstrap** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "**AWS CDK bootstrap** action reference (p. 386)" in the Workflow definition reference (p. 354).

**Adding the "AWS Lambda invoke" action**

This section describes how to add the **AWS Lambda invoke** action to your workflow. The **AWS Lambda invoke** action invokes a Lambda function that you specify.

In addition to invoking your function, the **AWS Lambda invoke** action also converts each top-level key in the response payload received from the Lambda function into a workflow output variable (p. 337).
These variables can then be referenced in subsequent workflow actions. If you don't want all top-level keys to be converted to variables, you can use filters to specify the exact ones. For more information, see ResponseFilters (p. 410) property description in the "AWS Lambda invoke" action reference (p. 402).

Topics
- When to use this action (p. 257)
- Example workflow (p. 257)
- Adding the "AWS Lambda invoke" action (p. 258)
- Variables produced by the "AWS Lambda invoke" action (p. 259)
- "AWS Lambda invoke" action definition (p. 259)

When to use this action
Use this action if you want to add functionality to your workflow that is encapsulated in, and performed by, a Lambda function.

For example, you might want your workflow to send a Build started notification to a Slack channel before starting a build of your application. In this case, your workflow would include an AWS Lambda invoke action to invoke a Lambda to send out the Slack notification, and a build action (p. 139) to build your application.

As another example, you might want your workflow to conduct a vulnerability scan on your application before it is deployed. In this case, you would use a build action to build your application, an AWS Lambda invoke action to invoke a Lambda to scan for vulnerabilities, and a deploy action to deploy the scanned application.

Example workflow

Note
The following example workflow is for illustrative purposes only, and requires additional setup work in order to function properly. It is meant to give you an example of what a workflow might look like when it is configured with the AWS Lambda invoke action.

The following workflow includes the AWS Lambda invoke action, along with a deploy action. The workflow sends out a Slack notification indicating that a deployment has started, and then deploys an application into AWS using an AWS CloudFormation template. The workflow consists of the following building blocks that run sequentially:

- A trigger – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see Working with triggers (p. 327).
- An AWS Lambda invoke action (LambdaNotify) – On trigger, this action invokes the Notify-Start Lambda function in the specified AWS account and Region (my-aws-account, and us-west-2). On invocation, the Lambda function sends a Slack notification indicating a deployment has started.
- A Deploy AWS CloudFormation stack action (Deploy) – On completion of the AWS Lambda invoke action, the Deploy AWS CloudFormation stack action runs the template (cfn-template.yml) to deploy your application stack. For more information about the Deploy AWS CloudFormation stack action, see Adding the "Deploy AWS CloudFormation stack" action (p. 228).

Name: codecatalyst-lamda-invoke-workflow
SchemaVersion: 1.0

Triggers:
- Type: PUSH
  Branches:
  - main
Adding the "AWS Lambda invoke" action

Use the following instructions to add the AWS Lambda invoke action to your workflow.

Prerequisite

Before you begin, make sure your AWS Lambda function and associated Lambda execution role are ready and available in AWS. For more information, see the Lambda execution role topic in the AWS Lambda Developer Guide.

Visual

To add the "AWS Lambda invoke" action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the AWS Lambda invoke action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
Or

- Choose **AWS Lambda invoke**. The action details dialog box appears. On this dialog box:
  - (Optional) Choose **View source** to view the action's source code (p. 402).
  - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. In the **Inputs**, **Configuration**, and **Outputs** tabs, complete the fields according to your needs. For a description of each field, see the ‘**AWS Lambda invoke** action reference (p. 402). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

---

**YAML**

**To add the "AWS Lambda invoke" action using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **Amazon CodeCatalyst**.
9. Search for the **AWS Lambda invoke** action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   
   Or
   - Choose **AWS Lambda invoke**. The action details dialog box appears. On this dialog box:
     - (Optional) Choose **View source** to view the action's source code (p. 402).
     - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "**AWS Lambda invoke** action reference (p. 402)."

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

---

**Variables produced by the "AWS Lambda invoke" action**

When the **AWS Lambda invoke** action runs, it produces variables that you can use in subsequent workflow actions. For details, see "**AWS Lambda invoke** action variables (p. 349) in the **List of predefined variables** (p. 346).

"**AWS Lambda invoke**" action definition

The **AWS Lambda invoke** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "**AWS Lambda invoke** action reference (p. 402) in the **Workflow definition reference** (p. 354)."
Adding the "Render Amazon ECS task definition" action

This section describes how to add the Render Amazon ECS task definition action to your workflow. This action updates the image field in an Amazon Elastic Container Service (Amazon ECS) task definition file with a Docker image name that is supplied by your workflow at runtime.

Note
You can also use this action to update the task definition's environment field with environment variables.

Topics
- When to use this action (p. 260)
- How it works (p. 260)
- Example workflow (p. 261)
- Adding the "Render Amazon ECS task definition" action (p. 263)
- Viewing the updated task definition file (p. 265)
- Variables produced by the "Render Amazon ECS task definition" action (p. 265)
- "Render Amazon ECS task definition" action definition (p. 266)

When to use this action

Use this if you have a workflow that builds and tags a Docker image with dynamic content, such as a commit ID or timestamp.

Do not use this action if your task definition file contains an image value that always stays the same. In this case, you can manually enter the name of your image into the task definition file.

How it works

You must use the Render Amazon ECS task definition action with the build and Deploy to Amazon ECS actions in your workflow. Together, these actions work as follows:

1. The build action builds your Docker image and tags it with a name, a commit ID, timestamp, or other dynamic content. For example, your build action might look like this:

   ```
   MyECSWorkflow
   Actions:
   BuildAction:
     Identifier: aws/build@v1
     ...  
   Configuration:
   Steps:
   # Build, tag, and push the Docker image...
   ...
   ```

   In the preceding code, the docker build -t directive indicates to build the Docker image and tag it with the commit ID at action runtime. The generated image name might look like this:

   ```
   MyDockerImage:a37bd7e
   ```

2. The Render Amazon ECS task definition action adds the dynamically generated image name, MyDockerImage:a37bd7e, to your task definition file, like this:

   ```
   {
   "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role",
   "containerDefinitions": [
   ```
Optionally, you can also have the **Render Amazon ECS task definition** action add environment variables to the task definition, like this:

```json
{
   "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role",
   "containerDefinitions": [
      {
         "name": "codecatalyst-ecs-container",
         "image": MyDockerImage:a37bd7e,
         ...
      },
      ...
   ],
   ...
}
```

For more information about environment variables, see [Specifying environment variables](http://amazonelasticcontainer) in the *Amazon Elastic Container Service Developer Guide*.

3. The **Deploy to Amazon ECS** action registers the updated task definition file with Amazon ECS. Registering the updated task definition file deploys the new image, MyDockerImage:a37bd7e into Amazon ECS.

**Example workflow**

The following is an example of a full workflow that includes the **Render Amazon ECS task definition** action, along with build and deploy actions. The workflow’s purpose is to build and deploy a Docker image into your Amazon ECS cluster. The workflow consists of the following building blocks that run sequentially:

- **A trigger** – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see [Working with triggers](p. 327).
- **A build** action (BuildDocker) – On trigger, the action builds the Docker image using the Dockerfile, tags it with a commit ID, and pushes the image to Amazon ECR. For more information about the build action, see [Building using workflows in CodeCatalyst](p. 138).
- **A Render Amazon ECS task definition** action (RenderTaskDef) – On completion of the build action, this action updates an existing taskdef.json located in the root of your source repository with an
image field value that includes the correct commit ID. It saves the updated file with a new file name (task-definition-random-string.json) and then creates an output artifact that contains this file. The render action also generates a variable called task-definition and sets it to the name of the new task definition file. The artifact and variable will be used the deploy action, which is next.

- **A Deploy to Amazon ECS action** (DeployToECS) – On completion of the Render Amazon ECS task definition action, the Deploy to Amazon ECS action looks for the output artifact generated by the render action (TaskDefArtifact), finds the task-definition-random-string.json file inside of it, and registers it with your Amazon ECS service. The Amazon ECS service then follows the instructions in the task-definition-random-string.json file to run Amazon ECS tasks—and associated Docker image containers—inside your Amazon ECS cluster.

Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
- Type: PUSH
  Branches:
    - main

Actions:
BuildDocker:
  Identifier: aws/build@v1
  Environment:
    Name: codecatalyst-ecs-environment
  Connections:
    - Name: codecatalyst-account-connection
      Role: codecatalyst-ecs-build-role
  Inputs:
    Variables:
      - Name: REPOSITORY_URI
        Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-repo
      - Name: IMAGE_TAG
        Value: ${WorkflowSource.CommitId}
  Configuration:
    Steps:
    #pre_build:
    - Run: echo Logging in to Amazon ECR...
    - Run: aws --version
    - Run: aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 111122223333.dkr.ecr.us-east-2.amazonaws.com
    #build:
    - Run: echo Build started on `date`
    - Run: echo Building the Docker image...
    - Run: docker build -t $REPOSITORY_URI:latest .
    - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
    #post_build:
    - Run: echo Build completed on `date`
    - Run: echo Pushing the Docker images...
    - Run: docker push $REPOSITORY_URI:latest
    - Run: docker push $REPOSITORY_URI:$IMAGE_TAG

RenderTaskDef:
  DependsOn:
    - BuildDocker
  Identifier: aws/ecs-render-task-definition@v1
  Inputs:
    Variables:
      - Name: REPOSITORY_URI
        Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-repo
      - Name: IMAGE_TAG
        Value: ${WorkflowSource.CommitId}
  Configuration:
    task-definition: taskdef.json
Adding the "Render Amazon ECS task definition" action

Use the following instructions to add the **Render Amazon ECS task definition** action to your workflow.

**Prerequisite**

Before you begin, make sure you have a workflow that includes a build action that dynamically generates a Docker image. See the preceding example workflow (p. 261) for details.

**Visual**

To add the "Render Amazon ECS task definition" action using the visual editor

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose Amazon CodeCatalyst.
9. Search for the **Render Amazon ECS task definition** action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.

   Or
Choose **Render Amazon ECS task definition**. The action details dialog box appears. On this dialog box:

- (Optional) Choose **View source** to view the action's source code (p. 477).
- Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. In the **Inputs** and **Configuration** tabs, complete the fields according to your needs. For a description of each field, see the "Render Amazon ECS task definition" action reference (p. 452). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

### YAML

**To add the "Render Amazon ECS task definition" action using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).

2. Choose your project.

3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.

4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Choose **Edit**.

6. Choose **YAML**.

7. At the top-left, choose **+ Actions** to open the action catalog.

8. From the drop-down list, choose **Amazon CodeCatalyst**.

9. Search for the **Render Amazon ECS task definition** action, and do one of the following:

   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.

   Or

   - Choose **Render Amazon ECS task definition**. The action details dialog box appears. On this dialog box:

     - (Optional) Choose **View source** to view the action's source code (p. 477).
     - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "Render Amazon ECS task definition" action reference (p. 452).

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

### Next steps

After adding the render action, add the **Deploy to Amazon ECS** action to your workflow following the instructions in [Adding the "Deploy to Amazon ECS" action](p. 230). While adding the deploy action, do the following:

1. In the **Inputs** tab of the deploy action, in **Artifacts - optional**, select the artifact that was generated by the render action. It contains the updated task definition file.

   For more information about artifacts, see [Working with artifacts](p. 286).
2. In the **Configuration** tab of the deploy action, in the **Task definition** field, specify the following action variable: `${action-name}.task-definition` where *action-name* is the name of your render action, for example, *RenderTaskDef*. The render action sets this variable to the new name of the task definition file.

   For more information about variables, see *Working with variables* (p. 337).

   For more information about how to configure the deploy action, see the preceding *example workflow* (p. 261).

**Viewing the updated task definition file**

You can view the name and contents of the updated task definition file.

**To view the name of the updated task definition file, after the Render Amazon ECS task definition action has processed it.**

1. Find the run that includes a completed render action:
   a. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
   b. Choose your project.
   c. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
   d. Choose the name of the workflow that contains the render action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
   e. Choose a run that includes the completed render action.
2. In the workflow diagram, choose the render action.
3. Choose **Outputs**.
4. Choose **Variables**.
5. The task definition file name is displayed. It looks similar to *task-definition--259-0a2r7gx1TF5X--.json*.

**To view the contents of the updated task definition file**

1. Find the run that includes a completed render action:
   a. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
   b. Choose your project.
   c. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
   d. Choose the name of the workflow that contains the render action. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
   e. Choose a run that includes the completed render action.
2. In the workflow run, at the top, next to **Visual** and **YAML**, choose **Workflow outputs**.
3. In the **Artifacts** section, choose **Download** next to the artifact that contains the updated task definition file. This artifact will have a **Produced by** column set to the name of your render action.
4. Open the .zip file to view the task definition .json file.

**Variables produced by the "Render Amazon ECS task definition" action**

When the **Render Amazon ECS task definition** action runs, it produces variables that you can use in subsequent workflow actions. For details, see "*Render Amazon ECS task definition* action variables" (p. 351) in the List of predefined variables (p. 346).
"Render Amazon ECS task definition" action definition

The Render Amazon ECS task definition action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "Render Amazon ECS task definition" action reference (p. 452) in the Workflow definition reference (p. 354).

Adding GitHub Actions

A GitHub Action is a lot like a CodeCatalyst action (p. 249), except that it was developed for use with GitHub workflows. For details about GitHub Actions, see the GitHub Actions documentation.

You can use a GitHub Action alongside native CodeCatalyst actions in a CodeCatalyst workflow.

There are two ways to add a GitHub Action to a CodeCatalyst workflow:

- You can select the GitHub Action from a curated list in the CodeCatalyst console. Several popular GitHub Actions are available. For more information, see Adding a curated GitHub Action (p. 273).
- If the GitHub Action that you want to use is not available in the CodeCatalyst console, you can add it using a GitHub Actions action.

A GitHub Actions action is a CodeCatalyst action that wraps a GitHub Action and makes it compatible with CodeCatalyst workflows.

Here is an example of a GitHub Actions action wrapping the Super-Linter GitHub Action:

```
Actions:
  GitHubAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Lint Code Base
          uses: github/super-linter@v4
          env:
            VALIDATE_ALL_CODEBASE: "true"
            DEFAULT_BRANCH: main
```

In the previous code, the CodeCatalyst GitHub Actions action (identified by aws/github-actions-runner@v1) wraps the Super-Linter action (identified by github/super-linter@v4), making it work in a CodeCatalyst workflow.

For more information, see Adding the "GitHub Actions" action (p. 272).

All GitHub Actions—both curated and not—must be wrapped inside a GitHub Actions action (aws/github-actions-runner@v1), as shown in the previous example. The wrapper is required for the action to function properly.

Topics

- How are GitHub Actions different from CodeCatalyst actions? (p. 267)
- Can GitHub Actions interact with other CodeCatalyst actions in the workflow? (p. 267)
- Which GitHub Actions can I use? (p. 267)
- Limitations of GitHub Actions in CodeCatalyst (p. 267)
- How do I add a GitHub Action (high-level steps)? (p. 267)
- Does the GitHub Action run in GitHub? (p. 268)
- Can I use GitHub workflows too? (p. 268)
- Tutorial: Lint code using a GitHub Action (p. 268)
- Adding the "GitHub Actions" action (p. 272)
How are GitHub Actions different from CodeCatalyst actions?

GitHub Actions that are used inside a CodeCatalyst workflow do not have the same level of access and integration with AWS and CodeCatalyst features (such as environments (p. 307) and issues (p. 459)) that CodeCatalyst actions do.

Can GitHub Actions interact with other CodeCatalyst actions in the workflow?

Yes. For example, GitHub Actions can use variables produced by other CodeCatalyst actions as input, and can also share output parameters and artifacts with CodeCatalyst actions. For more information, see Working with GitHub Actions output parameters (p. 276).

Which GitHub Actions can I use?

You can use any GitHub Action available through the CodeCatalyst console, and any GitHub Action available in the GitHub Marketplace. If you decide to use a GitHub Action from the Marketplace, keep in mind the following limitations (p. 267).

Limitations of GitHub Actions in CodeCatalyst

- GitHub Actions cannot be used with the CodeCatalyst Lambda compute type (p. 294).
- GitHub Actions that internally rely on the github context or that reference GitHub-specific resources won't work in CodeCatalyst. For example, the following actions won't work in CodeCatalyst:
  - Actions that attempt to add, change, or update GitHub resources. Examples include actions that update pull requests, or create issues in GitHub.
  - Almost all actions listed in https://github.com/actions.
- GitHub Actions that are Docker container actions will work, but they must be run by the default Docker user (root). Do not run the action as user 1001. (At the time of writing, user 1001 works in GitHub, but not in CodeCatalyst.) For more information, see the USER topic in Dockerfile support for GitHub Actions.

For a list of GitHub Actions available through the CodeCatalyst console, see Adding a curated GitHub Action (p. 273).

How do I add a GitHub Action (high-level steps)?

The high-level steps to add a GitHub Action to a CodeCatalyst workflow are as follows:

1. In your CodeCatalyst project, you create a workflow. The workflow is where you define how to build, test, and deploy your application. For more information, see Getting started with workflows in CodeCatalyst (p. 134).
2. In the workflow, you add a curated GitHub Action or you add the GitHub Actions action.
3. You do one of the following:
   - If you chose to add a curated action, configure it. For more information, see Adding a curated GitHub Action (p. 273).
   - If you chose to add a non-curated action, within the GitHub Actions action, you paste the GitHub Action's YAML code. You can find this code on the details page of your chosen GitHub Action in the GitHub Marketplace. You will likely need to modify the code slightly to have it work in CodeCatalyst. For more information, see Adding the "GitHub Actions" action (p. 272).
4. (Optional) Within the workflow, you add other actions like the build and test actions. For more information, see Build, test, and deploy with workflows in CodeCatalyst (p. 128).
5. You **start the workflow** either manually or automatically through a trigger. The workflow runs the GitHub Action and any other actions in the workflow. For more information, see [Starting a workflow run](p. 314).

For detailed steps, see:

- Adding a curated GitHub Action (p. 273).
- Adding the "GitHub Actions" action (p. 272).

**Does the GitHub Action run in GitHub?**


**Can I use GitHub workflows too?**

No.

**Tutorial: Lint code using a GitHub Action**

In this tutorial, you add the **Super-Linter GitHub Action** to an Amazon CodeCatalyst workflow. The Super-Linter action inspects code, finds areas where the code has errors, formatting issues, and suspicious constructs, and then outputs the results to the CodeCatalyst console. After adding the linter to your workflow, you run the workflow to lint a sample Node.js application (app.js). You then fix the reported problems and run the workflow again to see if the fixes worked.

*Tip*

Consider using Super-Linter to lint YAML files, such as AWS CloudFormation templates.

**Topics**

- Prerequisites (p. 268)
- Step 1: Create a source repository (p. 268)
- Step 2: Add an app.js file (p. 269)
- Step 3: Create a workflow that runs the Super-Linter action (p. 270)
- Step 4: Fix problems that the Super-Linter found (p. 271)
- Clean up (p. 272)

**Prerequisites**

Before you begin, you'll need:

- A CodeCatalyst **space** with a connected AWS account. For more information, see [Creating additional spaces in CodeCatalyst](p. 53).
- An empty **project** in your CodeCatalyst space called codecatalyst-linter-project. Choose the **Start from scratch** option to create this project.

For more information, see [Creating an empty project in Amazon CodeCatalyst](p. 58).

**Step 1: Create a source repository**

In this step, you create a source repository in CodeCatalyst. You'll use this repository to store the sample application source file, app.js, for this tutorial.

For more information about source repositories, see [Creating a source repository](p. 89).
To create a source repository

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your project, codecatalyst-linter-project.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Create repository.
5. In Repository name, enter:

   codecatalyst-linter-source-repository

6. Choose Create.

Step 2: Add an app.js file

In this step, you add an app.js file to your source repository. The app.js contains function code that has a few mistakes that the linter will find.

To add the app.js file

1. In the CodeCatalyst console, choose your project, codecatalyst-linter-project.
2. In the navigation pane, choose Code, and then choose Source repositories.
3. From the list of source repositories, choose your repository, codecatalyst-linter-source-repository.
4. In Files, choose Create file.
5. In the text box, enter the following code:

```javascript
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;

/**
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
 * lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
 * context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
 * lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
  try {
      // const ret = await axios(url);
      response = {
          statusCode: 200,
          'body': JSON.stringify({
              message: 'hello world'
          // location: ret.data.trim()
          })
      }
    } catch (err) {
        console.log(err)
        return err
    }
```
6. For **File name**, enter `app.js`. Keep the other default options.
7. Choose **Commit**.

   You have now created a file called `app.js`.

**Step 3: Create a workflow that runs the Super-Linter action**

In this step, you create a workflow that runs the Super-Linter action when you push code to your source repository. The workflow consists of the following building blocks, which you define in a YAML file:

- **A trigger** – This trigger starts the workflow run automatically when you push a change to your source repository. For more information about triggers, see [Working with triggers](p. 327).
- **A "GitHub Actions" action** – On trigger, the **GitHub Actions** action runs the Super-Linter action, which in turn inspects all files in your source repository. If the linter finds an issue, the workflow action fails.

**To create a workflow that runs the Super-Linter action**

1. In the CodeCatalyst console, choose your project, `codecatalyst-linter-project`.
2. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
3. Choose **Create workflow**.
4. For **Source repository**, choose `codecatalyst-linter-source-repository`.
5. For **Branch**, choose `main`.
6. Choose **Create**.
7. Delete the YAML sample code.
8. Add the following YAML:

   ```yaml
   Name: codecatalyst-linter-workflow
   SchemaVersion: "1.0"
   Triggers:
   - Type: PUSH
     Branches:
     - main
   Actions:
   SuperLinterAction:
     Identifier: aws/github-actions-runner@v1
     Configuration:
     Steps:
     - github-action-code
   ```

   In the preceding code, replace `github-action-code` with the Super-Linter action code, as instructed in the following steps of this procedure.

9. Go to the **Super-Linter page** in the GitHub Marketplace.
10. Under **steps:** (lowercase), find the code and paste it into the CodeCatalyst workflow under **Steps:** (uppercase).

    Adjust the GitHub Action code to conform to CodeCatalyst standards, as shown in the following code.

    Your CodeCatalyst workflow now looks like this:

    ```yaml
    Name: codecatalyst-linter-workflow
    SchemaVersion: "1.0"
    ```
Triggers:
- Type: PUSH
  Branches:
  - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: "true"
          DEFAULT_BRANCH: main

11. (Optional) Choose Validate to make sure the YAML code is valid before committing.

12. Choose Commit, enter a Commit message, select your codecatalyst-linter-source-repository Repository, and choose Commit again.

You have now created a workflow. A workflow run starts automatically because of the trigger defined at the top of the workflow.

To view the workflow run in progress

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose the workflow you just created: codecatalyst-linter-workflow.
3. In the workflow diagram, choose SuperLinterAction.
4. Wait for the action to fail. This failure is expected because the linter found problems in the code.
5. Leave the CodeCatalyst console open and go to Step 4: Fix problems that the Super-Linter found (p. 271).

Step 4: Fix problems that the Super-Linter found

The Super-Linter should have found problems in the app.js code, as well as the README.md file included in your source repository.

To fix the problems the linter found

1. In the CodeCatalyst console, choose the Logs tab, and then choose Lint Code Base.

   The logs that the Super-Linter action generated are displayed.

2. In the Super-Linter logs, scroll down to around line 90, where you find the start of the problems. They look similar to the following:

   ```
   /github/workspace/hello-world/app.js:32:2: Newline required at end of file but not found.
   ```

3. Fix app.js and README.md in your source repository and commit your changes.

Tip
To fix the README.md, add markdown to the code block, like this:

```markdown
```
Your changes start another workflow run automatically. Wait for the workflow to finish. If you fixed all the problems, the workflow should succeed.

Clean up

Clean up in CodeCatalyst to remove traces of this tutorial from your environment.

To clean up in CodeCatalyst

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Delete codecatalyst-linter-source-repository.
3. Delete codecatalyst-linter-workflow.

In this tutorial, you learned how to add the Super-Linter GitHub Action to a CodeCatalyst workflow in order to lint some code.

Adding the "GitHub Actions" action

A GitHub Actions action is a CodeCatalyst action that wraps a GitHub Action and makes it compatible with CodeCatalyst workflows.

For more information, see Adding GitHub Actions (p. 266).

To add the GitHub Actions action to a workflow, follow these steps.

Tip

For a tutorial that shows you how to use the GitHub Actions action, see Tutorial: Lint code using a GitHub Action (p. 268).

Visual

To add the "GitHub Actions" action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose GitHub.
9. Search for the GitHub Actions action, and do one of the following:
   • Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   Or
   • Choose GitHub Actions. The action details dialog box appears. On this dialog box:
     • (Optional) Choose View source to view the action's source code (p. 272).
• Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.

10. In the **Inputs** and **Configuration** tabs, complete the fields according to your needs. For a description of each field, see the "GitHub Actions" action reference (p. 439). This reference provides detailed information about each field (and corresponding YAML property value) as it appears in both the YAML and visual editors.

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

12. Choose **Commit**, enter a commit message, and choose **Commit** again.

---

**YAML**

**To add the "GitHub Actions" action using the YAML editor**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. At the top-left, choose **+ Actions** to open the action catalog.
8. From the drop-down list, choose **GitHub**.
9. Search for the **GitHub Actions** action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Or
     - Choose **GitHub Actions**. The action details dialog box appears. On this dialog box:
       - (Optional) Choose **View source** to view the action's source code (p. 439).
       - Choose **Add to workflow** to add the action to the workflow diagram and open its configuration pane.
10. Modify the properties in the YAML code according to your needs. An explanation of each available property is provided in the "GitHub Actions" action reference (p. 439).
11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
12. Choose **Commit**, enter a commit message, and choose **Commit** again.

"GitHub Actions" action definition

The **GitHub Actions** action is defined as a set of YAML properties inside your workflow definition file. For information about these properties, see "GitHub Actions" action reference (p. 439) in the Workflow definition reference (p. 354).

Adding a curated GitHub Action

A curated GitHub Action is a GitHub Action that is made available in the CodeCatalyst console, and serves as an example of how to use a GitHub Action inside a CodeCatalyst workflow.

Curated GitHub Actions are wrapped in the CodeCatalyst-authored GitHub Actions action (p. 272), identified by the `aws/github-actions-runner@v1` identifier. For example, here's what the curated GitHub Action, **TruffleHog OSS**, looks like:
Actions:
TruffleHogOSS_e8:
  Identifier: aws/github-actions-runner@v1
  Inputs:
    Sources:
      - WorkflowSource # This specifies that the action requires this Workflow as a source
  Configuration:
    Steps:
      - uses: trufflesecurity/trufflehog@v3.16.0
        with:
          path: './' # Required; description: Repository path
          base: 'main' # Required; description: Start scanning from here (usually main branch).
          head: 'HEAD' # Optional; description: Scan commits until here (usually dev branch).
          extra_args: '--debug --only-verified' # Optional; description: Extra args to be passed to the trufflehog cli.

In the previous code, the CodeCatalyst GitHub Actions action (identified by aws/github-actions-runner@v1) wraps the TruffleHog OSS action (identified by trufflesecurity/trufflehog@v3.16.0), making it work in a CodeCatalyst workflow.

To configure this action, you would replace the empty strings under with: with your own values. For example:

Actions:
TruffleHogOSS_e8:
  Identifier: aws/github-actions-runner@v1
  Inputs:
    Sources:
      - WorkflowSource # This specifies that the action requires this Workflow as a source
  Configuration:
    Steps:
      - uses: trufflesecurity/trufflehog@v3.16.0
        with:
          path: './'
          base: 'main' # Required; description: Start scanning from here (usually main branch).
          head: 'HEAD' # Optional; description: Scan commits until here (usually dev branch).
          extra_args: '--debug --only-verified' # Optional; description: Extra args to be passed to the trufflehog cli.

To add a curated GitHub Action to a workflow, use the following procedure. For general information about using GitHub Actions in a CodeCatalyst workflow, see Adding GitHub Actions (p. 266).

Note
If you don't see your GitHub Action among the list of curated actions, you can still add it to your workflow using the GitHub Actions action. For more information, see Adding the "GitHub Actions" action (p. 272).

Visual

To add a curated GitHub action using the visual editor
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Choose Edit.


7. At the top-left, choose + Actions to open the action catalog.

8. From the drop-down list, choose GitHub.

9. Browse or search for a GitHub Action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Choose the name of the GitHub Action. The action details dialog box appears. On this dialog box:
     - (Optional) Choose View source to view the action's source code (p. 439).
     - Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.

10. In the Inputs, Configuration, and Outputs tabs, complete the fields according to your needs. For a description of each field, see the “GitHub Actions” action reference (p. 439). This reference provides detailed information about each field (and corresponding YAML property value) available to the GitHub Actions action, as it appears in both the YAML and visual editors.

    For information about the configuration options available to the curated GitHub Action, see its documentation.

11. (Optional) Choose Validate to validate the workflow's YAML code before committing.

12. Choose Commit, enter a commit message, and choose Commit again.

YAML

To add a curated GitHub action using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. At the top-left, choose + Actions to open the action catalog.
8. From the drop-down list, choose GitHub.
9. Browse or search for a GitHub Action, and do one of the following:
   - Choose the plus sign (+) to add the action to the workflow diagram and open its configuration pane.
   - Choose the name of the GitHub Action. The action details dialog box appears. On this dialog box:
     - (Optional) Choose View source to view the action's source code (p. ___).
     - Choose Add to workflow to add the action to the workflow diagram and open its configuration pane.
10. Modify the properties in the YAML code according to your needs. An explanation of each property available to the GitHub Actions action is provided in the "GitHub Actions" action reference (p. 439).

   For information about the configuration options available to the curated GitHub Action, see its documentation.

11. (Optional) Choose Validate to validate the workflow's YAML code before committing.

12. Choose Commit, enter a commit message, and choose Commit again.

Working with GitHub Actions output parameters

You can integrate GitHub output parameters into your CodeCatalyst workflows.

Note

Another word for output parameter is variable. Because GitHub uses the term output parameter in its documentation, we'll use this term too.

Topics

• Exporting a GitHub output parameter so that other actions can use it (p. 276)
• Referencing a GitHub output parameter (p. 277)

Exporting a GitHub output parameter so that other actions can use it

Use the following instructions to export a GitHub output parameter so that it is available for use in other CodeCatalyst workflow actions.

To export a GitHub output parameter

1. Open a workflow and choose Edit. For instructions, see To edit a workflow (p. 248).
2. In the GitHub Actions action that generates the output parameter that you want to export, add an Outputs section with an underlying Variables property that looks like this:

   Actions:
   MyGitHubAction:
       Identifier: aws/github-actions-runner@v1
   Outputs:
       Variables:
           - 'step-id_output-name'

   Replace:

   • step-id with value of the id: property in the GitHub action's steps section.
   • output-name with the name of the GitHub output parameter.

Example

The following example shows you how to export a GitHub output parameter called SELECTEDCOLOR.

   Actions:
   MyGitHubAction:
       Identifier: aws/github-actions-runner@v1
   Outputs:
       Variables:
           - 'random-color-generator_SELECTEDCOLOR'
   Configuration:
Referencing a GitHub output parameter

Use the following instructions to reference a GitHub output parameter.

**To reference a GitHub output parameter**

1. Complete the steps in Exporting a GitHub output parameter so that other actions can use it (p. 276).

   The GitHub output parameter is now available for use in other actions.

2. Note the output parameter's Variables value. It includes an underscore (_).

3. Refer to the output parameter using the following syntax:

   $\{action-name.output-name\}$

Replace:

- **action-name** with the name of the CodeCatalyst GitHub Action that produces the output parameter (do not use the GitHub action's name or id).
- **output-name** with the output parameter's Variables value you noted earlier.

**Example**

**BuildActionB:**

Identifier: aws/build@v1
Configuration:
Steps:
  - Run: echo $\{MyGitHubAction.random-color-generator_SELECTEDCOLOR\}

**Example with context**

The following example shows you how to set a SELECTEDCOLOR variable in GitHubActionA, output it, and then refer to it in BuildActionB.

**Actions:**

GitHubActionA:

Identifier: aws/github-actions-runner@v1
Configuration:
Steps:
  - name: Set selected color
    run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
    id: random-color-generator

Outputs:
  Variables:
    - 'random-color-generator_SELECTEDCOLOR'

BuildActionB:

Identifier: aws/build@v1
Configuration:
Steps:
  - Run: echo $\{GitHubActionA.random-color-generator_SELECTEDCOLOR\}
Removing an action

Use the following instructions to remove an action from a workflow.

**Visual**

To remove an action using the visual editor

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, in the action you want to remove, choose the vertical ellipsis icon, and choose Remove.
8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

**YAML**

To remove an action using the YAML editor

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. Find the section of the YAML that contains the action you want to remove.
   
   Select the section and press the delete key on your keyboard.
8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Developing a custom action

You can develop a custom action to use in your workflows using the CodeCatalyst Action Development Kit (ADK). You can then publish the action to the CodeCatalyst actions catalog, so that other CodeCatalyst users can view and use it in their workflows.

To develop, test, and publish an action (high-level tasks)

1. Install the required tools and packages necessary to develop an action.
2. Create a CodeCatalyst repository to store your action code.
3. Initialize the action. This lays down the source files required by the action, including an action definition file (action.yml) that you can update with your own code.
4. Bootstrap the action code to get the necessary tools and libraries to build, test, and release the action project.
5. Build the action on your local computer, and push the changes to your CodeCatalyst repository.
6. Test the action with unit tests locally, and run the ADK-generated workflow in CodeCatalyst.
7. Publish the action to the CodeCatalyst actions catalog by choosing the Publish button in the CodeCatalyst console.

For detailed steps, see the Amazon CodeCatalyst Action Development Kit Developer Guide.

### Grouping actions into action groups

An action group contains one or more actions. Grouping actions into action groups helps you keep your workflow organized, and also allows you to configure dependencies between different groups of actions.

**Note**
You cannot nest action groups within other actions or action groups.

Use the following instructions to define an action group.

**Visual**

Not available. Choose YAML to view the YAML instructions.

**YAML**

**To define an action group**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In Actions, add code similar to the following:

```yaml
Actions:
  action-group-name:
    Actions:
      action-1:
        Identifier: aws/build@v1
        Configuration:
        ...
      action-2:
        Identifier: aws/ecs-deploy@v1
        Configuration:
        ...
```

For another example, see Example: Defining two action groups (p. 279). For more information, see the description of the action-group-name property in the Actions (p. 363) of the Workflow definition reference (p. 354).

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

**Example: Defining two action groups**

The following example shows how to define two action groups: BuildAndTest and Deploy. The BuildAndTest action group includes two actions (Build and Test), and the Deploy action group also includes two actions (DeployCloudFormationStack and DeployToECS).
Actions:
BuildAndTest: # Action group 1
  Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
    ...
  Test:
    Identifier: aws/managed-test@v1
    Configuration:
Deploy: #Action group 2
  Actions:
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Configuration:
    ...
  DeployToECS:
    Identifier: aws/ecs-deploy@v1
    Configuration:
    ...

Configuring actions to depend on other actions

By default, when you add actions to a workflow, they are added side by side in the visual editor (p. 129). This means that the actions will run in parallel when you start a workflow run. If you want actions to run in sequential order (and appear vertically in the visual editor), you must set up dependencies between them. For example, you might set up a Test action to depend on the Build action so that the test action runs after the build action.

You can set up dependencies between actions, between action groups, and between actions and action groups. You can also configure one-to-many dependencies so that one action depends on several others in order to start. Consult the Guidelines for setting up dependencies (p. 281) to ensure your dependency setup conforms with the workflow’s YAML syntax.

Topics
- Setting up dependencies (p. 280)
- Guidelines for setting up dependencies (p. 281)
- Examples (p. 281)

Setting up dependencies

Use the following instructions to set up dependencies between actions in a workflow.

Visual

To set up dependencies using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action that will depend on another one.
8. Choose the Inputs tab.
9. In **Depends on - optional**, do the following:

   Specify an action or action group that must run successfully in order for this action to run.

   For more information about the 'depends on' functionality, see [Configuring actions to depend on other actions](p. 280).

10. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

11. Choose **Commit**, enter a commit message, and choose **Commit** again.

### YAML

**To set up dependencies using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. In an action that will depend on another, add code similar to the following:

   ```yaml
   action-name:
     DependsOn:
       - action-1
   ```

   For more examples, see [Examples](p. 281). For general guidelines, see [Guidelines for setting up dependencies](p. 281). For more information, see the description of the `DependsOn` property in the [Workflow definition reference](p. 354) for your action.

8. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

9. Choose **Commit**, enter a commit message, and choose **Commit** again.

### Guidelines for setting up dependencies

When configuring dependencies, follow these guidelines:

- If an action is inside an action group, that action can only depend on other actions within the same action group.

- Actions and action groups can depend on other actions and action groups *at the same level* in the YAML hierarchy, but not at a different level.

### Examples

The following examples show how to configure dependencies between actions and action groups in the workflow definition file.

#### Topics

- [Example: Configuring a simple dependency](p. 282)
- [Example: Configuring an action group to depend on an action](p. 282)
- [Example: Configuring an action group to depend on another action group](p. 282)
- [Example: Configuring an action group to depend on multiple actions](p. 282)
Example: Configuring a simple dependency

The following example shows how to configure a Test action to depend on the Build action using the DependsOn property.

Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      ...
  Test:
    DependsOn:
      - Build
    Identifier: aws/managed-test@v1
    Configuration:
      ...

Example: Configuring an action group to depend on an action

The following example shows how to configure a DeployGroup action group to depend on the FirstAction action. Notice that action and action group are at the same level.

Actions:
  FirstAction: #An action outside an action group
    Identifier: aws/github-actions-runner@v1
    Configuration:
      ...
  DeployGroup: #An action group containing two actions
    DependsOn:
      - FirstAction
    Actions:
      DeployAction1:
        ...
      DeployAction2:
        ...

Example: Configuring an action group to depend on another action group

The following example shows how to configure a DeployGroup action group to depend on the BuildAndTestGroup action group. Notice that the action groups are at the same level.

Actions:
  BuildAndTestGroup: # Action group 1
    Actions:
      BuildAction:
        ...
      TestAction:
        ...
  DeployGroup: #Action group 2
    DependsOn:
      - BuildAndTestGroup
    Actions:
      DeployAction1:
        ...
      DeployAction2:
        ...

Example: Configuring an action group to depend on multiple actions

The following example shows how to configure a DeployGroup action group to depend on the FirstAction action, the SecondAction action, as well as the BuildAndTestGroup action...
group. Notice that DeployGroup is at the same level as FirstAction, SecondAction, and BuildAndTestGroup.

<table>
<thead>
<tr>
<th>Actions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstAction: #An action outside an action group</td>
</tr>
<tr>
<td>SecondAction: #Another action</td>
</tr>
<tr>
<td>BuildAndTestGroup: #Action group 1</td>
</tr>
<tr>
<td>Actions:</td>
</tr>
<tr>
<td>Build:</td>
</tr>
<tr>
<td>Test:</td>
</tr>
<tr>
<td>DeployGroup: #Action group 2</td>
</tr>
<tr>
<td>DependsOn:</td>
</tr>
<tr>
<td>- FirstAction</td>
</tr>
<tr>
<td>- SecondAction</td>
</tr>
<tr>
<td>- BuildAndTestGroup</td>
</tr>
<tr>
<td>Actions:</td>
</tr>
<tr>
<td>DeployAction1:</td>
</tr>
<tr>
<td>DeployAction2:</td>
</tr>
</tbody>
</table>

**Viewing an action's source code**

You can view an action's source code to make sure it doesn't contain risky code, security vulnerabilities, or other defects.

Use the following instructions to view the source code of a [CodeCatalyst](p. 249), [CodeCatalyst Labs](p. 249), or [third-party](p. 250) action.

**Note**
To view the source code of a [GitHub Action](p. 249), go to the action's page in the [GitHub Marketplace](https://marketplace.github.com). The page includes a link to the action's repository, where you can find the action's source code.

**Note**
You cannot view the source code of the following CodeCatalyst actions: build (p. 138), test (p. 146), GitHub Actions (p. 272).

**Note**
AWS does not support or guarantee the action code of GitHub Actions or third-party actions.

**To view an action's source code**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. Find the action whose code you want to view:
   a. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
   b. Choose the name of any workflow, or create one. For information about creating a workflow, see [Creating, editing, and deleting a workflow](p. 246).
   c. Choose **Edit**.
   d. At the top-left, choose **+ Actions** to open the action catalog.
   e. In the drop-down list, choose Amazon CodeCatalyst to view CodeCatalyst, CodeCatalyst Labs, and third-party actions.
   f. Search for an action, and choose its name. Do not choose the plus sign (+).
Details about the action appear.

4. In the action details dialog box, near the bottom, choose Download.

A page appears, showing the Amazon S3 bucket where the action's source code resides. For information about Amazon S3, see What is Amazon S3? in the Amazon Simple Storage Service User Guide.

5. Inspect the code to ensure it meets your expectations for quality and security.

### Working with action versions

By default, when you add an action to a workflow, Amazon CodeCatalyst adds the full version to the workflow definition file using the format:

\`vmajor.minor.patch\`

For example:

```yaml
My-Build-Action:
  Identifier: aws/build@v1.0.0
```

You can shorten the full version in the Identifier property so that the workflow always uses the latest minor or patch version of the action.

For example, if you specify:

```yaml
My-CloudFormation-Action:
  Identifier: aws/cfn-deploy@v1.0
```

...and the latest patch version is 1.0.4, then the action will use 1.0.4. If a later version is released, say 1.0.5, then the action will use 1.0.5. If a minor version is released, say 1.1.0, then the action will continue to use 1.0.5.

For detailed instructions on specifying versions, see one of the following topics.

**Topics**

- [Specifying the action version to use (p. 284)](#)
- [Determining which versions of an action are available (p. 285)](#)

### Specifying the action version to use

Use the following instructions to indicate which version of an action you want your workflow to use. You can specify the latest major or minor version, or a specific patch version.

We recommend using the latest minor or patch version of an action.

**Visual**

*Not available. Choose YAML to view the YAML instructions.*

**YAML**

To configure a workflow to use the latest version of an action, or a specific patch version

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Choose **Edit**.

6. Choose **YAML**.

7. Find the action whose version you want to edit.

8. Find the action's **Identifier** property, and set the version to one of the following:

   - `action-identifier@v_major` – Use this syntax to have the workflow use a specific major version, and allow the latest minor and patch versions to be chosen automatically.
   - `action-identifier@v_major.minor` – Use this syntax to have the workflow use a specific minor version, and allow the latest patch version to be chosen automatically.
   - `action-identifier@v_major.minor.patch` – Use this syntax to have the workflow use a specific patch version.

   **Note**
   If you're not sure which versions are available, see Determining which versions of an action are available (p. 285).

   **Note**
   You cannot omit the major version.

9. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.

10. Choose **Commit**, enter a commit message, and choose **Commit** again.

---

**Determining which versions of an action are available**

Use the following instructions to determine which versions of an action are available for you to use in a workflow.

**Visual**

**To determine which action versions are available**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. Find the action whose versions you want to view:
   a. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
   b. Choose the name of any workflow, or create one. For information about creating a workflow, see Creating, editing, and deleting a workflow (p. 246).
   c. Choose **Edit**.
   d. At the top-left, choose **+ Actions** to open the action catalog.
   e. In the drop-down list, choose **Amazon CodeCatalyst** to view CodeCatalyst, CodeCatalyst Labs, and third-party actions, or choose **GitHub** to view curated GitHub Actions.
   f. Search for an action, and choose its name. Do not choose the plus sign (+).

   Details about the action appear.
4. In the action details dialog box, near the top-right, choose the **Versions** drop-down list to see a list of available versions of the action.

**YAML**

*Not available. Choose 'visual' to view the visual editor instructions.*
Working with artifacts

An artifact is the output of a workflow action, and typically consists of a folder or archive of files. Artifacts are important because they allow you to share files and information between actions.

For example, you might have a build action that generates a `sam-template.yml` file, but you want a deploy action to use it. In this scenario, you would use an artifact to allow the build action to share the `sam-template.yml` file with the deploy action. The code might look something like this:

```
Actions:
  BuildAction:
    Identifier: aws/build@v1
    Steps:
      - Run: sam package --output-template-file sam-template.yml
  Outputs:
    Artifacts:
      - Name: MYARTIFACT
        Files:
          - sam-template.yml
  DeployAction:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Artifacts:
        - MYARTIFACT
    Configuration:
      template: sam-template.yml
```

In the previous code, the build action (BuildAction) generates a `sam-template.yml` file, and then adds it to an output artifact called MYARTIFACT. A subsequent deploy action (DeployAction) specifies MYARTIFACT as an input, giving it access to the `sam-template.yml` file.

Topics

- Defining an output artifact (p. 286)
- Defining an input artifact (p. 288)
- Referencing files in an artifact (p. 289)
- Downloading artifacts (p. 290)
- Examples (p. 291)

Defining an output artifact

Use the following instructions to define an artifact that you want an action to output. This artifact then becomes available for other actions to use.

**Note**

Not all actions support output artifacts. To determine whether your action supports them, run through the visual editor instructions that follow, and see if the action includes an Output artifacts button on the Outputs tab. If yes, output artifacts are supported.

**Visual**

**To define an output artifact using the visual editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. In the workflow diagram, choose the action that will produce the artifact.
8. Choose the **Outputs** tab.
9. Under **Artifacts**, choose **Add artifact**.
10. Choose **Add artifact**, and enter information into the fields, as follows.

**Build artifact name**

Specify the name of an artifact generated by the action. Artifact names must be unique within a workflow, and are limited to alphanumeric characters (a-z, A-Z, 0-9) and underscores (_). Spaces, hyphens (-), and other special characters are not allowed. You cannot use quotation marks to enable spaces, hyphens, and other special characters in output artifact names.

For more information about artifacts, including examples, see [Working with artifacts](p. 286).

**Files produced by build**

Specify the files that CodeCatalyst includes in the artifact that is output by the action. These files are generated by the workflow action when it runs, and are also available in your source repository. File paths can reside in a source repository or an artifact from a previous action, and are relative to the source repository or artifact root. You can use glob patterns to specify paths. Examples:

- To specify a single file that is in the root of your build location or source repository location, use `my-file.jar`.
- To specify a single file in a subdirectory, use `directory/my-file.jar` or `directory/subdirectory/my-file.jar`.
- To specify all files, use `"**/*."`. The ** glob pattern indicates to match any number of subdirectories.
- To specify all files and directories in a directory named `directory`, use `"directory/**/*"`. The ** glob pattern indicates to match any number of subdirectories.
- To specify all files in a directory named `directory`, but not any of its subdirectories, use `"directory/*"`.

**Note**

If your file path includes one or more asterisks (*) or other special character, enclose the path with double quotation marks (""). For more information about special characters, see [Syntax guidelines and conventions](p. 355).

For more information about artifacts, including examples, see [Working with artifacts](p. 286).

**Note**

You may need to add a prefix to the file path to indicate which artifact or source to find it in. For more information, see [Referencing files in a source repository](p. 327) and [Referencing files in an artifact](p. 289).

11. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
12. Choose **Commit**, enter a commit message, and choose **Commit** again.

**YAML**

**To define an output artifact using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In a workflow action, add code similar to the following:

```
action-name:
  Outputs:
    Artifacts:
      - Name: artifact-name
        Files:
          - file-path-1
          - file-path-2
```

For more examples, see Examples (p. 291). For more information, see the Workflow definition reference (p. 354) for your action.

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

### Defining an input artifact

If you want to use an artifact generated by another action, you must specify it as an input to the current action. You may be able to specify multiple artifacts as input—it depends on the action. For more information, see the Workflow definition reference (p. 354) for your action.

**Note**

You cannot reference artifacts from other workflows.

Use the following instructions to specify an artifact from another action as input to the current action.

**Prerequisite**

Before you begin, make sure you have output the artifact from the other action. For more information, see Defining an output artifact (p. 286). Outputting the artifact makes it available for other actions to use.

**Visual**

**To specify an artifact as input to an action (visual editor)**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action where you want to specify an artifact as input.
8. Choose Inputs.
9. In Artifacts - optional, do the following:
Specify artifacts from previous actions that you want to provide as input to this action. These artifacts must already be defined as output artifacts in previous actions.

If you do not specify any input artifacts, then you must specify at least one source repository under `action-name/Inputs/Sources`.

For more information about artifacts, including examples, see [Working with artifacts](#).  

**Note**
If the **Artifacts - optional** drop-down list is unavailable (visual editor), or if you get errors in when you validate your YAML (YAML editor), it might be because the action only supports one input. In this case, try removing the source input.

10. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
11. Choose **Commit**, enter a commit message, and choose **Commit** again.

YAML

**To specify an artifact as input to an action (YAML editor)**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. In the action where you want to specify the artifact as input, add code similar to the following:

```yaml
action-name:
  Inputs:
    Artifacts:
    - artifact-name
```

For more examples, see [Examples](#).

8. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
9. Choose **Commit**, enter a commit message, and choose **Commit** again.

**Referencing files in an artifact**

If you have a file that resides within an artifact, and you need to refer to this file in one of your workflow actions, complete the following procedure.

**Note**
See also [Referencing files in a source repository](#).

**To reference files in an artifact**

- In the action where you want to reference a file, add code similar to the following:

```yaml
Actions:
  My-action:
    Inputs:
      Sources:
```
Working with artifacts

- WorkflowSource
  Artifacts:
    - artifact-name
  Configuration:
  Steps:
    - run: cd $CATALYST_SOURCE_DIR_artifact-name/build-output && cat file.txt

In the previous code, the action looks in the build-output directory in the artifact-name artifact to find and display the file.txt file.

For more examples, see Examples (p. 291).

**Note**
You may be able to omit $CATALYST_SOURCE_DIR_artifact-name/ prefix depending on how you've configured your action. For more information, see the following guidance.

**Guidance on how to refer to variables:**

- If your action includes only one item under Inputs (for example, it includes one input artifact and no source), then you can omit the prefix and specify just the file path relative to the artifact root.
- You can also omit the prefix if the file resides in the primary input. The primary input is either the WorkflowSource, or the first input artifact listed, if there is no WorkflowSource.
- The prefix can be different depending on the action you're using. For more information, see the following table.

<table>
<thead>
<tr>
<th>Action type</th>
<th>File path prefix to use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build action (p. 364), test action (p. 364)</td>
<td>$CATALYST_SOURCE_DIR_artifact-name/</td>
<td>$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt</td>
</tr>
<tr>
<td>All other actions</td>
<td>$CATALYST_SOURCE_DIR_artifact-name/ or /artifacts/current-action-name/artifact-name/ (this path is a symbolic link to $CATALYST_SOURCE_DIR_artifact-name/)</td>
<td>$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt or /artifacts/MyCurrentAction/MyArtifact/folder1/file.txt</td>
</tr>
</tbody>
</table>

**Downloading artifacts**

You can download and inspect artifacts generated by your workflow actions for troubleshooting purposes. There are two types of artifact you can download:

- **Source artifacts** – An artifact that contains a snapshot of the source repository content as it existed when the run started.
- **Workflow artifacts** – An artifact defined in the Outputs property of your workflow's configuration file.

**To download artifacts output by the workflow**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Under the workflow's name, choose Runs.
6. In Run history, in the Run ID column, choose a run. For example, Run-95a4d.
7. Under the run's name, choose Artifacts.
8. Next to an artifact, choose Download. An archive file downloads. Its file name consists of seven random characters.
9. Extract the archive using an archive extraction utility of your choice.

Examples

The following examples show how to output and reference artifacts in the workflow definition file.

Topics

- Example: Outputting an artifact (p. 291)
- Example: Inputting an artifact generated by another action (p. 291)
- Example: Referencing files in multiple artifacts (p. 292)
- Example: Referencing a file in a single artifact (p. 292)
- Example: Referencing a file in an artifact when a WorkflowSource is present (p. 293)

Example: Outputting an artifact

The following example shows how to output an artifact that includes two .jar files.

```
Actions:
Build:
  Identifier: aws/build@v1
  Outputs:
    Artifacts:
      - Name: ARTIFACT1
        Files:
          - build-output/file1.jar
          - build-output/file2.jar
```

Example: Inputting an artifact generated by another action

The following example shows you how to output an artifact called ARTIFACT4 in BuildActionA, and input it into BuildActionB.

```
Actions:
BuildActionA:
  Identifier: aws/build@v1
  Outputs:
    Artifacts:
      - Name: ARTIFACT4
        Files:
          - build-output/file1.jar
          - build-output/file2.jar
BuildActionB:
  Identifier: aws/build@v1
  Inputs:
    Artifacts:
      - ARTIFACT4
```
Configuration:

**Example: Referencing files in multiple artifacts**

The following example shows you how to output two artifacts named ART5 and ART6 in BuildActionC, and then reference two files named file5.txt (in artifact ART5) and file6.txt (in artifact ART6) in BuildActionD (under Steps).

*Note*
For more information on referencing files, see [Referencing files in an artifact (p. 289)](#).

*Note*
Although the example shows the `$CATALYST_SOURCE_DIR_ART5` prefix being used, you could omit it. This is because ART5 is the primary input. To learn more about the primary input, see [Referencing files in an artifact (p. 289)](#).

**Actions:**

**BuildActionC:**
- Identifier: aws/build@v1
  - Outputs:
    - Artifacts:
      - Name: ART5
        Files:
          - build-output/file5.txt
      - Name: ART6
        Files:
          - build-output/file6.txt

**BuildActionD:**
- Identifier: aws/build@v1
  - Inputs:
    - Artifacts:
      - ART5
      - ART6
  - Configuration:
    - Steps:
      - run: cd $CATALYST_SOURCE_DIR_ART5/build-output && cat file5.txt
      - run: cd $CATALYST_SOURCE_DIR_ART6/build-output && cat file6.txt

**Example: Referencing a file in a single artifact**

The following example shows you how to output one artifact named ART7 in BuildActionE, and then reference file7.txt (in artifact ART7) in BuildActionF (under Steps).

Notice how the reference does not require the `$CATALYST_SOURCE_DIR_ART7` prefix in front of the `build-output` directory as it did in [Example: Referencing files in multiple artifacts (p. 292)](#). This is because there is only one item specified under Inputs.

*Note*
For more information on referencing files, see [Referencing files in an artifact (p. 289)](#).

**Actions:**

**BuildActionE:**
- Identifier: aws/build@v1
  - Outputs:
    - Artifacts:
      - Name: ART7
        Files:
          - build-output/file7.txt

**BuildActionF:**
- Identifier: aws/build@v1
  - Inputs:
    - Artifacts:
Example: Referencing a file in an artifact when a WorkflowSource is present

The following example shows you how to output one artifact named ART8 in BuildActionG, and then reference file8.txt (in artifact ART8) in BuildActionH (under Steps).

Notice how the reference requires the $CATALYST_SOURCE_DIR_artifact-name prefix, as it did in Example: Referencing files in multiple artifacts (p. 292). This is because there are multiple items specified under Inputs (a source and an artifact), so you need the prefix to indicate where to look for the file.

Note
For more information on referencing files, see Referencing files in an artifact (p. 289).

Actions:
BuildActionG:
  Identifier: aws/build@v1
  Outputs:
    Artifacts:
      - Name: ART8
        Files:
          - build-output/file8.txt
BuildActionH:
  Identifier: aws/build@v1
  Inputs:
    Sources:
      - WorkflowSource
    Artifacts:
      - ART8
  Configuration:
    Steps:
      - run: cd $CATALYST_SOURCE_DIR_ART8/build-output && cat file8.txt

Working with compute and runtime environment Docker images

In a CodeCatalyst workflow, you can specify the compute and runtime environment image that CodeCatalyst uses to run a build or test action. Compute refers to the computing engine (the CPU, memory, and operating system) managed and maintained by CodeCatalyst to run your workflows. A runtime environment image is a container image that runs on top of your chosen compute platform, and includes extra tools that a workflow action might need, such as the AWS CLI. For more information, see CodeCatalyst workflows (p. 132).

Topics
- Working with compute (p. 293)
- Sharing compute across actions (p. 298)
- Working with runtime environment Docker images (p. 302)

Working with compute

Compute refers to the computing engine (the CPU, memory, and operating system) managed and maintained by CodeCatalyst to run your workflows. You associate a compute type with either your workflow or your workflow actions (p. 249).
Note
If compute is defined as a property of the workflow, then it can't be defined as a property of any action in that workflow. Similarly, if compute is defined as a property of any action, it can't be defined in the workflow.

Topics
- About compute types (p. 294)
- About compute fleets (p. 294)
- On-demand fleet properties (p. 295)
- Provisioned fleet properties (p. 295)
- Creating, editing, and deleting a provisioned fleet (p. 296)
- Assigning a provisioned fleet or on-demand compute to an action (p. 297)

About compute types
CodeCatalyst offers the following compute types:

- Amazon EC2
- AWS Lambda

Amazon EC2 offers optimized flexibility during action runs and Lambda offers optimized action start-up speeds. Lambda supports faster workflow action runs due to a lower start-up latency. Lambda allows you to run basic workflows that can build, test, and deploy serverless applications with common runtimes. These runtimes include Node.js, Python, Java, .NET, and Go. However, there are some use-cases which Lambda does not support, and if they impact you, use the Amazon EC2 compute type:

- Lambda doesn't support runtime environment images from a specified registry.
- Lambda doesn't support tools that require root permissions. For tools such as yum or rpm, use the Amazon EC2 compute type or other tools that don't require root permissions.
- Lambda doesn't support Docker builds or runs. The following actions that use Docker images are not supported: Deploy AWS CloudFormation stack, Deploy to Amazon ECS, Amazon S3 publish, AWS CDK bootstrap, AWS CDK deploy, AWS Lambda invoke, and GitHub Actions. Docker-based GitHub Actions that are running within CodeCatalyst GitHub Actions action are also not supported with Lambda compute. You can use alternatives that don't require root permissions, such as Podman.
- Lambda doesn't support writing to files outside /tmp. When configuring your workflow actions, you can reconfigure your tools to install or write to /tmp. If you have a build action that installs npm, make sure you configure it to install to /tmp.
- Lambda doesn't support runtimes longer than 15 minutes.

About compute fleets
CodeCatalyst offers the following compute fleets:

- On-demand fleets
- Provisioned fleets

With on-demand fleets, when a workflow action starts, the workflow provisions the resources it needs. The machines are destroyed when the action finishes. You only pay for the number of minutes that you're running your actions. On-demand fleets are fully managed, and includes automatic scaling capabilities to handle spikes in demand.
CodeCatalyst also offers provisioned fleets which contain machines powered by Amazon EC2 that are maintained by CodeCatalyst. With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. With provisioned fleets, your machines are always running and will incur costs as long they're provisioned.

In order to create, update, or delete a fleet, you must have the Space administrator role or the Project administrator role.

**On-demand fleet properties**

CodeCatalyst provides the following on-demand fleets:

<table>
<thead>
<tr>
<th>Name</th>
<th>Operating system</th>
<th>Architecture</th>
<th>vCPUs</th>
<th>Memory (GiB)</th>
<th>Disk space</th>
<th>Supported compute types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux.Arm64</td>
<td>Amazon Linux</td>
<td>Arm64</td>
<td>2</td>
<td>4</td>
<td>64 GB</td>
<td>Amazon EC2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 GB</td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lambda</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lambda</td>
</tr>
</tbody>
</table>

If no fleet is selected, CodeCatalyst uses Linux.x86-64.Large.

**Provisioned fleet properties**

A provisioned fleet contains the following properties:

**Operating system**

The operating system. The following operating systems are available:

- Amazon Linux 2
- Windows Server 2022

  **Note**

  Windows fleets are only supported in the build action. Other actions do not currently support Windows.

**Architecture**

The processor architecture. The following architectures are available:

- x86_64
- Arm64
Machine type

The machine type for each instance. The following machine types are available:

<table>
<thead>
<tr>
<th>vCPUs</th>
<th>Memory (GiB)</th>
<th>Disk space</th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>64 GB</td>
<td>Amazon Linux 2</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>128 GB</td>
<td>Amazon Linux 2</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>128 GB</td>
<td>Amazon Linux 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Windows Server 2022</td>
</tr>
</tbody>
</table>

Capacity

The initial number of machines allocated to the fleet, which defines the number of actions that can run in parallel.

Scaling mode

Defines the behavior when the number of actions exceeds the fleet capacity.

Provision additional capacity on demand

Additional machines are set up on demand which automatically scale up in response to new actions running, and then scale down to the base capacity as actions finish. This can incur additional costs, since you pay by the minute for each machine running.

Wait until additional fleet capacity is available

Action runs are placed in a queue until a machine is available. This limits additional costs because no additional machines are allocated.

Creating, editing, and deleting a provisioned fleet

Use the following instructions to create, edit, and delete provisioned fleets.

Note

Provisioned fleets will be deactivated after 2 weeks of inactivity. If used again, they will be re-activated automatically, but this re-activation may cause a latency to occur.

To create a provisioned fleet

1. In the navigation pane, choose CI/CD, and then choose Compute.
2. Choose Create provisioned fleet.
3. In the Provisioned fleet name text field, enter a name for your fleet.
4. From the Operating system drop-down menu, choose the operating system.
5. From the Machine type drop-down menu, choose the machine type for your machine.
6. In the Capacity text field, enter the maximum number of machines in the fleet.
7. From the Scaling mode drop-down menu, choose the desired overflow behavior. For more information about these fields, see Provisioned fleet properties (p. 295).
8. Choose Create.

After creating the provisioned fleet, you are ready to assign it to an action. For more information, see Assigning a provisioned fleet or on-demand compute to an action (p. 297).
To edit a provisioned fleet

1. In the navigation pane, choose CI/CD, and then choose Compute.
2. In the Provisioned fleet list, choose the fleet you want to edit.
3. Choose Edit.
4. In the Capacity text field, enter the maximum number of machines in the fleet.
5. From the Scaling mode drop-down menu, choose the desired overflow behavior. For more information about these fields, see Provisioned fleet properties (p. 295).
6. Choose Save.

To delete a provisioned fleet

Warning
Before deleting a provisioned fleet, remove it from all actions by deleting the Fleet property from the action's YAML code. Any action that continues to reference a provisioned fleet after it is deleted will fail the next time the action runs.

1. In the navigation pane, choose CI/CD, and then choose Compute.
2. In the Provisioned fleet list, choose the fleet you want to delete.
3. Choose Delete.
4. Enter delete to confirm the deletion.
5. Choose Delete.

Assigning a provisioned fleet or on-demand compute to an action

By default, workflow actions use the Linux.x86-64.Large on-demand fleet with an Amazon EC2 compute type. To use a provisioned fleet instead, or to use a different on-demand fleet, such as Linux.x86-64.2XLarge, use the following instructions.

Visual

Before you begin

- If you want to assign a provisioned fleet, you must first create the provisioned fleet. For more information, see Creating, editing, and deleting a provisioned fleet (p. 296).

To assign a provisioned fleet or different fleet type to an action

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action that you want to assign your provisioned fleet or new fleet type to.
8. Choose the Configuration tab.
9. In Compute fleet, do the following:
   
   Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines...
Working with compute and runtime environment Docker images

are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

10. (Optional) Choose Validate to validate the workflow's YAML code before committing.

11. Choose Commit, enter a commit message, and choose Commit again.

YAML

Before you begin

- If you want to assign a provisioned fleet, you must first create the provisioned fleet. For more information, see Creating, editing, and deleting a provisioned fleet (p. 296).

To assign a provisioned fleet or different fleet type to an action

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. Find the action that you want to assign your provisioned fleet or new fleet type to.
8. In the action, add a Compute property and set Fleet to the name of your fleet or on-demand fleet type. For more information, see the description of the Fleet property in the Build and test action reference (p. 364) for your action.
9. (Optional) Choose Validate to validate the workflow's YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

Sharing compute across actions

Topics
- Running multiple actions on shared compute (p. 299)
- Considerations for compute sharing (p. 299)
- Turning on compute sharing (p. 300)
- Examples (p. 301)

By default, actions in a workflow run on separate instances in a fleet (p. 294). This behavior provides actions with isolation and predictability on the state of inputs. The default behavior requires explicit configuration to share context such as files and variables between actions.

Compute sharing is a capability that allows you to run all the actions in a workflow on the same instance. Using compute sharing can provide faster workflow runtimes because less time is spent provisioning instances. You can also share files (artifacts) between actions without additional workflow configuration.
When a workflow is run using compute sharing, an instance in the default or specified fleet is reserved for the duration of all actions in that workflow. When the workflow run completes, the instance reservation is released.

**Running multiple actions on shared compute**

You can use the `Compute` attribute in the definition YAML at the workflow level to specify both the fleet and compute sharing properties of actions. You can also configure compute properties using the visual editor in CodeCatalyst. To specify a fleet, set the name of an existing fleet, set the compute type to EC2, and turn on compute sharing.

**Note**
Compute sharing is only supported if the compute type is set to EC2, and it's not supported for the Windows Server 2022 operating system. For more information about compute fleets, compute types, and properties, see [Working with compute](#).

**Note**
If you're on the Free tier and you specify the `Linux.x86-64.XLarge` or `Linux.x86-64.2XLarge` fleet manually in the workflow definition YAML, the action will still run on the default fleet (`Linux.x86-64.Large`). For more information about compute availability and pricing, see the [table for the tiers options](#).

When compute sharing is turned on, the folder containing the workflow source is automatically copied across actions. You don't need to configure output artifacts and reference them as input artifacts throughout a workflow definition (YAML file). As a workflow author, you need to wire up environment variables using inputs and outputs, just as you would without using compute sharing. If you want to share folders between actions outside the workflow source, consider file caching. For more information, see [Working with artifacts](#) and [Working with file caching](#).

The source repository where your workflow definition file resides is identified by the label WorkflowSource. While using compute sharing, the workflow source is downloaded in the first action that references it and automatically made available for subsequent actions in the workflow run to use. Any changes made to the folder containing the workflow source by an action, such as adding, modifying, or removing files, are also visible in the subsequent actions in the workflow. You can reference files that reside in the workflow source folder in any of your workflow actions, just as you can without using compute sharing. For more information, see [Referencing files in a source repository](#).

**Note**
Compute sharing workflows need to specify a strict sequence of actions, so parallel actions can't be set. While output artifacts can be configured at any action in the sequence, input artifacts aren't supported.

**Considerations for compute sharing**

You can run workflows with compute sharing in order to accelerate workflow runs and share context between actions in a workflow that use the same instance. Consider the following to determine whether using compute sharing is appropriate for your scenario:

<table>
<thead>
<tr>
<th>Compute sharing</th>
<th>Without compute sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute type</td>
<td>Amazon EC2</td>
</tr>
<tr>
<td>Instance provisioning</td>
<td>Actions run on same instance</td>
</tr>
<tr>
<td>Operating system</td>
<td>Amazon Linux 2</td>
</tr>
<tr>
<td>Referencing files</td>
<td>$CATALYST_SOURCE_DIR_WorkflowSource/</td>
</tr>
<tr>
<td>Compute sharing</td>
<td>Without compute sharing</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Workflow structure</td>
<td>Actions can only run sequentially</td>
</tr>
<tr>
<td>Accessing data across workflow actions</td>
<td>Access cached workflow source (WorkflowSource)</td>
</tr>
</tbody>
</table>

## Turning on compute sharing

Use the following instruction to turn on compute sharing for a workflow.

### Visual

**To turn on compute sharing using the visual editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow.
5. Choose Edit.
7. Choose Workflow properties.
8. From the Compute type dropdown menu, choose EC2.
9. (Optional) From the Compute fleet - optional dropdown menu, choose a fleet you want to use to run workflow actions. You can choose an on-demand fleet or create and choose a provisioned fleet. For more information, see [Creating, editing, and deleting a provisioned fleet (p. 296)](https://codecatalyst.aws/) and [Assigning a provisioned fleet or on-demand compute to an action (p. 297)](https://codecatalyst.aws/).
10. Switch the toggle to turn on compute sharing and have actions in the workflow run on the same fleet.
11. (Optional) Choose the run mode for the workflow. For more information, see [Configuring queued, superseded, and parallel runs (p. 317)](https://codecatalyst.aws/).
12. Choose Commit, enter a commit message, and choose Commit again.

### YAML

**To turn on compute sharing using the YAML editor**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow.
5. Choose Edit.
6. Choose YAML.
7. Turn on compute sharing setting the SharedInstance field to TRUE and Type to EC2. Set Fleet to a compute fleet you want to use to run workflow actions. You can choose an on-demand fleet or create and choose a provisioned fleet. For more information, see [Creating, editing, and deleting a provisioned fleet (p. 296)](https://codecatalyst.aws/) and [Assigning a provisioned fleet or on-demand compute to an action (p. 297)](https://codecatalyst.aws/).
In a workflow YAML, add code similar to the following:

```yaml
Name: MyWorkflow
SchemaVersion: "1.0"
Compute: # Define compute configuration.
  Type: EC2
  Fleet: MyFleet # Optionally, choose an on-demand or provisioned fleet.
  SharedInstance: true # Turn on compute sharing. Default is False.
Actions:
  BuildFirst:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
      Configuration:
        Steps:
          - Run: ...
```

8. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
9. Choose **Commit**, enter a commit message, and choose **Commit** again.

**Examples**

**Topics**
- Example: Amazon S3 Publish (p. 301)

**Example: Amazon S3 Publish**

The following workflow examples show how to perform the Amazon Amazon S3 Publish action in two ways: first using input artifacts and then using compute sharing. With compute sharing, the input artifacts aren't needed since you can access the cached WorkflowSource. Additionally, the output artifact in the Build action is no longer needed. The S3 Publish action is configured to use the explicit DependsOn property to maintain sequential actions; the Build action must run successfully in order for the S3 Publish action to run.

- Without compute sharing, you need to use input artifacts and share the outputs with subsequent actions:

```yaml
Name: S3PublishUsingInputArtifact
SchemaVersion: "1.0"
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ArtifactToPublish
          Files: [output.zip]
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    Inputs:
```

301
When using compute sharing by setting `SharedInstance` to `TRUE`, you can run multiple actions on the same instance and share artifacts by specifying a single workflow source. Input artifacts aren't required and can't be specified:

```
Name: S3PublishUsingComputeSharing
SchemaVersion: "1.0"
Compute:
  Type: EC2
  Fleet: dev-fleet
  SharedInstance: TRUE
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - Build
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
        - Name: dev-connection
      Configuration:
        SourcePath: output.zip
        DestinationBucketName: dev-bucket
```

**Working with runtime environment Docker images**

A *Docker image* includes the base operating system, the runtime, your application code, and all dependencies needed to run your workflow actions. The curated environment Docker image also includes extra tools that a workflow action might need, such as the AWS CLI, Node.js, and tar.

By default, actions use an image that is curated and maintained by CodeCatalyst. Only build and test actions support custom images. For more information, see Assigning a custom environment Docker image to an action (p. 304).

**Topics**

- Curated images (p. 303)
- What if the curated image doesn't include the tools I need? (p. 304)
- Assigning a custom environment Docker image to an action (p. 304)
Curated images

The following images are curated by CodeCatalyst for use with CodeCatalyst actions.

Note
Currently the only curated image is the default image. As the curated image is updated, additional versions will be published.

Topics
• Default image (p. 303)

Default image

If no image is specified, CodeCatalyst uses a curated image based on your selected compute type and contains tools that are maintained by CodeCatalyst.

Note
Node.js 18 is not yet available in the default image. If you'd like to use Node.js 18, use a custom environment Docker image with Amazon ECR (p. 306) or Docker Hub (p. 307).

This image contains the following tools:

Default image tool versions

<table>
<thead>
<tr>
<th>Tool</th>
<th>CodeCatalyst Amazon EC2 for Linux (x86_64)</th>
<th>CodeCatalyst Lambda for Linux (x86_64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CLI</td>
<td>2.13.9</td>
<td>2.13.9</td>
</tr>
<tr>
<td>AWS Copilot CLI</td>
<td>0.6.0</td>
<td>0.6.0</td>
</tr>
<tr>
<td>Docker</td>
<td>23.01</td>
<td>N/A</td>
</tr>
<tr>
<td>Docker Compose</td>
<td>2.16.0</td>
<td>N/A</td>
</tr>
<tr>
<td>Git</td>
<td>2.40.0</td>
<td>2.40.0</td>
</tr>
<tr>
<td>Go</td>
<td>1.20.2</td>
<td>1.20.2</td>
</tr>
<tr>
<td>Gradle</td>
<td>8.0.2</td>
<td>8.0.2</td>
</tr>
<tr>
<td>Java</td>
<td>Corretto17</td>
<td>Corretto17</td>
</tr>
<tr>
<td>Maven</td>
<td>3.9.4</td>
<td>3.9.4</td>
</tr>
<tr>
<td>Node.js</td>
<td>16.20.2</td>
<td>16.20.2</td>
</tr>
<tr>
<td>Python</td>
<td>3.9.15</td>
<td>3.9.15</td>
</tr>
<tr>
<td>pip</td>
<td>22.2.2</td>
<td>22.2.2</td>
</tr>
<tr>
<td>.NET</td>
<td>6.0.407</td>
<td>6.0.407</td>
</tr>
</tbody>
</table>

For more information about working with Amazon Linux 2, see the following:

• Amazon Linux 2 FAQs
How do I install a software package from the Extras Library on an EC2 instance running Amazon Linux 2?

Amazon Linux Docker images

What if the curated image doesn't include the tools I need?

If the curated image doesn't include the tools you need, you have a couple of options:

- You can provide a custom environment Docker image that includes the necessary tools. For more information, see Assigning a custom environment Docker image to an action (p. 304).

  **Note**
  
  If you want to provide a custom environment Docker image, make sure that you have Git on your local computer.

- You can have your workflow's build or test action install the tools you need.

For example, you could include the following instructions in the Steps section of the build or test action's YAML code:

```
Configuration:
  Steps:
    - Run: ./setup-script
```

The `setup-script` instruction would then run the following script to install the Node package manager (npm):

```
#!/usr/bin/env bash

echo "Setting up environment"

touch ~/.bashrc
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
source ~/.bashrc
nvm install v16.1.0
source ~/.bashrc
```

For more information about the build action YAML, see Build and test action reference (p. 364).

Assigning a custom environment Docker image to an action

If you don't want to use the Default image (p. 303), you can provide a custom environment Docker image. If you want to provide a custom environment Docker image, make sure that you have Git on your local computer. The image can reside in Docker Hub, Amazon Elastic Container Registry, or any public repository.

Use the following instructions to assign your environment Docker image to an action. After specifying an image, CodeCatalyst deploys it to your compute platform when the action starts.

**Note**

The following actions do not support custom environment Docker images: Deploy AWS CloudFormation stack, Deploy to ECS, and GitHub Actions. Custom environment Docker images also do not support the Lambda compute type.

Visual

To assign a custom environment Docker image using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
3. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
4. Choose **Edit**.
5. Choose **Visual**.
6. In the workflow diagram, choose the action that will use your environment Docker image.
7. Choose the **Configuration** tab.
8. Near the bottom, fill out the following fields.

**Runtime environment Docker image - optional**

Specify the registry where your image is stored. Valid values include:

- **Docker Hub** (visual editor) or **DockerHub** (YAML editor)
  
  The image is stored in the Docker Hub image registry.

- **Other registry** (visual editor) or **Other** (YAML editor)
  
  The image is stored in a custom image registry. Any publicly available registry can be used.

- **Amazon Elastic Container Registry** (visual editor) or **ECR** (YAML editor)
  
  The image is stored in an Amazon Elastic Container Registry image repository. To use an image in an Amazon ECR repository, this action needs access to Amazon ECR. To enable this access, you must create an [IAM role](https://docs.aws.amazon.com/IAM/latest/userguide/tutorial-create-iam-role.html) that includes the following permissions and custom trust policy. (You can modify an existing role to include the permissions and policy, if you want.)

  The IAM role must include the following permissions in its role policy:
  
  - `ecr:BatchCheckLayerAvailability`
  - `ecr:BatchGetImage`
  - `ecr:GetAuthorizationToken`
  - `ecr:GetDownloadUrlForLayer`

  The IAM role must include the following custom trust policy:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": [
            "codecatalyst-runner.amazonaws.com",
            "codecatalyst.amazonaws.com"
          ]
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  ```

  For more information about creating IAM roles, see [Creating a role using custom trust policies](https://docs.aws.amazon.com/IAM/latest/userguide/tutorial-create-iam-role.html) (console) in the **IAM User Guide**.

  Once you have created the role, you must assign it to the action through an environment. For more information, see [Associating an environment, account connection, and IAM role with a workflow action](#) (p. 309).
ECR image URL, Docker Hub image or Image URL

Specify one of the following:

- If you are using a custom registry, set the image to the value expected by the custom registry.
- If you are using a Docker Hub registry, set the image to the Docker Hub image name and optional tag.

Example: postgres:latest
- If you are using an Amazon ECR registry, set the image to the Amazon ECR registry URI.

Example: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo

9. (Optional) Choose Validate to validate the workflow's YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

YAML

To assign a custom environment Docker image using the YAML editor

1. In the navigation pane, choose CI/CD, and then choose Workflows.
2. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
3. Choose Edit.
4. Choose YAML.
5. Find the action that you want to assign a environment Docker image to.
6. In the action, add a Container section and underlying Registry and Image properties. For more information, see the description of the Container, Registry and Image properties in the Actions (p. 363) for your action.
7. (Optional) Choose Validate to validate the workflow's YAML code before committing.
8. Choose Commit, enter a commit message, and choose Commit again.

Examples

The following examples show how to assign a custom environment Docker image to an action in the workflow definition file.

Topics

- Example: Using a custom environment Docker image to add support for Node.js 18 with Amazon ECR (p. 306)
- Example: Using a custom environment Docker image to add support for Node.js 18 with Docker Hub (p. 307)

Example: Using a custom environment Docker image to add support for Node.js 18 with Amazon ECR

The following example shows how to use a custom environment Docker image to add support for Node.js 18 with Amazon ECR.

Configuration:

Container:
### Working with environments

An environment, not to be confused with a Dev Environment, is where code is deployed to. It usually contains an instance of a running application along with its associated infrastructure. You can give your environment a name such as development, test, staging, or production. Any deployments generated by CodeCatalyst to an environment will appear on the Environments page. To set up an environment, you give it a name, such as my-production-environment, and then associate it with your AWS account.

In addition to displaying deployment information, an environment also serves as the mechanism through which to assign AWS IAM roles to workflow actions.

**Can multiple environments exist within a single workflow?**

Yes. If a workflow includes multiple actions, each of those actions can be assigned an environment. For example, you could have a workflow that includes two deploy actions, where one is assigned a my-staging-environment environment and another is assigned a my-production-environment environment.

**Which actions support environments?**

The following actions support having their deployment information displayed on the Environments page:

- **Deploy AWS CloudFormation stack** – For more information, see Adding the "Deploy AWS CloudFormation stack" action (p. 228)
- **Deploy to Amazon ECS** – For more information, see Adding the "Deploy to Amazon ECS" action (p. 230)
- **Deploy to Kubernetes cluster** – For more information, see Adding the "Deploy to Kubernetes cluster" action (p. 231)
- **AWS CDK deploy** – For more information, see Adding the "AWS CDK deploy" action (p. 234)

**Note**

If you want to allow an action to access and perform operations in your AWS account, you'll need to associate it with an environment. Many actions support an environment association, including, but not limited to, the actions listed previously. You can tell which actions support an environment association because they'll include an Environment drop-down list on their Configuration tab in the visual editor (p. 129).

### Supported Regions

The Environments page can display resources in any AWS Region.
Are environments mandatory?

An environment is mandatory if the workflow action to which it is assigned deploys resources into the AWS cloud, or communicates with AWS services for other reasons (such as monitoring and reporting).

Creating an environment

Use the following instructions to create an empty environment that you can later associate with a workflow action.

Before you begin

You need the following:

- A CodeCatalyst space. For more information, see Setting up CodeCatalyst (p. 7).
- A CodeCatalyst project. For more information, see Creating a project with a blueprint (p. 58).
- An AWS account connection that includes the IAM roles your workflow action will need to access AWS. For more information, see Managing AWS accounts (p. 36).

Note
You can create an environment without an account connection; however, you will need to come back and add the connection later.

To create an environment

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Environments.
4. In Environment name, enter a name, such as Production or Staging.
5. In Environment type, select one of the following:
   - **Non-production** – An environment where you can test your application to make sure it’s working as intended before moving it into production.
   - **Production** – A ‘live’ environment that is publicly-available and hosts your finalized application.

If you choose Production, a Production badge appears in the UI next to any actions that the environment is associated with. The badge helps you quickly see which actions are deploying to production. Other than the appearance of the badge, there are no differences between production and non-production environments.

6. (Optional) In Description, enter a description such as Production environment for the hello-world app.
7. In Connection - optional, choose the AWS account connection you want to associate with this environment. Make sure the account connection includes the IAM roles that you want to associate with the environment. For more information about creating this connection, see Managing AWS accounts (p. 36).

Next steps

- Now that you have created an environment, you are ready to associate it with a workflow action. For more information, see Associating an environment, account connection, and IAM role with a workflow action (p. 309).
Associating an environment, account connection, and IAM role with a workflow action

When you associate an environment, account connection, and IAM role with a supported workflow action (p. 307), the IAM role becomes available for use by the action. In addition to gaining access to an IAM role, the action may also have its deployment information imported into the Environments page. For more information, see Which actions support environments? (p. 307).

Use the following instructions to associate an environment, account connection, and IAM role with an action.

Step 1: Associate the environment, account connection, and role to a workflow action

Use the following procedure to associate an environment, account connection, and role with a workflow action.

Visual

To associate an environment, account connection, and role with a workflow action using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose an action that is supported with environments. For more information, see Which actions support environments? (p. 307).
8. Choose the Configuration tab, and specify information into the fields, as follows.

Environment

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Account connection or Connection - optional (whichever is available)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Role

Specify the name of the IAM role that this action uses in order to access and operate in AWS services such as Amazon S3 and Amazon ECR. Make sure this role is added to your account connection. To add an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).
Note

You may be able to specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, provided it has sufficient permissions. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

If you don't see the role in the list, it's because you have not associated it to the account connection. For more information, see Adding IAM roles to account connections (p. 40).

9. (Optional) Choose Validate to validate the workflow's YAML code before committing.
10. Choose Commit, enter a commit message, and choose Commit again.

YAML

To associate an environment, account connection, and role with a workflow action using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In the workflow action that you want to associate with an environment, add code similar to the following:

   ```yaml
   action-name
   Environment:
     Name: environment-name
   Connections:
     - Name: account-connection-name
       Role: iam-role-name
   
   For more information, see the Workflow definition reference (p. 354) for your action.
   8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
   9. Choose Commit, enter a commit message, and choose Commit again.

Step 2: Populate the environment

After associating an environment, account connection, and role to workflow action, you can populate the Environments page with deployment information. Use the following instructions to populate the Environments page.

Note

The Environments page is supported by only a subset of workflow actions. For more information, see Which actions support environments? (p. 307).
To populate the environment

1. If a workflow run did not start automatically when you committed your changes in Step 1: Associate the environment, account connection, and role to a workflow action (p. 309), manually start a run as follows:
   a. In the navigation pane, choose CI/CD, and then choose Workflows.
   b. Choose the name of the workflow where you want to start a run. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
   c. Choose Run.

   The workflow run starts a new deployment, which causes CodeCatalyst to add your application resource information under Environments.

2. Verify that your application resources appear under your environment:
   a. In the navigation pane, choose CI/CD, and then choose Environments.
   b. Choose your environment (for example, Production).
   c. Choose the Deployment activity tab, and verify that a deployment appears with a Status of SUCCEEDED. This indicates that a workflow run successfully deployed your application resources.
   d. Choose the Deployment targets tab, and verify that your application resources appear.

Working with file caching

**Note**
File caching is only available with the Amazon CodeCatalyst build (p. 138) and test (p. 146) actions, and only when they are configured to use the EC2 compute type (p. 294).

When file caching is enabled, the build and test actions save on-disk files to a cache and restore them from that cache in subsequent workflow runs. Caching reduces the latency caused by building or downloading dependencies that haven't changed between runs. CodeCatalyst also supports fallback caches, which can be used to restore partial caches containing some of the needed dependencies. This helps reduce the latency impacts of a cache miss. CodeCatalyst caching is available for all actions except those which are configured to run on Lambda compute or on Windows.

**Note**
CodeCatalyst uses multilayered caching, which consists of a local cache and a remote cache. When provisioned fleets or on-demand machines encounter a cache miss on a local cache, dependencies will be restored from a remote cache. As a result, some action runs may experience latency from downloading a remote cache.

Topics
- About file caching (p. 311)
- Creating a cache (p. 312)
- Constraints (p. 313)

About file caching

File caching allows you to organize your data into multiple caches, which are each referenced under the FileCaching property. Each cache saves a directory specified by a given path. The specified directory will be restored in future workflow runs. The following is an example YAML snippet for caching with multiple caches named cacheKey1 and cacheKey2.
BuildMyNpmApp:
  Identifier: aws/build@v1
  Inputs:
    Sources:
      - WorkflowSource
  Configuration:
    Steps:
      - Run: npm install
      - Run: npm run test
  Caching:
    FileCaching:
      cacheKey1:
        Path: file1.txt
        RestoreKeys:
          - restoreKey1
      cacheKey2:
        Path: /root/repository
        RestoreKeys:
          - restoreKey2
          - restoreKey3

Codecatalyst applies cache access restrictions to ensure that an action in one workflow cannot modify the caches from a different workflow. This protects each workflow from others that might push incorrect data that impact builds or deployments. Restrictions are enforced with cache-scopes which isolate caches to every workflow and branch pairing. For example, workflow-A in branch feature-A has a different file cache than workflow-A in sibling branch feature-B.

Cache misses occur when a workflow looks for a specified file cache and is unable to find it. This can occur for multiple reasons, such as when a new branch is created or when a new cache is referenced and it hasn't been created yet. It can also occur when a cache expires, which by default occurs 14 days after it was last used. To mitigate cache misses and increase the rate of cache hits, CodeCatalyst supports fallback caches. Fallback caches are alternate caches and provide an opportunity to restore partial-caches, which can be an older version of a cache. A cache is restored by first searching for a match under FileCaching for the property name, and if not found, evaluates RestoreKeys. If there is a cache miss for both the property name and all RestoreKeys, the workflow will continue to run, as caching is best effort and not guaranteed.

**Creating a cache**

You can use the following instructions to add a cache to your workflow.

**Visual**

**To add a cache using the visual editor**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action where you want to add your cache.
8. Choose Configuration.
9. Under File caching - optional, choose Add cache and enter information into the fields, as follows:
Key

Specify the name of your primary cache property name. Cache property names must be unique within your workflow. Each action can have up to five entries in FileCaching.

Path

Specify the associated path for your cache.

Restore properties - optional

Specify the restore property to use as a fallback when the primary cache property can't be found. Restore property names must be unique within your workflow. Each cache can have up to five entries in RestoreKeys.

10. (Optional) Choose Validate to validate the workflow's YAML code before committing.

11. Choose Commit, enter a commit message, and then choose Commit again.

YAML

To add a cache using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In a workflow action, add code similar to the following:

```
action-name:
  Configuration:
    Steps: ...
  Caching:
    FileCaching:
      key-name:
        Path: file-path
        # Specify any additional fallback caches
        # RestoreKeys:
        #   - restore-key
```

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Constraints

The following are the constraints for the property name and RestoreKeys:

- Names must be unique within a workflow.
- Names are limited to alphanumeric characters (A-Z, a-z, 0-9), hyphens (-), and underscores (_).
- Names can have up to 180 characters.
- Each action can have up to five caches in FileCaching.
- Each cache can have up to five entries in RestoreKeys.
Working with runs

A run is a single iteration of a workflow. During a run, CodeCatalyst performs the actions defined in the workflow configuration file and outputs the associated logs, artifacts, and variables.

Topics

- Starting a workflow run (p. 314)
- Stopping a workflow run (p. 314)
- Viewing workflow run status and details (p. 315)
- Configuring queued, superseded, and parallel runs (p. 317)

Starting a workflow run

Use the following procedure to start a workflow run manually.

**Note**

You can also start a workflow run automatically by configuring a trigger (p. 327).

To start a workflow run manually

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of the workflow you want to run. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Run.

Stopping a workflow run

Use the following procedure to stop a workflow run that's in progress. You might want to stop a run if it was started by accident.

When you stop a workflow run, CodeCatalyst waits for in-progress actions to complete before it marks the run as Stopped in the CodeCatalyst console. Any actions that didn't have a chance to start will not be started, and will be marked as Abandoned.

**Note**

If a run is queued (that is, it has no in-progress actions), then the run is stopped immediately.

To stop a workflow run

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Under Workflows, choose Runs and choose the in-progress run from the list.
5. Choose Stop.
Viewing workflow run status and details

You can view the status and details of a single workflow run, or multiple runs at the same time.

Topics

- Workflow run statuses (p. 315)
- Viewing the status and details of a single run (p. 315)
- Viewing the status and details of all runs in your project (p. 316)
- Viewing the status and details of all runs of a specific workflow (p. 316)
- Viewing runs of a workflow in the workflow diagram (p. 317)

Workflow run statuses

A workflow run can have one of the following statuses:

- **Succeeded** – The workflow run was processed successfully.
- **Failed** – One or more actions in the workflow run failed.
- **In progress** – The workflow run is currently being processed.
- **Stopped** – A person stopped the workflow run while it was in progress.
- **Stopping** – The workflow run is currently being stopped.
- **Cancelled** – The workflow run was canceled by CodeCatalyst because the associated workflow was deleted or updated while the run was in progress.
- **Superseded** – Only occurs if you have configured superseded run mode (p. 320). The workflow run was canceled by CodeCatalyst because a later workflow run superseded it.

Viewing the status and details of a single run

You might want to view the status and details of a single workflow run to check whether it was successful, to see at what time it was completed, or to view who or what started it.

To view the status and details of a single run

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Under the workflow's name, choose Runs.
6. In Run history, in the Run ID column, choose a run. For example, Run-95a4d.
7. Under the run's name, do one of the following:

   - **Visual** to see a workflow diagram showing your workflow run's actions and their status (see Workflow run statuses (p. 315)). This view also shows the source repository and branch used during the run.
   
     In the workflow diagram, choose an action to see details such as logs, reports, and outputs generated by the action during the run. The information shown depends on which action type is selected. For more information about viewing build or deploy logs, see Viewing the results of a build action (p. 140) or Viewing the deployment logs (p. 239).
   
   - **YAML** to see the workflow definition file that was used for the run.
   
   - **Artifacts** to see the artifacts produced by the workflow run. For more information about artifacts, see Working with artifacts (p. 286).
• **Reports** to see the test reports and other types of reports produced by the workflow run. For more information about reports, see [Test report types](p. 147).

• **Variables** to see the output variables produced by the workflow run. For more information about variables, see [Working with variables](p. 337).

**Note**
If the run's parent workflow was deleted, a message indicating this fact appears at the top of the run details page.

---

**Viewing the status and details of all runs in your project**

You might want to view the status and details of all workflow runs within your project understand how much workflow activity is going on in your project, and learn about the overall health of your workflows.

**To view the status and details of all runs in your project**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Under **Workflows**, choose **Runs**.

All the runs, for all workflows, in all branches, across all repositories in your project, are displayed.

The page includes the following columns:

• **Run ID** – The unique identifier of the run. Choose the run ID link to view detailed information about the run.

• **Status** – The processing status of the workflow run. For more information about run statuses, see [Workflow run statuses](p. 315).

• **Trigger** – The person, commit, pull request (PR), or schedule that started the workflow run. For more information, see [Working with triggers](p. 327).

• **Workflow** – The name of the workflow for which a run was started, and the source repository and branch where the workflow definition file resides. You might need to expand the column width to see this information.

**Note**
If this column is set to Not available, it's usually because the associated workflow was deleted or moved.

• **Start time** – The time when the workflow run started.

• **Duration** – How long the workflow run took to process. Very long or very short durations might indicate problems.

• **End time** – The time when the workflow run ended.

---

**Viewing the status and details of all runs of a specific workflow**

You might want to view the status and details of all runs associated with a specific workflow to see if any runs are creating bottlenecks within the workflow, or to see which runs are currently in progress or have completed.

**To view the status and details of all runs of a specific workflow**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.

4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

5. Under the workflow’s name, choose **Runs**.

The runs associated with the chosen workflow appear.

The page is divided into two sections:

- **Active runs** – Displays runs that are in progress. These runs will be in one of the following states: *In progress*.
- **Run history** – Displays runs that have completed (that is, not in progress).

For more information about run statuses, see **Workflow run statuses** (p. 315).

### Viewing runs of a workflow in the workflow diagram

You can view the status of all runs of a workflow as they progress together through the workflow. The runs are displayed within the workflow diagram (as opposed to in a list view). This gives you a visual representation of which runs are being processed by which actions, and which runs are waiting in a queue.

#### To view the status of multiple runs as they progress together through a workflow

**Note**

This procedure only applies if your workflow is using the queued or superseded run mode. For more information, see **Configuring queued, superseded, and parallel runs** (p. 317).

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of the workflow that contains the runs you want to view. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

**Note**

Make sure you're looking at a workflow page and not a run page.

5. Choose the **Latest state** tab on the upper left.

A workflow diagram appears.

6. Review the workflow diagram. The diagram shows all the runs that are currently in progress within the workflow, and the latest runs that have finished. More specifically:

   - Runs that appear at the top, before **Sources**, are queued and waiting to start.
   - Runs that appear between actions are queued and waiting to be processed by the next action.
   - Runs that appear within an action are 1. currently being processed by the action, 2. have finished being processed by the action, or 3. were not processed by the action (usually because a previous action failed).

### Configuring queued, superseded, and parallel runs

By default, when multiple workflow runs occur at the same time, CodeCatalyst queues them up, and processes them one by one, in the order that they were started. You can change this default behavior by specifying a *run mode*. There are a few run modes:

- (Default) Queued run mode – CodeCatalyst processes runs one by one
• Superseded run mode – CodeCatalyst processes runs one by one, with newer runs overtaking older ones
• Parallel run mode – CodeCatalyst processes runs in parallel

Topics
• About queued run mode (p. 318)
• About superseded run mode (p. 320)
• About parallel run mode (p. 320)
• Changing the run mode (p. 321)

About queued run mode

In *queued run mode*, runs occur in series, with waiting runs forming a queue.

Queues form at the entry points to actions and action groups, so you can have *multiple queues within the same workflow* (see *Figure 1 (p. 318)*). When a queued run enters an action, the action is locked and no other runs can enter. When the run finishes and exits the action, the action becomes unlocked and ready for the next run.

*Figure 1 (p. 318)* illustrates a workflow configured in queued run mode. It shows:

• Seven runs working their way through the workflow.
• Two queues: one outside the entry to the input source (*Repo:main*), and one outside the entry to the *BuildTestActionGroup* action.
• Two locked blocks: the input source (*Repo:main*) and the *BuildTestActionGroup*.

Here’s how things will transpire as the workflow runs finish processing:

• When *Run-4d444* finishes cloning the source repository, it will exit the input source and join the queue behind *Run-3c333*. Then, *Run-5e555* will enter the input source.
• When *Run-1a111* finishes building and testing, it will exit the *BuildTestActionGroup* action and enter the *DeployAction* action. Then, *Run-2b222* will enter the *BuildTestActionGroup* action.

*Figure 1*: A workflow configured in 'queued run mode'
Use queued run mode if:

- **You want to keep a one-to-one relationship between features and runs – these features may be grouped when using superseded mode.** For example, when you merge feature 1 in commit 1, run 1 starts, and when you merge feature 2 in commit 2, run 2 starts, and so on. If you were to use superseded mode instead of queued mode, your features (and commits) will be grouped together in the run that supersedes the others.

- **You want to avoid race conditions and unexpected problems that may occur when using parallel mode.** For example, if two software developers, Wang and Saanvi, start workflow runs at roughly the same time to deploy to an Amazon ECS cluster, Wang's run might begin integration tests on the cluster while Saanvi's run deploys new application code to the cluster, causing Wang's tests either to fail or to test the wrong code. As another example, you might have a target that doesn't have a locking mechanism, in which case the two runs could overwrite each other's changes in unexpected ways.

- **You want to limit the load** on the compute resources that CodeCatalyst uses to process your runs. For example, if you have three actions in your workflow, you can have a maximum of three runs occurring...
at the same time. Imposing a limit on the number of runs that can occur at once makes run throughput more predictable.

- **You want to constrain the number of requests made to third-party services** by the workflow. For example, your workflow might have a build action that includes instructions to pull an image from Docker Hub. **Docker Hub limits the number of pull requests** you can make to a certain number per hour per account, and you will be locked out if you go over the limit. Using queued run mode to slow down your run throughput will have the effect of generating fewer requests to Docker Hub per hour, thus limiting the potential for lockouts and resulting build and run failures.

**Maximum queue size**: 50

Notes on **Maximum queue size**:

- The maximum queue size refers to the maximum number of runs allowed across all queues in the workflow.
- If a queue becomes longer than 50 runs, then CodeCatalyst drops the 51st and subsequent runs.

**Failure behavior**:

If a run becomes unresponsive while it's being processed by an action, then the runs behind it are held up in the queue until the action times out. Actions time out after an hour.

If a run fails inside an action, then the first queued run behind it is allowed to proceed.

**About superseded run mode**

*Superseded run mode* is the same as *queued run mode* except that:

- If a queued run catches up to another run in the queue, the later run supersedes (takes over from) the earlier run, and the earlier run is canceled and marked as 'superseded'.
- As an outcome of the behavior described in the first bullet, a queue can only include one run when superseded run mode is used.

Using the workflow in **Figure 1 (p. 318)** as a guide, applying superseded run mode to this workflow would result in the following:

- **Run-7g777** would supersede the other two runs in its queue, and would be the only run remaining in **Queue #1**. **Run-6f666** and **Run-5e555** would be canceled.
- **Run-3c333** would supersede **Run-2b222** and be the only run remaining in **Queue #2**. **Run-2b222** would be canceled.

Use superseded run mode if you want:

- better throughput than with queued mode
- even fewer requests into third-party services than with queued mode; this is advantageous if the third-party service has rate limits, such as Docker Hub

**About parallel run mode**

In *parallel run mode*, runs are independent of one another and don’t wait for other runs to complete before starting. There are no queues, and run throughput is limited only by how fast the actions inside the workflow take to complete.

Use parallel run mode in development environments where each user has their own feature branch and deploys to targets that are not shared by other users.
Important
If you have a shared target that multiple users can deploy to, such as a Lambda function in a production environment, do not use parallel mode, because race conditions may result. A race condition occurs when parallel workflow runs attempt to change a shared resource at the same time, leading to unpredictable results.

Maximum number of parallel runs: 1000 per CodeCatalyst space

Changing the run mode

You can set the run mode to queued, superseded, or parallel. The default is queued.

When you change the run mode from queued or superseded to parallel, CodeCatalyst cancels the runs that are queued, and allows the runs that are currently being processed by an action to finish before canceling them.

When you change the run mode from parallel to queued or superseded, CodeCatalyst lets all currently-running parallel runs complete. Any runs that you start after changing the run mode to queued or superseded use the new mode.

Visual

To change the run mode using the visual editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. At the top-right, choose Workflow properties.
7. Expand Advanced, and under Run mode, choose one of the following:
   a. Queued – see About queued run mode (p. 318)
   b. Superseded – see About superseded run mode (p. 320)
   c. Parallel – see About parallel run mode (p. 320)
8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

YAML

To change the run mode using the YAML editor

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. Add the RunMode property, like this:

   Name: Workflow_6d39
Working with secrets

There may be times when you need to use sensitive data, such as authentication credentials, in your workflows. Storing these values in plaintext anywhere in your repository should be avoided because anyone with access to the repository which contains the secret can see them. Similarly, these values shouldn't be used directly in any workflow definitions because they will be visible as files in your repository. With CodeCatalyst, you can protect these values by adding a secret to your project, and then referencing the secret in your workflow definition file.

Note
Secrets can only be used to replace passwords and sensitive information in the workflow definition file.

Topics
• Creating a secret (p. 322)
• Editing a secret (p. 323)
• Using a secret (p. 323)
• Deleting a secret (p. 324)

Creating a secret

Use the following procedure to create a secret. The secret contains the sensitive information that you want to hide from view.

Note
Secrets are visible to actions and are not masked when written to a file.

To create a secret
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the navigation pane, choose CI/CD, and then choose Secrets.
3. Choose Create secret.
4. Enter the following information:
   Name
   Enter a name for your secret.
   Value
   Enter the value for the secret. This is the sensitive information that you want to hide from view. By default, the value is not displayed. To display the value, choose Show value.
   Description
   (Optional) Enter a description for your secret.
5. Choose **Create**.

**Editing a secret**

Use the following procedure to edit a secret.

**To edit a secret**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose **CI/CD**, and then choose **Secrets**.
3. In the secrets list, choose the secret that you want to edit.
4. Choose **Edit**.
5. Edit the following properties:
   - **Value**: Enter the value for the secret. This is the value that you want to hide from view. By default, the value is not displayed.
   - **Description** (Optional): Enter a description for your secret.
6. Choose **Save**.

**Using a secret**

To use a secret in a workflow action, you must obtain the reference identifier of the secret and use that identifier in the workflow action.

**Topics**

- Obtaining the identifier of a secret (p. 323)
- Referencing a secret in a workflow (p. 323)

**Obtaining the identifier of a secret**

Use the following procedure to obtain the reference identifier of the secret. You'll add this identifier to your workflow.

**To obtain the reference identifier of the secret**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose **CI/CD**, and then choose **Secrets**.
3. In the list of secrets, find the secret that you want to use.
4. In the **Reference ID** column, copy the identifier of the secret. The following is the syntax for the Reference ID:

   ```
   ${Secrets.<name>}
   ```

**Referencing a secret in a workflow**

Use the following procedure to reference a secret in a workflow.
To reference a secret

1. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
2. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
3. Choose **Edit**.
4. Choose **YAML**.
5. Modify the YAML to use the identifier of the secret. For example, to use a user name and password that are stored as secrets with the `curl` command, you would use a Run command similar to the following:

   ```yaml
   - Run: curl -u <username-secret-identifier>:<password-secret-identifier> https://example.com
   ```

6. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
7. Choose **Commit**, enter a commit message, and choose **Commit** again.

Deleting a secret

Use the following procedure to delete a secret and the secret reference identifier.

**Note**
Before deleting a secret, we recommend that you remove the secret's reference identifier from all workflow actions. If you delete the secret without deleting the reference identifier, the action will fail the next time it runs.

**To delete a secret's reference identifier from a workflow**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
3. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
4. Choose **Edit**.
5. Choose **YAML**.
6. Search the workflow for the following string:

   ```
   ${Secrets.
   ```

   This finds all reference identifiers of all secrets.
7. Delete the reference identifier of the chosen secret, or replace it with a plaintext value.
8. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
9. Choose **Commit**, enter a commit message, and choose **Commit** again.

**To delete a secret**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the navigation pane, choose **CI/CD**, and then choose **Secrets**.
3. In the secrets list, choose the secret you want to delete.
4. Choose **Delete**.
5. Enter `delete` to confirm the deletion.
Working with sources

A source, also called an input source, is a source repository that a workflow action (p. 249) needs access to in order to carry out its tasks. For example, a workflow action might need to access a source to obtain unit tests and run them against your application source files.

CodeCatalyst workflows support the following sources:

- CodeCatalyst source repositories – For more information, see Source repositories in CodeCatalyst (p. 79).
- GitHub source repositories – For more information, see Using GitHub repositories in CodeCatalyst (p. 558).

Topics

- Specifying the source that will store the workflow definition file (p. 325)
- Specifying the source that a workflow action will use (p. 325)
- Referencing files in a source repository (p. 327)
- Variables produced by the source (p. 229)

Specifying the source that will store the workflow definition file

Use the following instructions to specify the CodeCatalyst source repository where you want to store your workflow definition file. If you’d rather specify a GitHub source repository, see instead Using GitHub repositories in CodeCatalyst (p. 558).

The source repository where your workflow definition file resides is identified by the label, WorkflowSource.

Note

You specify the source repository where your workflow definition file resides when you first commit your workflow definition file. After this commit, the repository and workflow definition file are linked together permanently. The only way to change the repository after the initial commit is to re-create the workflow in a different repository.

To specify the source repository that will store the workflow definition file

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose Create workflow and create the workflow. For more information, see To create a workflow using the visual editor (p. 246).

During the workflow creation process, you are asked to specify the CodeCatalyst repository where you want to store your workflow definition file.

Specifying the source that a workflow action will use

Use the following instructions to specify a source repository to use with a workflow action. On startup, the action bundles the files at the configured source repository into an artifact, downloads the artifact to the runtime environment Docker image (p. 302) where the action is running, and then completes its processing using the downloaded files.
**Note**
Currently, within a workflow action, you can only specify one source repository, which is the source repository where the workflow definition file resides (in the `.codecatalyst/workflows/` directory). This source repository is represented by the label `WorkflowSource`.

**Visual**

**To specify the source repository that an action will use (visual editor)**
1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action where you want to specify the source.
8. Choose Inputs.
9. In Sources - optional do the following:

   Specify the labels that represent the source repositories that will be needed by the action. Currently, the only supported label is `WorkflowSource`, which represents the source repository where your workflow definition file is stored.

   If you omit a source, then you must specify at least one input artifact under `action-name/Inputs/Artifacts`.

   For more information about sources, see Working with sources (p. 325).
10. (Optional) Choose Validate to validate the workflow's YAML code before committing.
11. Choose Commit, enter a commit message, and choose Commit again.

**YAML**

**To specify the source repository that an action will use (YAML editor)**
1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In an action, add code similar to the following:

   ```
   action-name:
   Inputs:
   Sources:
   - WorkflowSource
   ```

   For more information, see the description of the Sources property in Workflow definition reference (p. 354) for your action.
8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Referencing files in a source repository

If you have files that reside in a source repository, and you need to refer to these files in one of your workflow actions, complete the following procedure.

**Note**
See also Referencing files in an artifact (p. 289).

**To reference a file in a source repository**

- In the action where you want to reference a file, add code similar to the following:

```yaml
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - run: cd my-app && cat file1.jar
```

In the previous code, the action looks in the my-app directory in the root of the WorkflowSource source repository to find and display the file1.jar file.

Variables produced by the source

When a workflow runs, its source produces variables which you can use in subsequent workflow actions. For details, see “BranchName” and “CommitId” variables (p. 347) in the List of predefined variables (p. 346).

Working with triggers

A workflow trigger, or simply a trigger, allows you to start a workflow run automatically when certain events occur, like a code push. You might want to configure triggers to free your software developers from having to start workflow runs manually through the CodeCatalyst console.

You can use three types of trigger:

- **Push** – A code push trigger causes a workflow run to start whenever a commit is pushed. The workflow run uses the files in the branch that your pushing to (called, the destination branch).

- **Pull request** – A pull request trigger causes a workflow run to start whenever a pull request is either created, revised, or closed. The workflow run uses the files in the branch that you're pulling from (called the source branch).

- **Schedule** – A schedule trigger causes a workflow run to start on a schedule that you define. Consider using a schedule trigger to run nightly builds of your software so that the latest build is ready for your software developers to work on the next morning.

You can use push, pull request, and schedule triggers alone or in combination in the same workflow.

**Tip**
To see a trigger in action, launch a project with a blueprint. Most blueprints contain a workflow with a trigger. Look for the Trigger property in the blueprint's workflow definition file. For more information about blueprints, see Creating a project with a blueprint (p. 58).
A common trigger configuration

This section describes how to set up triggers for a common software release and branching strategy.

Software release and branching strategy:

- You have application code in a source repository.
- Your main branch contains finalized code that is always release-ready.
- Your software developers make their changes in development branches off the main branch, and prefix these branches with feature-12345, where 12345 is the ticket number associated with the change.
- Your software developers create a pull request (p. 101) asking to merge their feature branch into main when their feature is ready. The pull request includes a reviewer who must approve the request before the feature can be merged.
- After the approval, your software developers merge the pull request (p. 106).
- You have a Build and test workflow that builds and tests your application code. You want it to start automatically using triggers.

Proposed trigger configuration:

Given the software branching strategy outlined previously, you might want to add a trigger to your Build and test workflow definition file that looks like this:

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - feature-.*
    Events:
      - OPEN
      - REVISION
  - Type: PUSH
    Branches:
      - main
```

The previous trigger code automatically starts a workflow run whenever:

- A software developer creates a pull request (or modifies one (p. 105)) asking to merge their feature-12345 branch to the main branch.

  When a pull request occurs, the PULLREQUEST trigger is activated. CodeCatalyst starts a workflow run using the code on the feature-12345 branch. If the workflow run succeeds, the developer is assured that their feature works and can be merged to the main branch. If the workflow fails, the developer can commit fixes until the workflow run succeeds.

- A software developer merges a pull request from their feature-12345 branch to the main branch.

  When a merge to main occurs, the PUSH trigger is activated. CodeCatalyst starts a workflow run using the code in the main branch (which now includes the code from the pull request). If the workflow succeeds, the developer is assured that the application on the main branch is ready for deployment to a public location such as the Internet. If the workflow fails, the developer can revert the pull request merge and make fixes on their feature branch.
For instructions on adding triggers to a workflow definition file, see Adding a push, pull, or schedule trigger (p. 329).

For more examples of triggers and additional explanations, see Examples of triggers (p. 333).

Adding a push, pull, or schedule trigger

Use the following instructions to add a push, pull, or schedule trigger to your workflow.

**Visual**

**To add a trigger (visual editor)**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the Source and Triggers box.
8. In the configuration pane, Choose Add trigger.
9. In the Add trigger dialog box, supply information in the fields, as follows.

**Trigger type**

Specify the type of trigger. You can use one of the following values:

- **Push** (visual editor) or PUSH (YAML editor)

  A push trigger starts a workflow run when a change is pushed to your source repository. The workflow run will use the files in the branch that you're pushing to (that is, the destination branch).

- **Pull request** (visual editor) or PULLREQUEST (YAML editor)

  A pull request trigger starts a workflow run when a pull request is opened, updated, or closed in your source repository. The workflow run will use the files in the branch that you're pulling from (that is, the source branch).

- **Schedule** (visual editor) or SCHEDULE (YAML editor)

  A schedule trigger starts workflow runs on a schedule defined by a cron expression that you specify. A separate workflow run will start for each branch in your source repository using the branch's files. (To limit the branches that the trigger activates on, use the Branches field (visual editor) or Branches property (YAML editor).)

  When configuring a schedule trigger, follow these guidelines:
  - Only use one schedule trigger per workflow.
  - If you've defined multiple workflows in your CodeCatalyst space, we recommend that you schedule no more than 10 of them to start concurrently.
  - Make sure you configure the trigger's cron expression with adequate time between runs. For more information, see Expression (p. 361).

For examples, see Examples of triggers (p. 333).

**Events for pull request**
This field only appears if you selected the **Pull request** trigger type.

Specify the type of pull request events that will start a workflow run. The following are the valid values:

- **Pull request is created** (visual editor) or OPEN (YAML editor)
  
  The workflow run is started when a pull request is created.

- **Pull request is closed** (visual editor) or CLOSED (YAML editor)
  
  The workflow run is started when a pull request is closed. The CLOSED event's behavior is tricky, and is best understood through an example. See *Example: A trigger with a push, a pull, branches, and a 'CLOSED' event (p. 336)* for more information.

- **New revision is made to pull request** (visual editor) or REVISION (YAML editor)
  
  The workflow run is started when a revision to a pull request is created. The first revision is created when the pull request is created. After that, a new revision is created every time someone pushes a new commit to the source branch specified in the pull request. If you include the REVISION event in your pull request trigger, you can omit the OPEN event, since REVISION is a superset of OPEN.

You can specify multiple events in the same pull request trigger.

For examples, see *Examples of triggers (p. 333)*.

**Schedule**

This field only appears if you selected the **Schedule** trigger type.

Specify the cron expression that describes when you want your scheduled workflow runs to occur.

Cron expressions in CodeCatalyst use the following six-field syntax, where each field is separated by a space:

\[
\text{minutes} \ \text{hours} \ \text{days-of-month} \ \text{month} \ \text{days-of-week} \ \text{year}
\]

**Examples of cron expressions**

<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Days of month</th>
<th>Month</th>
<th>Days of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Runs a workflow at midnight (UTC+0) every Monday through Friday.</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>Runs a workflow at 2:00 am (UTC+0) every day.</td>
</tr>
</tbody>
</table>
When specifying cron expressions in CodeCatalyst, make sure you follow these guidelines:

- Specify a single cron expression per SCHEDULE trigger.
- Enclose the cron expression in double-quotes ("") in the YAML editor.
- Specify the time in Coordinated Universal Time (UTC). Other time zones are not supported.
- Configure at least 30 minutes between runs. A faster cadence is not supported.
- Specify the days-of-month or days-of-week field, but not both. If you specify a value or an asterisk (*) in one of the fields, you must use a question mark (?) in the other. The asterisk means 'all' and the question mark means 'any'.

For more examples of cron expressions and information about wildcards like ?, *, and L, see the Cron expressions reference in the Amazon EventBridge User Guide. Cron expressions in EventBridge and CodeCatalyst work exactly the same way.
For examples of schedule triggers, see Examples of triggers (p. 333).

Branches and Branch pattern

(Optional)

Specify the branches in your source repository that the trigger monitors in order to know when to start a workflow run. You can use regex patterns to define your branch names. For example, use main.* to match all branches beginning with main.

The branches to specify are different depending on the trigger type:

- For a push trigger, specify the branches you're pushing to, that is, the destination branches. One workflow run will start per matched branch, using the files in the matched branch.

Examples: main.*, mainline

- For a pull request trigger, specify the branches you're pulling from, that is, the source branches. One workflow run will start per matched branch, using the files in the matched branch.

Examples: feature\-.*, *(matches branches that start with feature\-), bugfix\/\.* *(matches branches that start with bugfix/)

- For a schedule trigger, specify the branches that contain the files that you want your scheduled run to use. One workflow run will start per matched branch, using the files in the matched branch.

Examples: main.*, version\-1\.0

Note
If you don't specify branches, the trigger monitors all branches, and will start a workflow run using the workflow definition file and source files in:

- The branch you're pushing to (for push triggers). For more information, see Example: A simple code push trigger (p. 334).
- The branch you're pulling from (for pull request triggers). For more information, see Example: A simple pull request trigger (p. 334).
- All branches (for schedule triggers). One workflow run will start per branch in your source repository. For more information, see Example: A simple schedule trigger (p. 335).

For more examples, see Examples of triggers (p. 333).

Files changed

This field only appears if you selected the Push or Pull request trigger type.

Specify the files or folders in your source repository that the trigger monitors in order to know when to start a workflow run. You can use regular expressions to match file names or paths.

For examples, see Examples of triggers (p. 333).

10. (Optional) Choose Validate to validate the workflow's YAML code before committing.

11. Choose Commit, enter a commit message, and choose Commit again.
YAML

To add a trigger (YAML editor)

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. Add a Triggers section and underlying properties using the following example as a guide. For more information, see the Triggers (p. 359) in the Workflow definition reference (p. 354).

A code push trigger might look like this:

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

A pull request trigger might look like this:

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - feature\/*
    Events:
      - OPEN
      - REVISION
      - CLOSED
```

A schedule trigger might look like this:

```
Triggers:
  - Type: SCHEDULE
    Branches:
      - main.*
    # Run the workflow at 1:15 am (UTC+0) every Friday until the end of 2023
    Expression: "15 1 ? * FRI 2022-2023"
```

For more examples of cron expressions you can use in the Expression property, see Expression (p. 361).

For more examples of push, pull request, and schedule triggers, see Examples of triggers (p. 333).

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Examples of triggers

The following examples show how to add different types of triggers in the workflow definition file.

Topics
- Example: A simple code push trigger (p. 334)
Example: A simple 'push to main' trigger

The following example shows a trigger that starts a workflow run whenever code is pushed to any branch in your source repository.

When this trigger is activated, CodeCatalyst starts a workflow run using the files in the branch that you're pushing to (that is, the destination branch).

For example, if you push a commit to main, CodeCatalyst starts a workflow run using the workflow definition file and other source files on main.

As another example, if you push a commit to feature-branch-123, CodeCatalyst starts a workflow run using the workflow definition file and other source files on feature-branch-123.

Triggers:
- Type: PUSH

Note
If you want a workflow run to start only when you push to main, see Example: A simple 'push to main' trigger (p. 334).

Example: A simple 'push to main' trigger

The following example shows a trigger that starts a workflow run whenever code is pushed to the main branch—and only the main branch—in your source repository.

Triggers:
- Type: PUSH
  Branches:
  - main

Example: A simple pull request trigger

The following example shows a trigger that starts a workflow run whenever a pull request is created or revised in your source repository.

When this trigger is activated, CodeCatalyst starts a workflow run using the workflow definition file and other source files in the branch that you're pulling from (that is, the source branch).

For example, if you create a pull request with a source branch called feature-123 and a destination branch called main, CodeCatalyst starts a workflow run using the workflow definition file and other source files on feature-123.

Triggers:
- Type: PULLREQUEST
  Events:
Example: A simple schedule trigger

The following example shows a trigger that starts a workflow run at midnight (UTC+0) every Monday through Friday.

When this trigger is activated, CodeCatalyst starts a single workflow run for each branch in your source repository that contains a workflow definition file with this trigger.

For example, if you have three branches in your source repository, `main`, `release-v1`, and `feature-123`, and each of these branches contains a workflow definition file with the trigger that follows, CodeCatalyst starts three workflow runs: one using the files in `main`, another using the files in `release-v1`, and another using the files in `feature-123`.

Triggers:
- Type: SCHEDULE
  Expression: "0 0 ? * MON-FRI *

For more examples of cron expressions you can use in the Expression property, see Expression (p. 361).

Example: A trigger with a schedule, and branches

The following example shows a trigger that starts a workflow run at 6:15 pm (UTC+0) every day.

When this trigger is activated, CodeCatalyst starts a workflow run using the files in the `main` branch, and starts additional runs for each branch that begins with `release-`.

For example, if you have branches named `main`, `release-v1`, `bugfix-1`, and `bugfix-2` in your source repository, CodeCatalyst starts two workflow runs: one using the the files in `main`, and another using the files in `release-v1`. It does not start workflow runs for the `bugfix-1` and `bugfix-2` branches.

Triggers:
- Type: SCHEDULE
  Expression: "15 18 * * ? *
  Branches:
    - main
    - release\-.*

For more examples of cron expressions you can use in the Expression property, see Expression (p. 361).

Example: A trigger with a schedule, a push, and branches

The following example shows a trigger that starts a workflow run at midnight (UTC+0) every day, and whenever code is pushed to the `main` branch.

In this example:

- A workflow run starts at midnight every day. The workflow run uses the workflow definition file and other source files in the `main` branch.
- A workflow run also starts whenever you push a commit to the `main` branch. The workflow run uses the workflow definition file and other source files in the destination branch (`main`).

Triggers:
Working with triggers

- Type: SCHEDULE
  Expression: "0 0 * * ? *"
  Branches:
    - main
- Type: PUSH
  Branches:
    - main

For more examples of cron expressions you can use in the Expression property, see Expression (p. 361).

Example: A trigger with a push, a pull, and branches

The following example shows a trigger that starts a workflow run whenever code is pushed to the main branch, or whenever a pull request is created from any branch that starts with feature-, bugfix-, release-, or docs-.

In this example:

- When you push a commit to the main branch, a workflow run starts automatically using the workflow definition file and other source files in the destination branch (main).
- When you create or modify a pull request with a source branch prefixed with feature-, bugfix-, release-, or docs-, a workflow run starts automatically using the workflow definition file and other source files in the source branch (feature-, bugfix-, release-, or docs-).
- When you merge a pull request from any branch to the main branch, a workflow run starts automatically using the workflow definition file and other source files in the main branch (because the merge commit activates the PUSH trigger).

Example: A trigger with a push, a pull, branches, and a 'CLOSED' event

The following example shows a trigger that starts a workflow run whenever code is pushed to the main branch, or whenever a pull request is closed on a branch that starts with feature-.

In this example:

- When you push a commit to the main branch, a workflow run starts automatically using the workflow definition file and other source files in the main branch.
- When you close a pull request from a source branch prefixed with feature-, a workflow run starts automatically using the workflow definition file and other source files in the (now closed) feature- branch.
- When you merge a pull request from any branch to the main branch, a workflow run starts automatically using the workflow definition file and other source files in the main branch (because the merge commit activates the PUSH trigger).
Note
If you've configured your source repository to delete branches automatically after a pull request is merged, these branches will never have the chance to enter the CLOSED state. This means that merged branches will not activate the pull request CLOSED trigger. The only way to activate the CLOSED trigger in this scenario is to close the pull request without merging it.

<table>
<thead>
<tr>
<th>Triggers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Type: PUSH</td>
<td></td>
</tr>
<tr>
<td>Branches:</td>
<td></td>
</tr>
<tr>
<td>- main</td>
<td></td>
</tr>
<tr>
<td>- Type: PULLREQUEST</td>
<td></td>
</tr>
<tr>
<td>Branches:</td>
<td></td>
</tr>
<tr>
<td>- feature-.*</td>
<td></td>
</tr>
<tr>
<td>Events:</td>
<td></td>
</tr>
<tr>
<td>- CLOSED</td>
<td></td>
</tr>
</tbody>
</table>

Example: A trigger with a push, branches, and files
The following example shows a trigger that starts a workflow run whenever a change is made to the filename.txt file, or any file in the src directory, on the main branch.

When this trigger is activated, CodeCatalyst starts a workflow run using the workflow definition file and other source files in the main branch.

<table>
<thead>
<tr>
<th>Triggers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Type: PUSH</td>
<td></td>
</tr>
<tr>
<td>Branches:</td>
<td></td>
</tr>
<tr>
<td>- main</td>
<td></td>
</tr>
<tr>
<td>FilesChanged:</td>
<td></td>
</tr>
<tr>
<td>- filename.txt</td>
<td></td>
</tr>
<tr>
<td>- src.*</td>
<td></td>
</tr>
</tbody>
</table>

Example: A trigger with a pull, branches, and files
The following example shows a trigger that starts a workflow run whenever someone creates or modifies a pull request that includes changes to the filename.txt file, or any file in the src directory. The .* directive under Branches indicates that the trigger must activate on pull requests from any branch. (You could also omit the Branches section to get the same result.)

When this trigger is activated, CodeCatalyst starts a workflow run using the workflow definition file and other source files in the branch that you're pulling from (that is, the source branch).

<table>
<thead>
<tr>
<th>Triggers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Type: PULLREQUEST</td>
<td></td>
</tr>
<tr>
<td>Events:</td>
<td></td>
</tr>
<tr>
<td>- OPEN</td>
<td></td>
</tr>
<tr>
<td>- REVISION</td>
<td></td>
</tr>
<tr>
<td>Branches:</td>
<td></td>
</tr>
<tr>
<td>- .*</td>
<td></td>
</tr>
<tr>
<td>FilesChanged:</td>
<td></td>
</tr>
<tr>
<td>- filename.txt</td>
<td></td>
</tr>
<tr>
<td>- src.*</td>
<td></td>
</tr>
</tbody>
</table>

Working with variables

A variable is a key-value pair that contains information that you can reference in your CodeCatalyst workflow.
There are two types of variable that you can use in a workflow:

- **User-defined variables** – These are key-value pairs that you define.
- **Predefined variables** – These are key-value pairs that are emitted by a workflow automatically. There is no need for you to define them.

**Note**

CodeCatalyst also supports GitHub output parameters, which behave like variables and can be referenced in other actions. For more information, see Working with GitHub Actions output parameters (p. 276).

**Topics**

- Using user-defined variables (p. 338)
- Using predefined variables (p. 346)

### Using user-defined variables

*User-defined variables* are key-value pairs that you define. There are two types:

- **Plaintext variables**, or simply **variables** – These are key-value pairs that you define in plaintext within the workflow definition file.
- **Secrets** – These are key-value pairs that you define on a separate Secrets page of the Amazon CodeCatalyst console. The *key* (name) is a public label, and the *value* contains the information you want to keep private. You only specify the key in the workflow definition file. Use secrets in place of passwords and other sensitive information in the workflow definition file.

**Note**

For brevity, this guide uses the term *variable* to mean *plaintext variable*.

**Topics**

- Defining a variable (p. 338)
- Defining a secret (p. 340)
- Exporting a variable so that other actions can use it (p. 341)
- Referencing a variable in the action that defines it (p. 342)
- Referencing a variable output by another action (p. 343)
- Referencing a secret (p. 344)
- Examples (p. 344)

### Defining a variable

You can define variables in two ways:

- In the Inputs section of a workflow action – see To define a variable in the "Inputs" section (p. 339)
- In the Steps section of a workflow action – see To define a variable in the "Steps" section (p. 340)

**Note**

The Steps method only works with the CodeCatalyst build, test, and GitHub Actions actions, because these are the only actions that include a Steps section.

For examples, see Examples (p. 344).
Visual

To define a variable in the "Inputs" section (visual editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action where you want to set the variable.
8. Choose Inputs.
9. In Variables - optional, choose Add variable, and then do the following:

   Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

   For more information about variables, including examples, see Working with variables (p. 337).

10. (Optional) Choose Validate to validate the workflow's YAML code before committing.
11. Choose Commit, enter a commit message, and choose Commit again.

YAML

To define a variable in the "Inputs" section (YAML editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In a workflow action, add code similar to the following:

```
action-name:
  Inputs:
  Variables:
    - Name: variable-name
      Value: variable-value
```

For more examples, see Examples (p. 344). For more information, see the Workflow definition reference (p. 354) for your action.

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.
Visual

**To define a variable in the "Steps" section (visual editor)**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. In the workflow diagram, choose the action where you want to set the variable.
8. Choose **Configuration**.
9. In **Shell commands** or **GitHub Actions YAML**, whichever is available, define a variable in the action's **Steps**, either explicitly or implicitly.
   - To define the variable explicitly, include it in a bash command directly to the **Steps** section.
   - To define a variable implicitly, specify it in a file that's referenced in the action's **Steps** section.

   For examples, see [Examples (p. 344)](#). For more information, see the [Workflow definition reference](#) for the action.
10. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
11. Choose **Commit**, enter a commit message, and choose **Commit** again.

YAML

**To define a variable in the "Steps" section (YAML editor)**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **YAML**.
7. In a workflow action, define a variable in the action's **Steps** section, either explicitly or implicitly.
   - To define the variable explicitly, include it in a bash command directly to the **Steps** section.
   - To define a variable implicitly, specify it in a file that's referenced in the action's **Steps** section.

   For examples, see [Examples (p. 344)](#). For more information, see the [Workflow definition reference](#) for the action.
8. (Optional) Choose **Validate** to validate the workflow's YAML code before committing.
9. Choose **Commit**, enter a commit message, and choose **Commit** again.

**Defining a secret**

You define a secret on the **Secrets** page of the CodeCatalyst console. For more information, see [Working with secrets (p. 322)](#).
For example, you might define a secret that looks like this:

- Name (key): `my-password`
- Value: `^*H3#!b9`

After the secret is defined, you can specify the secret's key (`my-password`) in the workflow definition file. For an example of how to do this, see Example: Referencing a secret (p. 346).

Exporting a variable so that other actions can use it

Use the following instructions to export a variable from an action so that you can reference it in other actions.

Before you export a variable, note the following:

- If you only need to reference the variable within the action where it's defined, then you don't need to export it.
- Not all actions support exporting variables. To determine whether your action supports this feature, run through the visual editor instructions that follow, and see if the action includes a Variables button on the Outputs tab. If yes, exporting variables is supported.
- To export a variable from a GitHub Action, see Exporting a GitHub output parameter so that other actions can use it (p. 276).

Prerequisite

Make sure you have defined the variable you want to export. For more information, see Defining a variable (p. 338).

Visual

To export a variable (visual editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
7. In the workflow diagram, choose the action that you want to export the variable from.
8. Choose Outputs.
9. In Variables - optional, choose Add variable, and then do the following:
   - Specify the name of a variable that you want the action to export. This variable must already be defined in the Inputs or Steps section of the same action.
10. (Optional) Choose Validate to validate the workflow's YAML code before committing.
11. Choose Commit, enter a commit message, and choose Commit again.

YAML

To export a variable (YAML editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In the action that you want to export the variable from, add code similar to the following:

```yaml
action-name:
  Outputs:
  Variables:
    - Name: variable-name
```

For more examples, see Examples (p. 344).
8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Referencing a variable in the action that defines it

Use the following instructions to reference a variable in the action that defines it.

**Note**

To reference a variable generated by a GitHub Action, see Referencing a GitHub output parameter (p. 277).

**Prerequisite**

Make sure you have defined the variable you want to reference. For more information, see Defining a variable (p. 338).

**Visual**

Not available. Choose YAML to view the YAML instructions.

**YAML**

To reference a variable in the action that defines it

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In the CodeCatalyst action that defines the variable that you want to refer to, add the variable using the following bash syntax:

```
$variable-name
```

For example:

MyAction:
Configuration:
Steps:
- Run: `$variable-name`

For more examples, see Examples (p. 344). For more information, see the reference information for your action in the Workflow definition reference (p. 354).

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.

Referencing a variable output by another action

Use the following instructions to reference variables output by other actions.

**Note**
To reference a variable output from a GitHub Action, see Referencing a GitHub output parameter (p. 277).

**Prerequisite**
Make sure you have exported the variable you want to reference. For more information, see Exporting a variable so that other actions can use it (p. 341).

**Visual**
Not available. Choose YAML to view the YAML instructions.

**YAML**

To reference a variable output by another action (YAML editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In the CodeCatalyst action, add a reference to the variable using the following syntax:

   ```yaml
   ${action-group-name.action-name.variable-name}
   ```

Replace:

- **action-group-name** with the name of the action group that contains the action that outputs variable.

  **Note**
  You can omit **action-group-name** if there is no action group, or if the variable is produced by an action in the same action group.

- **action-name** with the name of the action that outputs the variable.

- **variable-name** with the name of the variable.

For example:
MySecondAction:
  Configuration:
  Steps:
    - Run: ${MyFirstAction.TIMESTAMP}

For more examples, see Examples (p. 344). For more information, see the Workflow definition reference (p. 354) for your action.

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.

9. Choose Commit, enter a commit message, and choose Commit again.

Referencing a secret

For instructions on referencing a secret in the workflow definition file, see Using a secret (p. 323).

For an example, see Example: Referencing a secret (p. 346).

Examples

The following examples show how to define and reference variables in the workflow definition file.

Examples

- Example: Defining a variable using the Inputs property (p. 344)
- Example: Defining a variable using the Steps property (p. 344)
- Example: Exporting a variable using the Outputs property (p. 345)
- Example: Referencing a variable defined in the same action (p. 345)
- Example: Referencing a variable defined in another action (p. 345)
- Example: Referencing a secret (p. 346)

Example: Defining a variable using the Inputs property

The following example shows you how to define two variables, VAR1 and VAR2, in an Inputs section.

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: VAR1
          Value: "My variable 1"
        - Name: VAR2
          Value: "My variable 2"
```

Example: Defining a variable using the Steps property

The following example shows you how to define a DATE variable in the Steps section explicitly.

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
    Steps:
      - Run: DATE=$(date +%m-%d-%y)
```
Example: Exporting a variable using the Outputs property

The following example shows you how to define two variables, REPOSITORY-URI and TIMESTAMP, and export them using the Outputs section.

<table>
<thead>
<tr>
<th>Actions: Build:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: aws/build@v1</td>
<td></td>
</tr>
<tr>
<td>Inputs: Variables:</td>
<td></td>
</tr>
<tr>
<td>Name: REPOSITORY-URI</td>
<td></td>
</tr>
<tr>
<td>Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-repo</td>
<td></td>
</tr>
<tr>
<td>Configuration: Steps:</td>
<td></td>
</tr>
<tr>
<td>Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)</td>
<td></td>
</tr>
<tr>
<td>Outputs: Variables:</td>
<td></td>
</tr>
<tr>
<td>REPOSITORY-URI</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
</tr>
</tbody>
</table>

Example: Referencing a variable defined in the same action

The following example shows you how to specify a VAR1 variable in MyBuildAction, and then reference it in the same action using $VAR1.

<table>
<thead>
<tr>
<th>Actions: MyBuildAction:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: aws/build@v1</td>
<td></td>
</tr>
<tr>
<td>Inputs: Variables:</td>
<td></td>
</tr>
<tr>
<td>Name: VAR1</td>
<td></td>
</tr>
<tr>
<td>Value: my-value</td>
<td></td>
</tr>
<tr>
<td>Configuration: Steps:</td>
<td></td>
</tr>
<tr>
<td>Run: $VAR1</td>
<td></td>
</tr>
</tbody>
</table>

Example: Referencing a variable defined in another action

The following example shows you how to specify a TIMESTAMP variable in BuildActionA, export it using the Outputs property, and then reference it in BuildActionB using ${BuildActionA.TIMESTAMP}.

<table>
<thead>
<tr>
<th>Actions: BuildActionA:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: aws/build@v1</td>
<td></td>
</tr>
<tr>
<td>Configuration: Steps:</td>
<td></td>
</tr>
<tr>
<td>Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)</td>
<td></td>
</tr>
<tr>
<td>Outputs: Variables:</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BuildActionB:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: aws/build@v1</td>
<td></td>
</tr>
<tr>
<td>Configuration: Steps:</td>
<td></td>
</tr>
<tr>
<td>Run: docker build -t my-ecr-repo/image-repo:latest .</td>
<td></td>
</tr>
<tr>
<td>Run: docker tag my-ecr-repo/image-repo:latest ${BuildActionA.TIMESTAMP}</td>
<td></td>
</tr>
</tbody>
</table>

# Specifying just '$TIMESTAMP' here will not work
# because TIMESTAMP is not a variable
Example: Referencing a secret

The following example shows you how to reference a my-password secret. The my-password is the secret's key. This secret's key and corresponding password value must be specified on the Secrets page of the CodeCatalyst console prior to being used in the workflow definition file. For more information, see Working with secrets (p. 322).

Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: curl -u LiJuan:${Secrets.my-password} https://example.com

Using predefined variables

Predefined variables are key-value pairs that are emitted by a workflow automatically, and made available for you to use in workflow actions.

You can use predefined variables in any workflow action.

Topics

- List of predefined variables (p. 346)
- Referencing a predefined variable (p. 351)
- Determining which predefined variables your workflow emits (p. 353)
- Examples (p. 353)

List of predefined variables

Consult the following sections to view the predefined variables produced automatically by CodeCatalyst.

Note

This list only includes predefined variables emitted by CodeCatalyst workflows and CodeCatalyst actions (p. 249). If you're using other types of actions, such as GitHub Actions or CodeCatalyst Labs actions, see instead Determining which predefined variables your workflow emits (p. 353).

List

- "BranchName" and "CommitId" variables (p. 347)
- Build and test action variables (p. 347)
- "Amazon S3 publish" action variables (p. 347)
- "AWS CDK bootstrap" action variables (p. 347)
- "AWS CDK deploy" action variables (p. 348)
- "AWS Lambda invoke" action variables (p. 349)
- "Deploy AWS CloudFormation stack" action variables (p. 349)
- "Deploy to Amazon ECS" action variables (p. 350)
- "Deploy to Kubernetes cluster" action variables (p. 350)
- "Render Amazon ECS task definition" action variables (p. 351)
- "GitHub Actions" action variables (p. 351)
"BranchName" and "CommitId" variables

CodeCatalyst produces "BranchName" and "CommitId" variables when your workflow runs. See the following table for information about these variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommitId</td>
<td>The commit ID representing the state of the repository at the time the workflow run started. Example: example3819261db00a3ab594688c8b</td>
</tr>
<tr>
<td></td>
<td>See also: Example: Referencing the &quot;CommitId&quot; predefined variable (p. 353)</td>
</tr>
<tr>
<td>BranchName</td>
<td>The name of the branch against which the workflow run started. Examples: main, feature/branch, test-LiJuan</td>
</tr>
<tr>
<td></td>
<td>See also: Example: Referencing the &quot;BranchName&quot; predefined variable (p. 353)</td>
</tr>
</tbody>
</table>

Build and test action variables

The build and test actions do not produce variables automatically.

"Amazon S3 publish" action variables

The Amazon S3 publish action does not produce variables automatically.

"AWS CDK bootstrap" action variables

The AWS CDK bootstrap action produces the following predefined variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment-platform</td>
<td>The name of the deployment platform. Hardcoded to AWS:CloudFormation.</td>
</tr>
<tr>
<td>region</td>
<td>The region code of the AWS Region that the AWS CDK bootstrap stack was deployed to during the workflow run. Example: us-west-2</td>
</tr>
<tr>
<td>stack-id</td>
<td>The Amazon Resource Name (ARN) of the deployed AWS CDK bootstrap stack. Example: arn:aws:cloudformation:us-west-2:11112223333:stack/codecatalyst-cdk-bootstrap-stack/6aad4380-100a-11ec-a10a-03b8a84d40df</td>
</tr>
</tbody>
</table>
### Working with variables

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIP-DEPLOYMENT</td>
<td>A value of <code>true</code> indicates that deployment of your AWS CDK bootstrap stack was skipped during the workflow run. A stack deployment will be skipped if there is no change in the stack since the last deployment. This variable is only produced if its value is <code>true</code>. Hardcoded to <code>true</code>.</td>
</tr>
</tbody>
</table>

**"AWS CDK deploy" action variables**

The **AWS CDK deploy** action produces the following variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack-id</td>
<td>The Amazon Resource Name (ARN) of the AWS CDK application stack that was deployed to during the workflow run. Example: <code>arn:aws:cloudformation:us-west-2:11112223333:stack/codecatalyst-cdk-app-stack/6aad4380-100a-11ec-a10a-03b8a84d40df</code></td>
</tr>
<tr>
<td>deployment-platform</td>
<td>The name of the deployment platform. Hardcoded to AWS:CloudFormation.</td>
</tr>
<tr>
<td>region</td>
<td>The region code of the AWS Region that was deployed to during the workflow run. Example: <code>us-west-2</code></td>
</tr>
<tr>
<td>SKIP-DEPLOYMENT</td>
<td>A value of <code>true</code> indicates that deployment of your AWS CDK application stack was skipped during the workflow run. A stack deployment will be skipped if there is no change in the stack since the last deployment. This variable is only produced if its value is <code>true</code>. Hardcoded to <code>true</code>.</td>
</tr>
</tbody>
</table>

**AWS CloudFormation variables**

In addition to generating the variables listed previously, the **AWS CDK deploy** action also exposes CloudFormation output variables as workflow variables for use in subsequent workflow actions. By default, the action only exposes the first four (or fewer) CloudFormation variables that it finds. To determine which ones are exposed, run the **AWS CDK deploy** action once, and then look in the **Variables** tab of the run details page. If the variables listed on the **Variables** tab are not what you want, you can configure...
Working with variables

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>different ones</td>
<td>using the CfnOutputVariables YAML property. For more information, see the CfnOutputVariables (p. 401) property description in the &quot;AWS CDK deploy&quot; action reference (p. 393).</td>
</tr>
</tbody>
</table>

"AWS Lambda invoke" action variables

By default, the **AWS Lambda invoke** action produces one variable per top-level key in the Lambda response payload.

For example, if the response payload looks like this:

```python
responsePayload = {
    "name": "Saanvi",
    "location": "Seattle",
    "department": {
        "company": "Amazon",
        "team": "AWS"
    }
}
```

...then the action would generate the following variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Saanvi</td>
</tr>
<tr>
<td>location</td>
<td>Seattle</td>
</tr>
<tr>
<td>department</td>
<td>{&quot;company&quot;: &quot;Amazon&quot;, &quot;team&quot;: &quot;AWS&quot;}</td>
</tr>
</tbody>
</table>

**Note**

You can change which variables are generated using the ResponseFilters YAML property. For more information, see the ResponseFilters (p. 410) in the "AWS Lambda invoke" action reference (p. 402).

"Deploy AWS CloudFormation stack" action variables

The **Deploy AWS CloudFormation stack** action produces the following variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment-platform</td>
<td>The name of the deployment platform. Hardcoded to AWS:CloudFormation.</td>
</tr>
<tr>
<td>region</td>
<td>The region code of the AWS Region that was deployed to during the workflow run. Example: us-west-2</td>
</tr>
<tr>
<td>stack-id</td>
<td>The Amazon Resource Name (ARN) of the deployed stack.</td>
</tr>
</tbody>
</table>
"Deploy to Amazon ECS" action variables

The **Deploy to Amazon ECS** action produces the following variables.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster</td>
<td>The name of the Amazon ECS cluster that was deployed to during the workflow run.</td>
</tr>
<tr>
<td></td>
<td>Example: codecatalyst-ecs-cluster</td>
</tr>
<tr>
<td>deployment-platform</td>
<td>The name of the deployment platform. Hardcoded to AWS:ECS.</td>
</tr>
<tr>
<td>service</td>
<td>The name of the Amazon ECS service that was deployed to during the workflow run.</td>
</tr>
<tr>
<td></td>
<td>Example: codecatalyst-ecs-service</td>
</tr>
<tr>
<td>task-definition-arn</td>
<td>The Amazon Resource Name (ARN) of the task definition that was registered during the workflow run.</td>
</tr>
<tr>
<td></td>
<td>The :8 in the preceding example indicates the revision that was registered.</td>
</tr>
<tr>
<td>deployment-url</td>
<td>A link to the Amazon ECS console's Events tab, where you can view details of the Amazon ECS deployment associated with the workflow run.</td>
</tr>
<tr>
<td></td>
<td>Example: <a href="https://console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/codecatalyst-ecs-cluster/services/codecatalyst-ecs-service/events">https://console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/codecatalyst-ecs-cluster/services/codecatalyst-ecs-service/events</a></td>
</tr>
<tr>
<td>region</td>
<td>The region code of the AWS Region that was deployed to during the workflow run.</td>
</tr>
<tr>
<td></td>
<td>Example: us-west-2</td>
</tr>
</tbody>
</table>

"Deploy to Kubernetes cluster" action variables

The **Deploy to Kubernetes cluster** action produces the following variables.
### Key | Value
--- | ---
cluster | The Amazon.com Resource Name (ARN) of the Amazon EKS cluster that was deployed during the workflow run.  

deployment-platform | The name of the deployment platform.  
Hardcoded to AWS:EKS.

metadata | Reserved. JSON-formatted metadata related to the cluster deployed during the workflow run.

namespace | The Kubernetes namespace into which the cluster was deployed.  
Example: default

resources | Reserved. JSON-formatted metadata related to the resources deployed during the workflow run.

server | The name of the API server endpoint that you can use to communicate with your cluster using management tools such as kubectl.  
For more information about the API service endpoint, see [Amazon EKS cluster endpoint access control](https://docs.aws.amazon.com/eks/userguide) in the Amazon EKS User Guide.  
Example: https://random-string.gr7.us-west-2.eks.amazonaws.com

### "Render Amazon ECS task definition" action variables

The **Render Amazon ECS task definition** action produces the following variables.

### Key | Value
--- | ---
task-definition | The name given to the task definition file that was updated by the **Render Amazon ECS task definition** action. The name follows the format `task-definition-random-string`.  
Example: `task-definition-259-0a2r7gxlTF5Xr`

### "GitHub Actions" action variables

The **GitHub Actions** action does not produce variables automatically.

### Referencing a predefined variable

Use the following instructions to reference a predefined variable.
Prerequisite

Determine the name of the predefined variable you want to reference, such as CommitId. For more information, see Determining which predefined variables your workflow emits (p. 353).

Visual

Not available. Choose YAML to view the YAML instructions.

YAML

To reference a predefined variable (YAML editor)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In a CodeCatalyst action, add the predefined variable reference using the following syntax:

```
${action-group-name.action-name-or-WorkflowSource.variable-name}
```

Replace:

- `action-group-name` with the name of the action group.

  **Note**
  
  You can omit `action-group-name` if there is no action group, or if the variable is produced by an action in the same action group.

- `action-name-or-WorkflowSource` with:
  
  The name of the action that outputs the variable.
  
  or

  `WorkflowSource`, if the variable is the BranchName or CommitId variable.

- `variable-name` with the name of the variable.

For example:

```
MySecondAction:
  Configuration:
    Steps:
    - Run: echo ${MyFirstECSAction.cluster}
```

Another example:

```
MySecondAction:
  Configuration:
    Steps:
    - Run: echo ${WorkflowSource.CommitId}
```

For more examples, see Examples (p. 353). For more information, see the Workflow definition reference (p. 354) for your action.
Determining which predefined variables your workflow emits

You can determine which predefined variables your workflow emits in two ways:

- **Run the workflow once.** After the run finishes, the variables emitted by the workflow are displayed on the Variables tab of the run details page. For more information, see Viewing workflow run status and details (p. 315).
- **Consult the List of predefined variables (p. 346).** This reference lists the variable name (key) and value for each predefined variable.

**Note**
The maximum total size of a workflow's variables is listed in Quotas for workflows in CodeCatalyst (p. 457). If the total size exceeds the maximum, the action that occurs after the maximum is reached may fail.

Examples

The following examples show how to reference predefined variables in the workflow definition file.

**Examples**

- Example: Referencing the "CommitId" predefined variable (p. 353)
- Example: Referencing the "BranchName" predefined variable (p. 353)

Example: Referencing the "CommitId" predefined variable

The following example shows you how to refer to the CommitId predefined variable in the MyBuildAction action. The CommitId variable is output automatically by CodeCatalyst.

Although the example shows the variable being used in the build action, you can use CommitId in any action.

```yaml
MyBuildAction:
  Identifier: aws/build@v1
  Inputs:
    Sources:
    - WorkflowSource
  Configuration:
    Steps:
    #Build Docker image and tag it with a commit ID
    - Run: docker build -t image-repo/my-docker-image:latest .
    - Run: docker tag image-repo/my-docker-image:${[WorkflowSource.CommitId]}
```

Example: Referencing the "BranchName" predefined variable

The following example shows you how to refer to the BranchName predefined variable in the CDKDeploy action. The BranchName variable is output automatically by CodeCatalyst.

Although the example shows the variable being used in the AWS CDK deploy action, you can use BranchName in any action.

```yaml
CDKDeploy:
  Identifier: aws/cdk-deploy@v1
```
Workflow definition reference

The following is the reference documentation for the workflow definition file.

A workflow definition file is a YAML file that describes your workflow. The file is stored in a 
~/.codecatalyst/workflows/ folder in the root of your source repository (p. 89). The file can have a .yml or .yaml extension.

To create and edit the workflow definition file, you can use an editor such as vim, or you can use the CodeCatalyst console's visual editor or YAML editor. For more information, see Using the CodeCatalyst console's visual and YAML editors (p. 129).

**Note**

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

**Topics**

- Example of a workflow definition file (p. 354)
- Syntax guidelines and conventions (p. 355)
- Top-level properties (p. 356)
- Build and test action reference (p. 364)
- "Amazon S3 publish" action reference (p. 379)
- "AWS CDK bootstrap" action reference (p. 386)
- "AWS CDK deploy" action reference (p. 393)
- "AWS Lambda invoke" action reference (p. 402)
- "Deploy AWS CloudFormation stack" action reference (p. 411)
- "Deploy to Amazon ECS" action reference (p. 424)
- "Deploy to Kubernetes cluster" action reference (p. 432)
- "GitHub Actions" action reference (p. 439)
- "Render Amazon ECS task definition" action reference (p. 452)

**Example of a workflow definition file**

The following is an example of a simple workflow definition file. It includes a few top-level properties, a Triggers section, and an Actions section with two actions: Build and Test. For more information, see About the workflow definition file (p. 128).

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
```
Syntax guidelines and conventions

This section describes the syntax rules for the workflow definition file, as well as the naming conventions used in this reference documentation.

YAML syntax guidelines

The workflow definition file is written in YAML and follows the YAML 1.1 specification, so whatever is allowed in that specification is also allowed in the workflow YAML. If you’re new to YAML, here are some quick guidelines to ensure you’re supplying valid YAML code.

- **Case-sensitivity**: The workflow definition file is case-sensitive, so make sure you use the casing shown in this documentation.
- **Special characters**: We recommend using quotes or double-quotes around property values that include any of the following special characters: `{`, `}`, `[`, `]`, `,`, `;`, `&`, `*`, `#`, `?`, `|`, `<`, `>`, `=`, `!`, `%`, `@`, `:` and ``

  If you don’t include the quotes, the special characters listed previously may be interpreted in an unexpected way.

- **Property names**: Property names (as opposed to property values) are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotes or double-quotes to enable special characters and spaces in property names.

  Not permitted:

  'My#Build@action'

  My#Build@action

  My Build Action

  Permitted:

  My-Build-Action_1

- **Escape codes**: If your property value includes escape codes (for example, `\n` or `\t`), follow these guidelines:
  - Use single quotes to return the escape code as a string. For example, 'my string \n my string', returns the string my string \n my string.
  - Use double quotes to parse the escape code. For example, "my string \n my new line", returns: 355
Comments: Preface comments with #.

Example:

Name: MyWorkflow
# This is a comment.
SchemaVersion: 1.0

**Triple dash (---):** Do not use --- in your YAML code. CodeCatalyst ignores everything after the ---.

### Naming conventions

In this guide, we use the terms property and section to refer to the main items in a workflow definition file.

- A property is any item that includes a colon (:). For example, in the following code snippet, all of the following are properties: Name, SchemaVersion, RunMode, Triggers, Type, and Branches.
- A section is any property that has sub-properties. In the following code snippet, there is one Triggers section.

**Note**

In this guide, 'sections' are sometimes referred to as 'properties', and vise versa, depending on the context.

Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
  Branches:
    - main

### Top-level properties

The following is the reference documentation for the top-level properties in the workflow definition file.

```yaml
# Name
Name (p. 357): workflow-name

# Schema version
SchemaVersion (p. 357): 1.0

# Run mode
RunMode (p. 357): QUEUED|SUPERSEDED|PARALLEL

# Compute
Compute (p. 357):
...

# Triggers
Triggers (p. 359):
...

# Actions
```
Name
(Required)
The name of the workflow. The workflow name is shown in the workflows list and mentioned in
notifications and logs. The workflow name and workflow definition file name can match, or you can
name them differently. Workflow names do not need to be unique. Workflow names are limited to
alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You
cannot use quotation marks to enable special characters and spaces in workflow names.

Corresponding UI: visual editor/Workflow properties/Workflow name

SchemaVersion
(Required)
The schema version of the workflow definition. Currently, the only valid value is 1.0.

Corresponding UI: none

RunMode
(Optional)
How CodeCatalyst handles multiple runs. You can use one of the following values:

• QUEUED – Multiple runs are queued and run one after the other. You can have up to 50 runs in a queue.
• SUPERSEDED – Multiple runs are queued and run one after the other. A queue can only have one run,
so if two runs end up together in the same queue, the later run supersedes (takes over from) the earlier
run, and the earlier run is canceled.
• PARALLEL – Multiple runs occur simultaneously.

If this property is omitted, the default is QUEUED.

For more information, see Configuring queued, superseded, and parallel runs (p. 317).

Corresponding UI: visual editor/Workflow properties/Advanced/Run mode

Compute
(Optional)
The computing engine used to run your workflow actions. You can specify compute either at the
workflow level or at the action level, but not both. When specified at the workflow level, the compute
configuration applies to all actions defined in the workflow. At the workflow level, you can also
run multiple actions on the same instance. For more information, see Sharing compute across
actions (p. 298).

For more information about compute, see Working with compute and runtime environment Docker
images (p. 293).

Corresponding UI: none

Name: MyWorkflow
SchemaVersion: 1.0

... Compute (p. 357):
  Type (p. 358): EC2 | Lambda
  Fleet (p. 358): fleet-name
  SharedInstance (p. 358): true | false

**Type**

(Compute/Type)

(Required if Compute is set)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).

Corresponding UI: visual editor/Workflow properties/Advanced/Compute type

**Fleet**

(Compute/Fleet)

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

For more information about compute fleets, see About compute fleets (p. 294).

Corresponding UI: visual editor/Workflow properties/Advanced/Compute fleet

**SharedInstance**

(Compute/SharedInstance)

(Optional)

Specify the compute sharing capability for your actions. With compute sharing, actions in a workflow run on the same instance (runtime environment image). You can use one of the following values:

- **TRUE** means that the runtime environment image is shared between workflow actions.

- **FALSE** means that a separate runtime environment image is started and used for each action in a workflow, so you can't share resources such as artifacts and variables without extra configuration.
For more information about compute sharing, see Sharing compute across actions (p. 298).

Corresponding UI: none

**Triggers**

(Optional)

A sequence of one or more triggers for this workflow. If a trigger is not specified, then you must manually start your workflow.

For more information about triggers, see Working with triggers (p. 327).

Corresponding UI: visual editor/workflow diagram/Triggers

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Triggers (p. 359):
  ...  
  - Type (p. 359): PUSH
    Branches (p. 360):
      - branch-name
    FilesChanged (p. 361):
      - folder1/file
      - folder2/
  - Type (p. 359): PULLREQUEST
    Events (p. 360):
      - OPEN
      - CLOSED
      - REVISION
    Branches (p. 360):
      - branch-name
    FilesChanged (p. 361):
      - file1.txt
  - Type (p. 359): SCHEDULE
    # Run the workflow at 10:15 am (UTC+0) every Saturday
    Expression (p. 361) "15 10 ? * 7 *"
    Branches (p. 360):
      - branch-name
```

**Type**

(Triggers/Type)

(Required if Triggers is set)

Specify the type of trigger. You can use one of the following values:

- **Push** (visual editor) or PUSH (YAML editor)

  A push trigger starts a workflow run when a change is pushed to your source repository. The workflow run will use the files in the branch that you're pushing to (that is, the destination branch).

- **Pull request** (visual editor) or PULLREQUEST (YAML editor)

  A pull request trigger starts a workflow run when a pull request is opened, updated, or closed in your source repository. The workflow run will use the files in the branch that you're pulling from (that is, the source branch).

- **Schedule** (visual editor) or SCHEDULE (YAML editor)
A schedule trigger starts workflow runs on a schedule defined by a cron expression that you specify. A separate workflow run will start for each branch in your source repository using the branch's files. (To limit the branches that the trigger activates on, use the Branches field (visual editor) or Branches property (YAML editor).)

When configuring a schedule trigger, follow these guidelines:

- Only use one schedule trigger per workflow.
- If you've defined multiple workflows in your CodeCatalyst space, we recommend that you schedule no more than 10 of them to start concurrently.
- Make sure you configure the trigger's cron expression with adequate time between runs. For more information, see Expression (p. 361).

For examples, see Examples of triggers (p. 333).

Corresponding UI: visual editor/workflow diagram/Triggers/Trigger type

Events

(Triggers/Events)

(Required if the trigger Type is set to PULLREQUEST)

Specify the type of pull request events that will start a workflow run. The following are the valid values:

- **Pull request is created** (visual editor) or OPEN (YAML editor)
  
  The workflow run is started when a pull request is created.

- **Pull request is closed** (visual editor) or CLOSED (YAML editor)
  
  The workflow run is started when a pull request is closed. The CLOSED event's behavior is tricky, and is best understood through an example. See Example: A trigger with a push, a pull, branches, and a 'CLOSED' event (p. 336) for more information.

- **New revision is made to pull request** (visual editor) or REVISION (YAML editor)
  
  The workflow run is started when a revision to a pull request is created. The first revision is created when the pull request is created. After that, a new revision is created every time someone pushes a new commit to the source branch specified in the pull request. If you include the REVISION event in your pull request trigger, you can omit the OPEN event, since REVISION is a superset of OPEN.

You can specify multiple events in the same pull request trigger.

For examples, see Examples of triggers (p. 333).

Corresponding UI: visual editor/workflow diagram/Triggers/Events for pull request

Branches

(Triggers/Branches)

(Optional)

Specify the branches in your source repository that the trigger monitors in order to know when to start a workflow run. You can use regex patterns to define your branch names. For example, use main.* to match all branches beginning with main.

The branches to specify are different depending on the trigger type:
• For a push trigger, specify the branches you're pushing to, that is, the destination branches. One workflow run will start per matched branch, using the files in the matched branch.

   Examples: main.*, mainline

• For a pull request trigger, specify the branches you're pulling from, that is, the source branches. One workflow run will start per matched branch, using the files in the matched branch.

   Examples: feature\-.*, bugfix\/.* (matches branches that start with feature-), bugfix/ (matches branches that start with bugfix/)

• For a schedule trigger, specify the branches that contain the files that you want your scheduled run to use. One workflow run will start per matched branch, using the files in the matched branch.

   Examples: main.*, version\-1\.0

Note

If you don't specify branches, the trigger monitors all branches, and will start a workflow run using the workflow definition file and source files in:

• The branch you're pushing to (for push triggers). For more information, see Example: A simple code push trigger (p. 334).

• The branch you're pulling from (for pull request triggers). For more information, see Example: A simple pull request trigger (p. 334).

• All branches (for schedule triggers). One workflow run will start per branch in your source repository. For more information, see Example: A simple schedule trigger (p. 335).

For more examples, see Examples of triggers (p. 333).

Corresponding UI: visual editor/workflow diagram/Triggers/Branches

FilesChanged

(Triggers/FilesChanged)

(Optional if the trigger Type is set to PUSH, or PULLREQUEST. Not supported if the trigger Type is set to SCHEDULE.)

Specify the files or folders in your source repository that the trigger monitors in order to know when to start a workflow run. You can use regular expressions to match file names or paths.

For examples, see Examples of triggers (p. 333).

Corresponding UI: Examples of triggers (p. 333).

Expression

(Triggers/Expression)

(Required if the trigger Type is set to SCHEDULE)

Specify the cron expression that describes when you want your scheduled workflow runs to occur.

Cron expressions in CodeCatalyst use the following six-field syntax, where each field is separated by a space:

minutes hours days-of-month month days-of-week year

Examples of cron expressions
<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Days of month</th>
<th>Month</th>
<th>Days of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Runs a workflow at midnight (UTC+0) every Monday through Friday.</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Runs a workflow at 2:00 am (UTC+0) every day.</td>
</tr>
<tr>
<td>15</td>
<td>22</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Runs a workflow at 10:15 pm (UTC+0) every day.</td>
</tr>
<tr>
<td>0/30</td>
<td>22-2</td>
<td>?</td>
<td>*</td>
<td>SAT-SUN</td>
<td>*</td>
<td>Runs a workflow every 30 minutes Saturday through Sunday between 10:00 pm on the starting day and 2:00 am on the following day (UTC+0).</td>
</tr>
<tr>
<td>45</td>
<td>13</td>
<td>L</td>
<td>*</td>
<td>?</td>
<td>2023-2027</td>
<td>Runs a workflow at 1:45 pm (UTC+0) on the last day of the month between the years 2023 and 2027 inclusive.</td>
</tr>
</tbody>
</table>

When specifying cron expressions in CodeCatalyst, make sure you follow these guidelines:

- Specify a single cron expression per SCHEDULE trigger.
- Enclose the cron expression in double-quotes (""") in the YAML editor.
- Specify the time in Coordinated Universal Time (UTC). Other time zones are not supported.
• Configure at least 30 minutes between runs. A faster cadence is not supported.
• Specify the days-of-month or days-of-week field, but not both. If you specify a value or an asterisk (*) in one of the fields, you must use a question mark (?) in the other. The asterisk means 'all' and the question mark means 'any'.

For more examples of cron expressions and information about wildcards like ?, *, and L, see the Cron expressions reference in the Amazon EventBridge User Guide. Cron expressions in EventBridge and CodeCatalyst work exactly the same way.

For examples of schedule triggers, see Examples of triggers (p. 333).

Corresponding UI: visual editor/workflow diagram/Triggers/Schedule

Actions

A sequence of one or more actions for this workflow. CodeCatalyst supports several action types, such as build and test actions, which offer different types of functionality. Each action type has:

• an Identifier property that indicates the action's unique, hard-coded ID. For example, aws/build@v1 identifies the build action.
• a Configuration section that contains properties that are specific to the action.

For more information about each action type, see its reference documentation, available in the Workflow definition reference (p. 354).

The following is the YAML reference for actions and action groups in the workflow definition file.

For more information about actions, see Working with actions (p. 249).

Name: MyWorkflow
SchemaVersion: 1.0
...
Actions (p. 363):
  action-name (p. 363):
    Identifier: action-identifier
    Configuration:
      ...
    #Action groups
    action-group-name (p. 364):
      Actions:
        ...

action-name

(Actions/action-name)

(Required)

Replace action-name with a name you want to give the action. Action names must be unique within the workflow, and must only include alphanumeric characters, hyphens, and underscores. For more information about syntax rules, see YAML syntax guidelines (p. 355).

For more information about naming practices for actions, including restrictions, see the action-name (p. 363).

Corresponding UI: visual editor/action-name/Configuration tab/Action name or Action display name
action-group-name

(Optional)

An action group contains one or more actions. Grouping actions into action groups helps you keep your workflow organized, and also allows you to configure dependencies between different groups of actions.

Replace action-group-name with a name you want to give the action group. Action group names must be unique within the workflow, and must only include alphanumeric characters, hyphens, and underscores. For more information about syntax rules, see YAML syntax guidelines (p. 355).

For more information about action groups, see Grouping actions into action groups (p. 279).

Corresponding UI: none

Build and test action reference

The following is the action definition YAML reference for build and test actions. There is one reference for two actions because their YAML properties are very similar.

Choose a YAML property in the following code to see a description if it.

Note
Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
action-name (p. 365):
  Identifier (p. 366): aws/build@v1 | aws/managed-test@v1
  DependsOn (p. 366):
    - dependent-action-name-1
  Compute (p. 366):
    Type (p. 366): EC2 | Lambda
    Fleet (p. 367): fleet-name
    Timeout (p. 367): timeout-minutes
  Environment (p. 367):
    Name (p. 367): environment-name
  Connections (p. 368):
    - Name (p. 368): account-connection-name
    - Role (p. 368): iam-role-name
  Inputs (p. 368):
    Sources (p. 369):
      - source-name-1
      - source-name-2
    Artifacts (p. 369):
      - artifact-name
  Variables (p. 369):
    - Name: variable-name-1
      Value: variable-value-1
    - Name: variable-name-2
      Value: variable-value-2
  Outputs (p. 370):
    Artifacts (p. 370):
```
- **Name** (p. 370): *output-artifact-1*
  
  **Files** (p. 370):
  - build-output/artifact-1.jar
  - "build-output/build**"

- **Name** (p. 370): *output-artifact-2*
  
  **Files** (p. 370):
  - build-output/artifact-2.1.jar
  - build-output/artifact-2.2.jar

**Variables** (p. 371):
- *variable-name-1*
- *variable-name-2*

**AutoDiscoverReports** (p. 371):
- **Enabled** (p. 372): *true | false*
- **ReportNamePrefix** (p. 372): *AutoDiscovered*
- **IncludePaths** (p. 372):
  - "/**/"

**ExcludePaths** (p. 373):
- node_modules/cdk/junit.xml

**SuccessCriteria** (p. 373):
- **PassRate** (p. 373): *percent*
- **LineCoverage** (p. 374): *percent*
- **BranchCoverage** (p. 374): *percent*
- **Vulnerabilities** (p. 374):
  - **Severity**: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
  - **Number**: *whole-number*

**StaticAnalysisBug** (p. 375):
- **Severity**: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
- **Number**: *whole-number*

**StaticAnalysisSecurity** (p. 375):
- **Severity**: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
- **Number**: *whole-number*

**StaticAnalysisQuality** (p. 376):
- **Severity**: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
- **Number**: *whole-number*

**Reports** (p. 376):
- **report-name-1** (p. 376):
  - **Format** (p. 376): *format*
  - **IncludePaths** (p. 372):
    - "/**/"
  - **ExcludePaths** (p. 373):
    - report2.xml
    - report3.xml
  - SuccessCriteria (p. 373):
    - **PassRate** (p. 373): *percent*
    - **LineCoverage** (p. 374): *percent*
    - **BranchCoverage** (p. 374): *percent*
    - **Vulnerabilities** (p. 374):
      - **Severity**: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
      - **Number**: *whole-number*

**Configuration** (p. 377)
- **Container** (p. 377):
  - **Registry** (p. 377): *registry*
  - **Image** (p. 378): *image*

**Steps** (p. 379):
- Run: "step 1"
- Run: "step 2"

**action-name**

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.
Identifier

(action-name/Identifier)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Use aws/build@v1 for build actions.
Use aws/managed-test@v1 for test actions.

Corresponding UI: Workflow diagram/Action-name/aws/build@v1|aws/managed-test@v1 label

DependsOn

(action-name/DependsOn)

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the ‘depends on’ functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: none

Compute

(action-name/Compute)

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see Sharing compute across actions (p. 298).

Corresponding UI: none

Type

(action-name/Compute/Type)

(Required if Compute (p. 366) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).
Fleet

*(action-name/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see *On-demand fleet properties (p. 295)*.

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see *Provisioned fleet properties (p. 295)*.

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Compute type

Timeout

*(action-name/Timeout)*

(Optional)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in *Quotas for workflows in CodeCatalyst (p. 457)*. The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Compute fleet/Timeout - optional

Environment

*(action-name/Environment)*

(Optional)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see *Working with environments (p. 307)* and *Creating an environment (p. 308)*.

Corresponding UI: Configuration tab/Environment

Name

*(action-name/Environment/Name)*

(Optional)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment name
Connections

(action-name/Environment/Connections)
(Optional)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/

Name

(action-name/Environment/Connections/Name)
(Optional)

Specify the name of the account connection.

Corresponding UI: Configuration tab/

Role

(action-name/Environment/Connections/Role)
(Optional)

Specify the name of the IAM role that this action uses in order to access and operate in AWS services such as Amazon S3 and Amazon ECR. Make sure this role is added to your account connection. To add an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

Note
You may be able to specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, provided it has sufficient permissions. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Warning
Limit the permissions to those required by the build and test actions. Using a role with broader permissions might pose a security risk.

Corresponding UI: Configuration tab/

Inputs

(action-name/Inputs)
(Optional)

The Inputs section defines the data that an action needs during a workflow run.

Note
A maximum of four inputs (one source and three artifacts) are allowed per build action or test action. Variables do not count towards this total.
If you need to refer to files residing in different inputs (say a source and an artifact), the source input is the primary input, and the artifact is the secondary input. References to files in secondary inputs take a special prefix to distinguish them from the primary. For details, see Example: Referencing files in multiple artifacts (p. 292).

Corresponding UI: Inputs tab

**Sources**

*(action-name)/Inputs/Sources*

(Optional)

Specify the labels that represent the source repositories that will be needed by the action. Currently, the only supported label is WorkflowSource, which represents the source repository where your workflow definition file is stored.

If you omit a source, then you must specify at least one input artifact under action-name/Inputs/Artifacts.

For more information about sources, see Working with sources (p. 325).

Corresponding UI: none

**Artifacts - input**

*(action-name)/Inputs/Artifacts*

(Optional)

Specify artifacts from previous actions that you want to provide as input to this action. These artifacts must already be defined as output artifacts in previous actions.

If you do not specify any input artifacts, then you must specify at least one source repository under action-name/Inputs/Sources.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

**Note**

If the Artifacts - optional drop-down list is unavailable (visual editor), or if you get errors in when you validate your YAML (YAML editor), it might be because the action only supports one input. In this case, try removing the source input.

Corresponding UI: Inputs tab/Artifacts - optional

**Variables - input**

*(action-name)/Inputs/Variables*

(Optional)

Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Inputs tab/Variables - optional
Outputs

(action-name/Outputs)

(Optional)

Defines the data that is output by the action during a workflow run.

Corresponding UI: Outputs tab

Artifacts - output

(action-name/Outputs/Artifacts)

(Optional)

Specify the name of an artifact generated by the action. Artifact names must be unique within a workflow, and are limited to alphanumerics (a-z, A-Z, 0-9) and underscores (_). Spaces, hyphens (-), and other special characters are not allowed. You cannot use quotation marks to enable spaces, hyphens, and other special characters in output artifact names.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Outputs tab/Artifacts

Name

(action-name/Outputs/Artifacts/Name)

(Required if Artifacts - output (p. 370) is included)

Specify the name of an artifact generated by the action. Artifact names must be unique within a workflow, and are limited to alphanumerics (a-z, A-Z, 0-9) and underscores (_). Spaces, hyphens (-), and other special characters are not allowed. You cannot use quotation marks to enable spaces, hyphens, and other special characters in output artifact names.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Outputs tab/Artifacts/New output/Build artifact name

Files

(action-name/Outputs/Artifacts/Files)

(Required if Artifacts - output (p. 370) is included)

Specify the files that CodeCatalyst includes in the artifact that is output by the action. These files are generated by the workflow action when it runs, and are also available in your source repository. File paths can reside in a source repository or an artifact from a previous action, and are relative to the source repository or artifact root. You can use glob patterns to specify paths. Examples:

- To specify a single file that is in the root of your build location or source repository location, use my-file.jar.
- To specify a single file in a subdirectory, use directory/my-file.jar or directory/subdirectory/my-file.jar.
- To specify all files, use "**/*". The ** glob pattern indicates to match any number of subdirectories.
- To specify all files and directories in a directory named directory, use "directory/**/*". The ** glob pattern indicates to match any number of subdirectories.
• To specify all files in a directory named `directory`, but not any of its subdirectories, use "directory/**".

**Note**
If your file path includes one or more asterisks (*) or other special character, enclose the path with double quotation marks (""'). For more information about special characters, see Syntax guidelines and conventions (p. 355).

For more information about artifacts, including examples, see Working with artifacts (p. 286).

**Note**
You may need to add a prefix to the file path to indicate which artifact or source to find it in. For more information, see Referencing files in a source repository (p. 327) and Referencing files in an artifact (p. 289).

Corresponding UI: Outputs tab/Artifacts/New output/Files produced by build

**Variables - output**

*(action-name)/Outputs/Variables*

(Optional)
Specify the variables that you want the action to export so that they are available for use by subsequent actions.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Outputs tab/Variables/Add variable

**variable-name-1**

*(action-name)/Outputs/Variables*

(Optional)
Specify the name of a variable that you want the action to export. This variable must already be defined in the Inputs or Steps section of the same action.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Outputs tab/Variables/Add variable/Name

**AutoDiscoverReports**

*(action-name)/Outputs/AutoDiscoverReports*

(Optional)
Defines the configuration for the auto-discovery feature.

When you enable auto-discovery, CodeCatalyst searches all Inputs passed into the action as well as all files generated by the action itself, looking for test, code coverage, and software composition analysis (SCA) reports. For each report that is found, CodeCatalyst transforms it into a CodeCatalyst report. A CodeCatalyst report is a report that is fully integrated into the CodeCatalyst service and can be viewed and manipulated through the CodeCatalyst console.

**Note**
By default, the auto-discover feature inspects all files. You can limit which files are inspected using the IncludePaths (p. 372) or ExcludePaths (p. 373) properties.
Corresponding UI: Outputs tab/Reports/Auto-discover reports

**Enabled**

*(action-name)/Outputs/AutoDiscoverReports/Enabled*  
(Optional)  
Enable or disable the auto-discovery feature.  
Valid values are *true* or *false*.  
If Enabled is omitted, the default is *true*.  

Corresponding UI: Outputs tab/Reports/Auto-discover reports

**ReportNamePrefix**

*(action-name)/Outputs/AutoDiscoverReports/ReportNamePrefix*  
(Required if AutoDiscoverReports (p. 371) is included and enabled)  
Specify a prefix that CodeCatalyst prepends to all the reports it finds in order to name their associated CodeCatalyst reports. For example, if you specify a prefix of AutoDiscovered, and CodeCatalyst auto-discovers two test reports, TestSuiteOne.xml and TestSuiteTwo.xml, then the associated CodeCatalyst reports will be named AutoDiscoveredTestSuiteOne and AutoDiscoveredTestSuiteTwo.  
Corresponding UI: Outputs tab/Reports/Prefix name

**IncludePaths**

*(action-name)/Outputs/AutoDiscoverReports/IncludePaths*  
Or  
*(action-name)/Outputs/Reports/report-name-1/IncludePaths*  
(Required if AutoDiscoverReports (p. 371) is included and enabled, or if Reports (p. 376) is included)  
Specify the files and file paths that CodeCatalyst includes when searching for raw reports. For example, if you specify "/test/report/**", CodeCatalyst searches the entire *build image* (p. 302) used by the action looking for the /test/report/** directory. When it finds that directory, CodeCatalyst then looks for reports in that directory.  

**Note**  
If your file path includes one or more asterisks (*) or other special characters, enclose the path with double quotation marks (""). For more information about special characters, see Syntax guidelines and conventions (p. 355).  
If this property is omitted, the default is "**/**", meaning the search includes all files at all paths.  

**Note**  
For manually configured reports, IncludePaths must be a glob pattern that matches a single file.  

Corresponding UI:  
• Outputs tab/Reports/Auto-discover reports/include/exclude paths/Include paths

372
• Outputs tab/Reports/Manually configure reports/report-name-1/Include/exclude paths/Include paths

**ExcludePaths**

(action-name/Outputs/AutoDiscoverReports/ExcludePaths)

Or

(Outputs/Reports/report-name-1/ExcludePaths)

(Optional)

Specify the files and file paths that CodeCatalyst excludes when searching for raw reports. For example, if you specify "/test/my-reports/**/*", CodeCatalyst will not search for files in the /test/my-reports/ directory. To ignore all files in a directory, use the **/* glob pattern.

**Note**

If your file path includes one or more asterisks (*) or other special characters, enclose the path with double quotation marks (""). For more information about special characters, see Syntax guidelines and conventions (p. 355).

Corresponding UI:

• Outputs tab/Reports/Auto-discover reports/include/exclude paths/Exclude paths
• Outputs tab/Reports/Manually configure reports/report-name-1/Include/exclude paths/Exclude paths

**SuccessCriteria**

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria)

Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria)

(Optional)

Specify the success criteria for the test, code coverage, software composition analysis (SCA), and static analysis (SA) reports.

For more information, see Configuring success criteria for reports (p. 149).

Corresponding UI: Output tab/Reports/Success criteria

**PassRate**

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)

Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/PassRate)

(Optional)

Specify the percentage of tests in a test report that must pass for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. The pass rate criteria are applied only to test reports. For more information about test reports, see Test reports (p. 147).
Corresponding UI: Output tab/Reports/Success criteria/Pass rate

**LineCoverage**

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)

Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/LineCoverage)

(Optional)

Specify the percentage of lines in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. Line coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see [Code coverage reports](p. 147).

Corresponding UI: Output tab/Reports/Success criteria/Line coverage

**BranchCoverage**

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)

Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/BranchCoverage)

(Optional)

Specify the percentage of branches in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. Branch coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see [Code coverage reports](p. 147).

Corresponding UI: Output tab/Reports/Success criteria/Branch coverage

**Vulnerabilities**

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)

Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/Vulnerabilities)

(Optional)

Specify the maximum number and severity of vulnerabilities permitted in the SCA report for the associated CodeCatalyst report to be marked as passed. To specify vulnerabilities, you must specify:

- The minimum severity of the vulnerabilities you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

  For example, if you choose HIGH, then HIGH and CRITICAL vulnerabilities will be tallied.

- The maximum number of vulnerabilities of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Vulnerabilities criteria are applied only to SCA reports. For more information about SCA reports, see [Software composition analysis reports](p. 148).
To specify the minimum severity, use the `Severity` property. To specify the maximum number of vulnerabilities, use the `Number` property.

Corresponding UI: Output tab/Reports/Success criteria/Vulnerabilities

**StaticAnalysisBug**

```action-name/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisBug```

Or

```action-name/Outputs/Reports/report-name-1/SuccessCriteria/StaticAnalysisBug```

(Optional)

Specify the maximum number and severity of bugs permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify bugs, you must specify:

- The minimum severity of the bugs you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.
  
  For example, if you choose HIGH, then HIGH and CRITICAL bugs will be tallied.

- The maximum number of bugs of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Bugs criteria are applied only to SA reports. For more information about SA reports, see [Static analysis reports](p. 148).

To specify the minimum severity, use the `Severity` property. To specify the maximum number of vulnerabilities, use the `Number` property.

Corresponding UI: Output tab/Reports/Success criteria/Bugs

**StaticAnalysisSecurity**

```action-name/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisSecurity```

Or

```action-name/Outputs/Reports/report-name-1/SuccessCriteria/StaticAnalysisSecurity```

(Optional)

Specify the maximum number and severity of security vulnerabilities permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify security vulnerabilities, you must specify:

- The minimum severity of the security vulnerabilities you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.

  For example, if you choose HIGH, then HIGH and CRITICAL security vulnerabilities will be tallied.

- The maximum number of security vulnerabilities of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Security vulnerabilities criteria are applied only to SA reports. For more information about SA reports, see [Static analysis reports](p. 148).
To specify the minimum severity, use the `Severity` property. To specify the maximum number of vulnerabilities, use the `Number` property.

**StaticAnalysisQuality**

```
(action-name)/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisQuality
```

Or

```
(action-name)/Outputs/Reports/report-name-1/SuccessCriteria/StaticAnalysisQuality
```

(Optional)

Specify the maximum number and severity of quality issues permitted in the SA report for the associated CodeCatalyst report to be marked as passed. To specify quality issues, you must specify:

- The minimum severity of the quality issues you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.
  
  For example, if you choose HIGH, then HIGH and CRITICAL quality issues will be tallied.
- The maximum number of quality issues of the specified severity you want to permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Quality issues criteria are applied only to SA reports. For more information about SA reports, see Static analysis reports (p. 148).

To specify the minimum severity, use the `Severity` property. To specify the maximum number of vulnerabilities, use the `Number` property.

**Reports**

```
(action-name)/Outputs/Reports
```

(Optional)

A section that specifies the configuration for test reports.

Corresponding UI: Outputs tab/Reports

**report-name-1**

```
(action-name)/Outputs/Reports/report-name-1
```

(Required if Reports (p. 376) is included)

The name you want to give to the CodeCatalyst report that will be generated from your raw reports.

Corresponding UI: Outputs tab/Reports/Manually configure reports

**Format**

```
(action-name)/Outputs/Reports/report-name-1/Format
```

(Required if Reports (p. 376) is included)
Specify the file format that you're using for your reports. Possible values are as follows.

- For test reports:
  - For Cucumber JSON, specify **Cucumber** (visual editor) or **CUCUMBERJSON** (YAML editor).
  - For JUnit XML, specify **JUnit** (visual editor) or **JUNITXML** (YAML editor).
  - For NUnit XML, specify **NUnit** (visual editor) or **UNITXML** (YAML editor).
  - For NUnit 3 XML, specify **NUnit3** (visual editor) or **NUNIT3XML** (YAML editor).
  - For Visual Studio TRX, specify **Visual Studio TRX** (visual editor) or **VISUALSTUDIO_TRX** (YAML editor).
  - For TestNG XML, specify **TestNG** (visual editor) or **TESTNGXML** (YAML editor).

- For code coverage reports:
  - For Clover XML, specify **Clover** (visual editor) or **CLOVERXML** (YAML editor).
  - For Cobertura XML, specify **Cobertura** (visual editor) or **COBERTURAXML** (YAML editor).
  - For JaCoCo XML, specify **JaCoCo** (visual editor) or **JACOCOXML** (YAML editor).
  - For SimpleCov JSON generated by **simplecov**, not **simplecov-json**, specify **Simplecov** (visual editor) or **SIMPLECOV** (YAML editor).

- For software composition analysis (SCA) reports:
  - For SARIF, specify **SARIF** (visual editor) or **SARIFSCA** (YAML editor).

Corresponding UI: Outputs tab/Reports/Manually configure reports/Add/configure reports/report-name-1/Report type and Report format

**Configuration**

*(action-name/Configuration)*

(Required) A section where you can define the configuration properties of the action.

Corresponding UI: **Configuration** tab

**Container**

*(action-name/Configuration/Container)*

(Optional)

Specify the Docker image, or **container**, that the action uses to complete its processing. You can use the default image ([p. 303](#)), also called the default curated image, that is pre-packaged with CodeCatalyst, or you can use your own image. If you choose to use your own image, it can reside in Amazon ECR, Docker Hub, or another registry. If you don't specify a Docker image, the action uses the default image ([p. 303](#)) for its processing.

For more information about specifying your own Docker image, see [Assigning a custom environment Docker image to an action](#).

Corresponding UI: **Runtime environment Docker image - optional**

**Registry**

*(action-name/Configuration/Container/Registry)*

(Required if Container is included)

Specify the registry where your image is stored. Valid values include:
• **Docker Hub** (visual editor) or **DockerHub** (YAML editor)

The image is stored in the Docker Hub image registry.

• **Other registry** (visual editor) or **Other** (YAML editor)

The image is stored in a custom image registry. Any publicly available registry can be used.

• **Amazon Elastic Container Registry** (visual editor) or **ECR** (YAML editor)

The image is stored in an Amazon Elastic Container Registry image repository. To use an image in an Amazon ECR repository, this action needs access to Amazon ECR. To enable this access, you must create an **IAM role** that includes the following permissions and custom trust policy. (You can modify an existing role to include the permissions and policy, if you want.)

The IAM role must include the following permissions in its role policy:

- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:GetAuthorizationToken`
- `ecr:GetDownloadUrlForLayer`

The IAM role must include the following custom trust policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "codecatalyst-runner.amazonaws.com",
               "codecatalyst.amazonaws.com"
            ],
            "Action": "sts:AssumeRole"
         }
      }
   ]
}
```

For more information about creating IAM roles, see [Creating a role using custom trust policies (console)](console) in the **IAM User Guide**.

Once you have created the role, you must assign it to the action through an environment. For more information, see [Associating an environment, account connection, and IAM role with a workflow action](p. 309).

**Corresponding UI:** **Amazon Elastic Container Registry, Docker Hub, and Other registry** options

**Image**

*(action-name/Configuration/Container/Image)*

(Required if Container is included)

Specify one of the following:

- If you are using a custom registry, set the image to the value expected by the custom registry.
- If you are using a Docker Hub registry, set the image to the Docker Hub image name and optional tag.
Example: postgres:latest

- If you are using an Amazon ECR registry, set the image to the Amazon ECR registry URI.

Example: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo

Corresponding UI: **Image URL**

**Steps**

(*action-name*/Configuration/Steps)

(Required)

Specify the shell commands that you want to run during the action to install, configure, and run your build tools.

Here is an example of how to build an npm project:

<table>
<thead>
<tr>
<th>Steps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Run: npm install</td>
<td></td>
</tr>
<tr>
<td>Run: npm run build</td>
<td></td>
</tr>
</tbody>
</table>

Here is an example of how to specify file paths:

<table>
<thead>
<tr>
<th>Steps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Run: cd $ACTION_BUILD_SOURCE_PATH_WorkflowSource/app &amp;&amp; cat file2.txt</td>
<td></td>
</tr>
<tr>
<td>Run: cd $ACTION_BUILD_SOURCE_PATH_MyBuildArtifact/build-output/ &amp;&amp; cat file.txt</td>
<td></td>
</tr>
</tbody>
</table>

For more information about specifying file paths, see [Referencing files in a source repository](p. 327) and [Referencing files in an artifact](p. 289).

Corresponding UI: Configuration tab/Shell commands

"Amazon S3 publish" action reference

The following is the action definition YAML reference for the Amazon S3 publish action. To learn how to use this action, see [Adding the "Amazon S3 publish" action](p. 251).

**Note**

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use **Ctrl+F**. The element will be listed with its associated YAML property.

```yaml
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
S3Publish nn (p. 380):
  Identifier (p. 380): aws/s3-publish@v1
  DependsOn (p. 380):
    - build-action
  Compute (p. 381):
```

---

379
### S3Publish

*(Required)*

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: S3Publish_nn.

**Identifier**

*(S3Publish/Identifier)*

*(Required)*

Identifies the action. Do not change this property unless you want to change the version. For more information, see [Working with action versions](p. 284).

Default: aws/s3-publish@v1.

**DependsOn**

*(S3Publish/DependsOn)*

*(Optional)*

Specify an action or action group that must run successfully in order for this action to run.

For more information about the 'depends on' functionality, see [Configuring actions to depend on other actions](p. 280).

**Corresponding UI:** Configuration tab/Action name

**Corresponding UI:** Workflow diagram/S3Publish_nn/aws/s3-publish@v1 label

**Corresponding UI:** none
Compute

(\texttt{S3Publish/Compute})

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see \href{https://docs.aws.amazon.com/CodeCatalyst/latest/userguide/share-compute-across-actions.html}{Sharing compute across actions (p. 298)}.

Corresponding UI: \textit{none}

Type

(\texttt{S3Publish/Compute/Type})

(Required if Compute (p. 381) is included)

The type of compute engine. You can use one of the following values:

- \texttt{EC2} (visual editor) or EC2 (YAML editor) 
  \begin{itemize}
  \item Optimized for flexibility during action runs.
  \item \texttt{Lambda} (visual editor) or Lambda (YAML editor) 
  \item Optimized action start-up speeds.
  \end{itemize}

For more information about compute types, see \href{https://docs.aws.amazon.com/CodeCatalyst/latest/userguide/about-compute-types.html}{About compute types (p. 294)}.

Corresponding UI: Configuration tab/\textit{Compute type}

Fleet

(\texttt{S3Publish/Compute/Fleet})

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see \href{https://docs.aws.amazon.com/CodeCatalyst/latest/userguide/ondemand-fleet-properties.html}{On-demand fleet properties (p. 295)}.

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see \href{https://docs.aws.amazon.com/CodeCatalyst/latest/userguide/provisioned-fleet-properties.html}{Provisioned fleet properties (p. 295)}.

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/\textit{Compute fleet}

Timeout

(\texttt{S3Publish/Timeout})

(Required)
Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

Inputs

(S3Publish/Inputs)
(Optional)

The Inputs section defines the data that the S3Publish needs during a workflow run.

**Note**
A maximum of four inputs (one source and three artifacts) are allowed for each AWS CDK deploy action. Variables do not count towards this total.

If you need to refer to files residing in different inputs (say a source and an artifact), the source input is the primary input, and the artifact is the secondary input. References to files in secondary inputs take a special prefix to distinguish them from the primary. For details, see Example: Referencing files in multiple artifacts (p. 292).

Corresponding UI: Inputs tab

Sources

(S3Publish/Inputs/Sources)
(Required if the files you want to publish to Amazon S3 are stored in a source repository)

If the files that you want to publish to Amazon S3 are stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If the files that you want to publish to Amazon S3 are not contained within a source repository, they must reside in an artifact generated by another action.

For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

Artifacts - input

(S3Publish/Inputs/Artifacts)
(Required if the files you want to publish to Amazon S3 are stored in an output artifact (p. 286) from a previous action)

If the files that you want to publish to Amazon S3 are contained in an artifact generated by a previous action, specify that artifact here. If your files are not contained within an artifact, they must reside in your source repository.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Configuration tab/Artifacts - optional

Variables - input

(S3Publish/Inputs/Variables)
Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Inputs tab/Variables - optional

**Environment**

*(S3Publish/Environment)*

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/connection/role'/Environment

**Name**

*(S3Publish/Environment/Name)*

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/connection/role'/Environment

**Connections**

*(S3Publish/Environment/Connections)*

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/connection/role'/Connection

**Name**

*(S3Publish/Environment/Connections/Name)*

(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/connection/role'/Connection
Role

(S3Publish/Environment/Connections/Role)

(Required)

Specify the name of the IAM role that the Amazon S3 publish action uses to access AWS and to copy files to Amazon S3. Make sure that this role includes:

- The following permissions policy:
  
  **Warning**
  
  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

  ```json
  {
  "Version": "2012-10-17",
  "Statement": [
  {
  "Sid": "VisualEditor0",
  "Effect": "Allow",
  "Action": ["s3:PutObject", "s3:ListBucket", "s3:DeleteObject"],
  "Resource": ["arn:aws:s3:::bucket-name", "arn:aws:s3:::bucket-name/*"]
  }
  ]
  }
  ```

- The following custom trust policy:

  ```json
  {
  "Version": "2012-10-17",
  "Statement": [
  {
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
  "Service": ["codecatalyst-runner.amazonaws.com", "codecatalyst.amazonaws.com"
  ],
  "Action": "sts:AssumeRole"
  }
  ]
  }
  ```

Make sure that this role is associated with your account connection. To learn more about associating an IAM role with an account connection, see Adding IAM roles to account connections (p. 40).

**Note**

You can specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the
CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/Environment/connection/role'/Role

**Configuration**

*S3Publish/Configuration*

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

**SourcePath**

*S3Publish/Configuration/SourcePath*

(Required)

Specify the name and path of a directory or file that you want to publish to Amazon S3. The directory or file can reside in a source repository or an artifact from a previous action, and is relative to the source repository or artifact root.

Examples:

Specifying ./myFolder/ copies the contents of /myFolder to Amazon S3, and preserves the underlying directory structure.

Specifying ./myFolder/myfile.txt copies just myfile.txt to Amazon S3. (The directory structure is removed.)

You cannot use wildcards.

**Note**

You may need to add a prefix to the directory or file path to indicate which artifact or source to find it in. For more information, see Referencing files in a source repository (p. 327) and Referencing files in an artifact (p. 289).

Corresponding UI: Configuration tab/Source path

**DestinationBucketName**

*S3Publish/Configuration/DestinationBucketName*

(Required)

Specify the name of the Amazon S3 bucket where you want to publish files.

Corresponding UI: Configuration tab/Destination bucket - optional

**TargetPath**

*S3Publish/Configuration/TargetPath*

(Optional)

Specify the name and path of the directory in Amazon S3 where you want to publish your files. If the directory does not exist, it will be created. The directory path must not include the bucket name.
Examples:

myS3Folder
dir/myS3Folder/myS3Subfolder

Corresponding UI: Configuration tab/Destination directory - optional

"AWS CDK bootstrap" action reference

The following is the action definition YAML reference for the **AWS CDK bootstrap** action. To learn how to use this action, see [Adding the "AWS CDK bootstrap" action](p. 254).

**Note**
Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```
# The workflow definition starts here.
# See Top-level properties for details.
Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
CDKBootstrapAction nn (p. 386):
  Identifier (p. 387): aws/cdk-bootstrap@v1
  DependsOn (p. 387):
    - action-name
  Compute (p. 387):
    Type (p. 387): EC2 | Lambda
    Fleet (p. 388): fleet-name
    Timeout (p. 388): timeout-minutes
  Inputs (p. 388):
    # Specify a source or an artifact, but not both.
    Sources (p. 388):
      - source-name-1
    Artifacts: (p. 389)
      - artifact-name
  Outputs: (p. 389)
    Artifacts (p. 389):
      - Name (p. 389): cdk_bootstrap_artifacts
      - Files (p. 389):
        - "cdk.out/**/*"
    Environment (p. 390):
      Name (p. 390): environment-name
    Connections (p. 390):
      - Name (p. 390): account-connection-name
    Role (p. 390): iam-role-name
  Configuration: (p. 393)
    Region (p. 393): us-west-2
```

**CDKBootstrapAction**

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: CDKBootstrapAction nn.
Corresponding UI: Configuration tab/Action display name

**Identifier**

*(CDKBootstrapAction/Identifier)*

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Default: aws/cdk-bootstrap@v1.

Corresponding UI: Workflow diagram/CDKBootstrapAction_nn/aws/cdk-bootstrap@v1 label

**DependsOn**

*(CDKBootstrapAction/DependsOn)*

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the ‘depends on’ functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: Inputs tab/Depends on - optional

**Compute**

*(CDKBootstrapAction/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see Sharing compute across actions (p. 298).

Corresponding UI: none

**Type**

*(CDKBootstrapAction/Compute/Type)*

(Required if Compute (p. 387) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).
Fleet

`CDKBootstrapAction/Compute/Fleet`

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

Timeout

`CDKBootstrapAction/Timeout`

(Required)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Inputs

`CDKBootstrapAction/Inputs`

(Optional)

The Inputs section defines the data that the AWS CDK bootstrap action needs during a workflow run.

Corresponding UI: Inputs tab

Note

Only one input (either a source or an artifact) is allowed for each AWS CDK bootstrap action.

Sources

`CDKBootstrapAction/Inputs/Sources`

(Required if your AWS CDK app is stored in a source repository)

If your AWS CDK app is stored in a source repository, specify the label of that source repository. The AWS CDK bootstrap action synthesizes the app in this repository before starting the bootstrapping process. Currently, the only supported repository label is WorkflowSource.

If your AWS CDK app is not contained within a source repository, it must reside in an artifact generated by another action.
For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

Artifacts - input

(CDKBootstrapAction/Inputs/Artifacts)

(Required if your AWS CDK app is stored in an output artifact (p. 286) from a previous action)

If your AWS CDK app is contained in an artifact generated by a previous action, specify that artifact here. The AWS CDK bootstrap action synthesizes the app in the specified artifact into a CloudFormation template before starting the bootstrapping process. If your AWS CDK app is not contained within an artifact, it must reside in your source repository.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Inputs tab/Artifacts - optional

Outputs

(CDKBootstrapAction/Outputs)

(Optional)

Defines the data that is output by the action during a workflow run.

Corresponding UI: Outputs tab

Artifacts - output

(CDKBootstrapAction/Outputs/Artifacts)

(Optional)

Specify the artifacts generated by the action. You can reference these artifacts as input in other actions.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Outputs tab/Artifacts

Name

(CDKBootstrapAction/Outputs/Artifacts/Name)

(Required if Artifacts - output (p. 389) is included)

Specify the name of the artifact that will contain the AWS CloudFormation template that is synthesized by the AWS CDK bootstrap action at runtime. The default value is cdk_bootstrap_artifacts. If you do not specify an artifact, then the action synthesizes the template, but won't save it in an artifact. Consider saving the synthesized template in an artifact to preserve a record of it for testing or troubleshooting purposes.

Corresponding UI: Outputs tab/Artifacts/Add artifact/Build artifact name

Files

(CDKBootstrapAction/Outputs/Artifacts/Files)

(Required if Artifacts - output (p. 389) is included)
Specify the files to include in the artifact. You must specify "cdk.out/**/*" to include your AWS CDK app's synthesized AWS CloudFormation template.

**Note**

- cdk.out is the default directory into which synthesized files are saved. If you specified an output directory other than cdk.out in your cdk.json file, specify that directory here instead of cdk.out.

Corresponding UI: Outputs tab/Artifacts/Add artifact/Files produced by build

### Environment

**CDKBootstrapAction/Environment**

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see [Working with environments](#) and [Creating an environment](#).

Corresponding UI: Configuration tab/'Environment/account/role'/Environment

### Name

**CDKBootstrapAction/Environment/Name**

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/'Environment/account/role'/Environment

### Connections

**CDKBootstrapAction/Environment/Connections**

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see [Managing AWS accounts](#). For information about how to associate an account connection with your environment, see [Creating an environment](#).

Corresponding UI: Configuration tab/'Environment/account/role'/AWS account connection

### Name

**CDKBootstrapAction/Environment/Connections/Name**

(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/'Environment/account/role'/AWS account connection

### Role

**CDKBootstrapAction/Environment/Connections/Role**
Specify the name of the IAM role that the **AWS CDK bootstrap** action uses to access AWS and add the bootstrap stack. Make sure that this role includes the following policies:

**Note**

The permissions shown in the following permissions policy are those required by the `cdk bootstrap` command to perform its bootstrapping. These permissions may change if the AWS CDK changes its bootstrap command.

**Warning**

Only use this role with the **AWS CDK bootstrap** action. It is very permissive, and using it with other actions might pose a security risk.

- The following permissions policy:

  **Warning**

  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "ssm:GetParameterHistory",
                "ecr:PutImageScanningConfiguration",
                "cloudformation:*",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "ssm:GetParameters",
                "iam:PutRolePolicy",
                "ssm:GetParameter",
                "ssm:DeleteParameters",
                "ecr:DeleteRepository",
                "ssm:PutParameter",
                "ssm:DeleteParameter",
                "iam:PassRole",
                "ecr:SetRepositoryPolicy",
                "ssm:GetParametersByPath",
                "ecr:DescribeRepositories",
                "ecr:GetLifecyclePolicy"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:aws-account:parameter/cdk-bootstrap/*",
                "arn:aws:ecr:aws-region:aws-account:repository/cdk-*",
                "arn:aws:iam::aws-account:role/cdk-*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:CreateUploadBucket",
                "cloudformation:ListExports",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:SetTypeDefaultVersion",
                "cloudformation:RegisterPublisher",
                "cloudformation:ActivateType",
                "cloudformation:ListTypes"
            ],
            "Resource": [
                "arn:aws:ecr:aws-region:aws-account:repository/cdk-*",
                "arn:aws:iam::aws-account:role/cdk-*"
            ]
        }
    ]
}
```
The first time the role is used, use the following wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

- The following custom trust policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ],
        "Service": [
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Make sure that this role is added to your account connection. To learn more about adding an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

**Note**

You can specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.
Corresponding UI: Configuration tab/Environment/account/role/Role

Configuration

(CDKBootstrapAction/Configuration)

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

Region

(CDKBootstrapAction/Configuration/Region)

(Required)

Specify the AWS Region into which the bootstrap stack will be deployed. This Region should match the one into which your AWS CDK app is deployed. For a list of Region codes, see Regional endpoints.

Corresponding UI: Configuration tab/Region

"AWS CDK deploy" action reference

The following is the action definition YAML reference for the AWS CDK deploy action. To learn how to use this action, see Adding the "AWS CDK deploy" action (p. 234).

Note

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
CDKDeploy_nn (p. 394):
  Identifier (p. 394): aws/cdk-deploy@v1
  DependsOn (p. 394):
    - CDKBootstrap
  Compute (p. 395):
    Type (p. 395): EC2 | Lambda
    Fleet (p. 395): fleet-name
    Timeout (p. 395): timeout-minutes
  Inputs (p. 396):
    # Specify a source or an artifact, but not both.
    Sources (p. 396):
      - source-name-1
    Artifacts (p. 396):
      - artifact-name
  Outputs (p. 396):
    Artifacts (p. 397):
      - Name (p. 397): cdk_artifact
        Files (p. 397):
          - "cdk.out/**/*"
    Environment (p. 397):
      Name (p. 398): environment-name
```
Connections (p. 398):

- Name (p. 398): account-connection-name
  Role (p. 398): iam-role-name

Configuration: (p. 399)

  StackName (p. 399): my-cdk-stack
  Region (p. 400): us-west-2
  Tags (p. 400): '{"key1": "value1", "key2": "value2"}'
  Context (p. 400): '{"key1": "value1", "key2": "value2"}'
  CdkRootPath (p. 400): directory/cdk.json
  CfnOutputVariables (p. 401): ['"CfnOutputKey1", "CfnOutputKey2", "CfnOutputKey3"]

CDKDeploy

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: CDKDeploy_nn.

Corresponding UI: Configuration tab/Action name

Identifier

(Info)

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Default: aws/cdk-deploy@v1.

Corresponding UI: Workflow diagram/CDKDeploy_nn/aws/cdk-deploy@v1 label

DependsOn

(Info)

(Optional)

Specify an action or action group that must run successfully in order for the AWS CDK deploy action to run. We recommend specifying the AWS CDK bootstrap action in the DependsOn property, like this:

```
CDKDeploy:
  Identifier: aws/cdk-deploy@v1
  DependsOn:
    - CDKBootstrap
```

Note

Bootstrapping is a mandatory prerequisite for deploying an AWS CDK app. If you do not include the AWS CDK Bootstrap action in your workflow, then you must find another way to deploy the AWS CDK bootstrap stack before running your AWS CDK deploy action. For more information, see Prerequisites (p. 235) in Adding the "AWS CDK deploy" action (p. 234).

For more information about the 'depends on' functionality, see Configuring actions to depend on other actions (p. 280).
**Compute**

*(CDKDeploy/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see *Sharing compute across actions (p. 298).*

Corresponding UI: *none*

**Type**

*(CDKDeploy/Compute/Type)*

(Required if *Compute (p. 395)* is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  - Optimized for flexibility during action runs.
- **Lambda** (visual editor) or Lambda (YAML editor)
  - Optimized action start-up speeds.

For more information about compute types, see *About compute types (p. 294).*

Corresponding UI: Configuration tab/Advanced - optional/*Compute type*

**Fleet**

*(CDKDeploy/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: *Linux.x86-64.Large*, *Linux.x86-64.XLarge*. For more information about on-demand fleets, see *On-demand fleet properties (p. 295).*

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see *Provisioned fleet properties (p. 295).*

If *Fleet* is omitted, the default is *Linux.x86-64.Large*.

Corresponding UI: Configuration tab/Advanced - optional/*Compute fleet*

**Timeout**

*(CDKDeploy/Timeout)*
(Required)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in [Quotas for workflows in CodeCatalyst](p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

**Inputs**

*(CDKDeploy/Inputs)*

(Optional)

The Inputs section defines the data that the CDKDeploy needs during a workflow run.

*Note*

Only one input (either a source or an artifact) is allowed for each AWS CDK deploy action.

Corresponding UI: Inputs tab

**Sources**

*(CDKDeploy/Inputs/Sources)*

(Required if the AWS CDK app you want to deploy is stored in a source repository)

If your AWS CDK app is stored in a source repository, specify the label of that source repository. The AWS CDK deploy action synthesizes the app in this repository before starting the deployment process. Currently, the only supported label is WorkflowSource.

If your AWS CDK app is not contained within a source repository, it must reside in an artifact generated by another action.

For more information about sources, see [Working with sources](p. 325).

Corresponding UI: Inputs tab/Sources - optional

**Artifacts - input**

*(CDKDeploy/Inputs/Artifacts)*

(Required if the AWS CDK app you want to deploy is stored in an output artifact (p. 286) from a previous action)

If your AWS CDK app is contained in an artifact generated by a previous action, specify that artifact here. The AWS CDK deploy action synthesizes the app in the specified artifact into a CloudFormation template before starting the deployment process. If your AWS CDK app is not contained within an artifact, it must reside in your source repository.

For more information about artifacts, including examples, see [Working with artifacts](p. 286).

Corresponding UI: Inputs tab/Artifacts - optional

**Outputs**

*(CDKDeploy/Outputs)*
(Optional)
Defines the data that is output by the action during a workflow run.

Corresponding UI: Outputs tab

**Artifacts - output**

*(CDKDeploy/Outputs/Artifacts)*

(Optional)
Specify the artifacts generated by the action. You can reference these artifacts as input in other actions.

For more information about artifacts, including examples, see *Working with artifacts (p. 286).*

Corresponding UI: Outputs tab/Artifacts

**Name**

*(CDKDeploy/Outputs/Artifacts/Name)*

(Required if *Artifacts - output (p. 397)* is included)

Specify the name of the artifact that will contain the AWS CloudFormation template that is synthesized by the *AWS CDK deploy* action at runtime. The default value is cdk_artifact. If you do not specify an artifact, then the action synthesizes the template but won't save it in an artifact. Consider saving the synthesized template in an artifact to preserve a record of it for testing or troubleshooting purposes.

Corresponding UI: Outputs tab/Artifacts/Add artifact/Build artifact name

**Files**

*(CDKDeploy/Outputs/Artifacts/Files)*

(Required if *Artifacts - output (p. 397)* is included)

Specify the files to include in the artifact. You must specify "cdk.out/**/**" to include your AWS CDK app's synthesized AWS CloudFormation template.

Note

`cdk.out` is the default directory into which synthesized files are saved. If you specified an output directory other than `cdk.out` in your `cdk.json` file, specify that directory here instead of `cdk.out`.

Corresponding UI: Outputs tab/Artifacts/Add artifact/files produced by build

**Environment**

*(CDKDeploy/Environment)*

(Required)
Specify the CodeCatalyst environment to use with the action.

For more information about environments, see *Working with environments (p. 307)* and *Creating an environment (p. 308).*

Corresponding UI: Configuration tab/Environment/account/role/Environment
Name

\((CDKDeploy/Environment/Name)\)

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/account/role/Environment

Connections

\((CDKDeploy/Environment/Connections)\)

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection

Name

\((CDKDeploy/Environment/Connections/Name)\)

(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection

Role

\((CDKDeploy/Environment/Connections/Role)\)

(Required)

Specify the name of the IAM role that the AWS CDK deploy action uses to access AWS and deploy the AWS CDK application stack. Make sure that this role includes:

- The following permissions policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackResources"
      ]
    }
  ]
}
```
The following custom trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [  
    {  
      "Sid": "",
      "Effect": "Allow",
      "Principal": {  
        "Service": [  
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Make sure that this role is added to your account connection. To learn more about adding an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

**Note**
You can specify the name of the CodeCatalystWorkflowDevelopmentRole-
`spaceName` role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-
`spaceName` role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-
`spaceName` role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/'Environment/account/role'/Role

**Configuration**

*(CDKDeploy/Configuration)*

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

**StackName**

*(CDKDeploy/Configuration/StackName)*

(Required)
The name of your AWS CDK app stack, as it appears in the entrypoint file in your AWS CDK app’s bin directory. The following example shows the contents of a TypeScript entrypoint file, with the stack name highlighted in red italics. If your entrypoint file is in a different language, it will look similar.

```javascript
import * as cdk from 'aws-cdk-lib';
import { CdkWorkshopTypescriptStack } from '../lib/cdk_workshop_typescript-stack';
const app = new cdk.App();
new CdkWorkshopTypescriptStack(app, 'CdkWorkshopTypescriptStack');
```

You can only specify one stack.

**Tip**
If you have multiple stacks, you can create a parent stack with nested stacks. You can then specify the parent stack in this action to deploy all stacks.

Corresponding UI: Configuration tab/Stack name

**Region**

*(CDKDeploy/Configuration/Region)*

(Required)

Specify the AWS Region into which the AWS CDK application stack will be deployed. For a list of Region codes, see [Regional endpoints](#).

Corresponding UI: Configuration tab/Region

**Tags**

*(CDKDeploy/Configuration/Tags)*

(Optional)

Specify tags that you want to apply to the AWS resources in the AWS CDK application stack. Tags are applied to the stack itself as well as to individual resources in the stack. For more information about tagging, see [Tagging](#) in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

Corresponding UI: Configuration tab/Advanced - optional/Tags

**Context**

*(CDKDeploy/Configuration/Context)*

(Optional)

Specify contexts, in the form of key-value pairs, to associate with the AWS CDK application stack. For more information about contexts, see [Runtime contexts](#) in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

Corresponding UI: Configuration tab/Advanced - optional/Context

**CdkRootPath**

*(CDKDeploy/Configuration/CdkRootPath)*

(Optional)
The path to the directory that contains your AWS CDK project's `cdk.json` file. The **AWS CDK deploy** action runs from this folder, and any outputs created by the action will be added to this directory. If unspecified, the **AWS CDK deploy** action assumes that the `cdk.json` file is in the root of your AWS CDK project.

Corresponding UI: Configuration tab/Advanced - optional/Directory where the cdk.json resides

**CfnOutputVariables**

(**CDKDeploy/Configuration/CfnOutputVariables**)

(Optional)

Specify which CfnOutput constructs in your AWS CDK application code you want to expose as workflow output variables. You can then reference the workflow output variables in subsequent actions in your workflow. For more information about variables in CodeCatalyst, see Working with variables (p. 337).

For example, if your AWS CDK application code looks like this:

```javascript
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';

export class HelloCdkStack extends Stack {
    constructor(scope: Construct, id: string, props?: StackProps) {
        super(scope, id, props);
        const bucket = new s3.Bucket(this, 'my-bucket', {
            removalPolicy: RemovalPolicy.DESTROY,
        });
        new CfnOutput(this, 'bucketName', {
            value: bucket.bucketName,
            description: 'The name of the s3 bucket',
            exportName: 'myBucket',
        });
        const table = new dynamodb.Table(this, 'todos-table', {
            partitionKey: {name: 'todoId', type: dynamodb.AttributeType.NUMBER},
            billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
            removalPolicy: RemovalPolicy.DESTROY,
        });
        new CfnOutput(this, 'tableName', {
            value: table.tableName,
            description: 'The name of the dynamodb table',
            exportName: 'myDynamoDbTable',
        });
        ...
    }
}
```

...and your CfnOutputVariables property looks like this:

```
Configuration:
...
CfnOutputVariables: '['bucketname","tableName"]'
```

...then the action generates the following workflow output variables:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>bucket.bucketName</td>
</tr>
</tbody>
</table>
You can then reference the `tableName` and `bucketName` variables in subsequent actions. To learn how to reference workflow output variables in subsequent actions, see "Referencing a predefined variable (p. 351)."

If you do not specify any `CfnOutput` constructs in the `CfnOutputVariables` property, then the action exposes the first four (or fewer) CloudFormation output variables it finds into workflow output variables. For more information, see "AWS CDK deploy action variables (p. 348)."

**Tip**
To obtain a list of all the CloudFormation output variables the action produces, run the workflow containing the `AWS CDK deploy` action once, and then look in the action's Logs tab. The logs contain a list of all the CloudFormation output variables associated with your AWS CDK app. Once you know what all the CloudFormation variables are, you can specify which ones you want to convert to workflow output variables using the `CfnOutputVariables` property.

For more information about AWS CloudFormation output variables, see the documentation for the `CfnOutput` construct, available at class `CfnOutput (construct)` in the AWS Cloud Development Kit (AWS CDK) API Reference.

Corresponding UI: Configuration tab/Advanced - optional/AWS CloudFormation output variables

"AWS Lambda invoke" action reference

The following is the action definition YAML reference for the `AWS Lambda invoke` action. To learn how to use this action, see Adding the "AWS Lambda invoke" action (p. 256).

**Note**
Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```yaml
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
LambdaInvoke_on (p. 403):
  Identifier (p. 403): aws/lambda-invoke@v1
  DependsOn (p. 403):
    - dependent-action
  Compute (p. 404):
    Type (p. 404): EC2 | Lambda
    Fleet (p. 404): fleet-name
    Timeout (p. 404): timeout-minutes
  Inputs (p. 405):
    # Specify a source or an artifact, but not both.
    Sources (p. 405):
      - source-name-1
    Artifacts: (p. 405)
      - request-payload
    Variables: (p. 406)
      - Name: variable-name-1
        Value: variable-value-1
      - Name: variable-name-2
```
LambdaInvoke

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: Lambda_Invoke_Action_Workflow_nn.

Corresponding UI: Configuration tab/Action name

Identifier

(LambdaInvoke/Identifier)

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Default: aws/lambda-involve@v1.

Corresponding UI: Workflow diagram/LambdaInvoke_nn/aws/lambda-involve@v1 label

DependsOn

(LambdaInvoke/DependsOn)

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the 'depends on' functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: none
**Compute**

*(LambdaInvoke/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see [Sharing compute across actions](p. 298).

Corresponding UI: *none*

**Type**

*(LambdaInvoke/Compute/Type)*

(Required if Compute (p. 404) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see [About compute types (p. 294)].

Corresponding UI: Configuration tab/Compute type

**Fleet**

*(LambdaInvoke/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see [On-demand fleet properties (p. 295)].

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see [Provisioned fleet properties (p. 295)].

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Compute fleet

**Timeout**

*(LambdaInvoke/Timeout)*

(Required)
Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

**Inputs**

*(LambdaInvoke/Inputs)*

(Required)

The Inputs section defines the data that the AWS Lambda invoke action needs during a workflow run.

*Note*

Only one input (either a source or an artifact) is allowed per AWS Lambda invoke action. Variables do not count towards this total.

Corresponding UI: Inputs tab

**Sources**

*(LambdaInvoke/Inputs/Sources)*

(Required if RequestPayloadFile (p. 409) is provided)

If you want to pass a request payload JSON file to the AWS Lambda invoke action, and this payload file is stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If your request payload file is not contained within a source repository, it must reside in an artifact generated by another action.

For more information about the payload file, see RequestPayloadFile (p. 409).

*Note*

Instead of specifying a payload file, you can add the payload's JSON code directly to the action using the RequestPayload property. For more information, see RequestPayload (p. 408).

For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

**Artifacts - input**

*(LambdaInvoke/Inputs/Artifacts)*

(Required if RequestPayloadFile (p. 409) is provided)

If you want to pass a request payload JSON file to the AWS Lambda invoke action, and this payload file is contained in an output artifact (p. 370) from a previous action, specify that artifact here.

For more information about the payload file, see RequestPayloadFile (p. 409).

*Note*

Instead of specifying a payload file, you can add the payload's JSON code directly to the action using the RequestPayload property. For more information, see RequestPayload (p. 408).

For more information about artifacts, including examples, see Working with artifacts (p. 286).
Variables - input

**(LambdaInvoke/Inputs/Variables)**

(Optional)

Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Inputs tab/Variables - optional

Environment

**(LambdaInvoke/Environment)**

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/connection/role/Environment

Name

**(LambdaInvoke/Environment/Name)**

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/connection/role/Environment

Connections

**(LambdaInvoke/Environment/Connections)**

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/connection/role/Connection

Name

**(LambdaInvoke/Environment/Connections/Name)**
(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/connection/role'/Connection

Role

(LambdaInvoke/Environment/Connections/Role)

(Required)

Specify the name of the IAM role that the AWS Lambda invoke action uses to access AWS and invoke your Lambda function. Make sure that this role includes:

- The following permissions policy:

  Warning
  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
      }
    ]
  }
  ```

- The following custom trust policy:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": [
            "codecatalyst-runner.amazonaws.com",
            "codecatalyst.amazonaws.com"
          ]
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  ```

Make sure that this role is associated with your account connection. To learn more about associating an IAM role with an account connection, see Adding IAM roles to account connections (p. 40).

Note

You can specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions.
which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/Environment/connection/role'/Role

**Configuration**

*(LambdaInvoke/Configuration)*

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

**Function**

*(LambdaInvoke/Configuration/Function)*

(Required)

Specify the AWS Lambda function that this action will invoke. You can specify the name of the function, or its Amazon Resource Name (ARN). You can find the name or ARN in the Lambda console.

**Note**

The AWS account where the Lambda function resides can be different from the account specified under Connections:

Corresponding UI: Configuration tab/Function

**AWSRegion**

*(LambdaInvoke/Configuration/AWSRegion)*

(Required)

Specify the AWS Region where your AWS Lambda function resides. For a list of Region codes, see Regional endpoints in the AWS General Reference.

Corresponding UI: Configuration tab/Destination bucket - optional

**RequestPayload**

*(LambdaInvoke/Configuration/RequestPayload)*

(Optional)

If you want to pass a request payload to the AWS Lambda invoke action, specify the request payload here, in JSON format.

Example request payload:

```
{ "key": "value" }
```

If you do not want to pass a request payload to your Lambda function, then omit this property.

**Note**

You can specify either RequestPayload or RequestPayloadFile, but not both.
For more information about the request payload, see the Invoke topic in the AWS Lambda API Reference.

Corresponding UI: Configuration tab/Request payload - optional

**RequestPayloadFile**

*(LambdaInvoke/Configuration/RequestPayloadFile)*

(Optional)

If you want to pass a request payload to the AWS Lambda invoke action, specify the path to this request payload file here. The file must be in JSON format.

The request payload file can reside in a source repository or an artifact from a previous action. The file path is relative to the source repository or artifact root.

If you do not want to pass a request payload to your Lambda function, then omit this property.

**Note**

You can specify either RequestPayload or RequestPayloadFile, but not both.

For more information about the request payload file, see the Invoke topic in the AWS Lambda API Reference.

Corresponding UI: Configuration tab/Request payload file - optional

**ContinueOnError**

*(LambdaInvoke/Configuration/RequestPayloadFile)*

(Optional)

Specify whether you want to mark the AWS Lambda invoke action as succeeded even if the invoked AWS Lambda function fails. Consider setting this property to true to allow subsequent actions in your workflow to start despite the Lambda failure.

The default is to fail the action if the Lambda function fails ("off" in the visual editor or false in the YAML editor).

Corresponding UI: Configuration tab/Continue on error

**LogType**

*(LambdaInvoke/Configuration/LogType)*

(Optional)

Specify whether you want to include error logs in the response from the Lambda function after it is invoked. You can view these logs in the Lambda invoke action's Logs tab in the CodeCatalyst console. Possible values are:

- Tail – return logs
- None – do not return logs

The default is Tail.

For more information about the log type, see the Invoke topic in the AWS Lambda API Reference.
For more information about viewing logs, see Viewing workflow run status and details (p. 315).

Corresponding UI: Configuration tab/Log type

### ResponseFilters

(LambdaInvoke/Configuration/ResponseFilters)

(Optional)

Specify which keys in the Lambda response payload you want to convert to output variables. You can then reference the output variables in subsequent actions in your workflow. For more information about variables in CodeCatalyst, see Working with variables (p. 337).

For example, if your response payload looks like this:

```json
responsePayload = {
    "name": "Saanvi",
    "location": "Seattle",
    "department": {
        "company": "Amazon",
        "team": "AWS"
    }
}
```

...and your response filters look like this:

```json
Configuration:
    ...
    ResponseFilters: '{"name": ".name", "company": ".department.company"} 
```

...then the action generates the following output variables:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Saanvi</td>
</tr>
<tr>
<td>company</td>
<td>Amazon</td>
</tr>
</tbody>
</table>

You can then reference the name and company variables in subsequent actions.

If you do not specify any keys in ResponseFilters, then the action converts each top-level key in the Lambda response into an output variable. For more information, see "AWS Lambda invoke" action variables (p. 349).

Consider using response filters to limit the generated output variables to only those you actually want to use.

Corresponding UI: Configuration tab/Response filters - optional

### Outputs

(LambdaInvoke/Outputs)

(Optional)
Defines the data that is output by the action during a workflow run.

Corresponding UI: **Outputs** tab

**Artifacts**

$(LambdaInvoke/Outputs/Artifacts)$

(Optional)

Specify the artifacts generated by the action. You can reference these artifacts as input in other actions. For more information about artifacts, including examples, see [Working with artifacts (p. 286)](#).

Corresponding UI: Outputs tab/Artifacts/Build artifact name

**Name**

$(LambdaInvoke/Outputs/Artifacts/Name)$

(Optional)

Specify the name of the artifact that will contain the Lambda response payload that is returned by the Lambda function. The default value is lambda_artifacts. If you do not specify an artifact, then the Lambda response payload can be viewed in the action's logs, which are available on the **Logs** tab for the action in the CodeCatalyst console. For more information about viewing logs, see [Viewing workflow run status and details (p. 315)](#).

Corresponding UI: Outputs tab/Artifacts/Build artifact name

**Files**

$(LambdaInvoke/Outputs/Artifacts/Files)$

(Optional)

Specify the files to include in the artifact. You must specify lambda-response.json so that the Lambda response payload file will be included.

Corresponding UI: Outputs tab/Artifacts/Files produced by build

"Deploy AWS CloudFormation stack" action reference

The following is the action definition YAML reference for the **Deploy AWS CloudFormation stack** action. To learn how to use this action, see [Adding the "Deploy AWS CloudFormation stack" action (p. 228)](#).

**Note**

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use **Ctrl+F**. The element will be listed with its associated YAML property.

```
# The workflow definition starts here.
# See [Top-level properties](#) for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
```


DeployCloudFormationStack

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: DeployCloudFormationStack_nn.

Corresponding UI: Configuration tab/Action display name

Identifier

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Default: aws/cfn-deploy@v1.

Corresponding UI: Workflow diagram/DeployCloudFormationStack_nn/aws/cfn-deploy@v1 label
**DependsOn**

*(DeployCloudFormationStack/DependsOn)*

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the 'depends on' functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: Inputs tab/Depends on - optional

**Compute**

*(DeployCloudFormationStack/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see Sharing compute across actions (p. 298).

Corresponding UI: none

**Type**

*(DeployCloudFormationStack/Compute/Type)*

(Required if Compute (p. 413) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).

Corresponding UI: Configuration tab/Advanced - optional/Compute type

**Fleet**

*(DeployCloudFormationStack/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux . x86-64 . Large, Linux . x86-64 . XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).
With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Advanced - optional/Compute fleet

**Timeout**

*(DeployCloudFormationStack/Timeout)*

(Optional)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout in minutes - optional

**Environment**

*(DeployCloudFormationStack/Environment)*

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role'/Environment

**Name**

*(DeployCloudFormationStack/Environment/Name)*

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/account/role'/Environment

**Connections**

*(DeployCloudFormationStack/Environment/Connections)*

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role'/AWS account connection
Name

*(DeployCloudFormationStack/Environment/Connections/Name)*

(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection

Role

*(DeployCloudFormationStack/Environment/Connections/Role)*

(Required)

Specify the name of the IAM role that the **Deploy AWS CloudFormation stack** action uses to access AWS and the AWS CloudFormation service.

Make sure this role includes the following policies:

- The following permissions policy:

  **Warning**
  
  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

  ```json
  {
  "Version": "2012-10-17",
  "Statement": [{
  "Action": [
  "cloudformation:CreateStack",
  "cloudformation:DeleteStack",
  "cloudformation:Describe*",
  "cloudformation:UpdateStack",
  "cloudformation:CreateChangeSet",
  "cloudformation:DeleteChangeSet",
  "cloudformation:ExecuteChangeSet",
  "cloudformation:SetStackPolicy",
  "cloudformation:ValidateTemplate",
  "cloudformation:List*",
  "iam:PassRole"
  ],
  "Resource": "*",
  "Effect": "Allow"
  }]
  }
  ```

  **Note**
  
  The first time the role is used, use the following wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

  "Resource": "*"

- The following custom trust policy:

  ```json
  {
  "Version": "2012-10-17",
  "Statement": [
  {
  ```
"Sid": "",
"Effect": "Allow",
"Principal": {
    "Service": [
        "codecatalyst-runner.amazonaws.com",
        "codecatalyst.amazonaws.com"
    ],
    "Action": "sts:AssumeRole"
}]
}

Make sure that this role is added to the account connection in your space. To learn more about adding an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

Note
You can specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/Environment/account/role'/Role

Inputs

(Optional)

The Inputs section defines the data that the DeployCloudFormationStack needs during a workflow run.

Note
A maximum of four inputs (one source and three artifacts) are allowed per Deploy AWS CloudFormation stack action.

If you need to refer to files residing in different inputs (say a source and an artifact), the source input is the primary input, and the artifact is the secondary input. References to files in secondary inputs take a special prefix to distinguish them from the primary. For details, see Example: Referencing files in multiple artifacts (p. 292).

Corresponding UI: Inputs tab

Sources

(Required if your CloudFormation or AWS SAM template is stored in a source repository)

If your CloudFormation or AWS SAM template is stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If your CloudFormation or AWS SAM template is not contained within a source repository, it must reside in an artifact generated by another action, or in an Amazon S3 bucket.

For more information about sources, see Working with sources (p. 325).
Corresponding UI: Inputs tab/Sources - optional

**Artifacts - input**

(\textcode{DeployCloudFormationStack}/Inputs/Artifacts)

(Required if your CloudFormation or AWS SAM template is stored in an output artifact (p. 286) from a previous action)

If the CloudFormation or AWS SAM template that you want to deploy is contained in an artifact generated by a previous action, specify that artifact here. If your CloudFormation template is not contained within an artifact, it must reside in your source repository or in an Amazon S3 bucket.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Configuration tab/Artifacts - optional

**Configuration**

(\textcode{DeployCloudFormationStack}/Configuration)

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

**name**

(\textcode{DeployCloudFormationStack}/Configuration/name)

(Required)

Specify a name for the CloudFormation stack that the \texttt{Deploy AWS CloudFormation stack} action creates or updates.

Corresponding UI: Configuration tab/Stack name

**region**

(Configuration/region)

(Required)

Specify the AWS Region into which the stack will be deployed. For a list of Region codes, see Regional endpoints.

Corresponding UI: Configuration tab/Stack region

**template**

(\textcode{DeployCloudFormationStack}/Configuration/template)

(Required)

Specify the name and path to your CloudFormation or AWS SAM template file. The template can be in JSON or YAML format, and can reside in a source repository, an artifact from a previous action, or an
Amazon S3 bucket. If the template file is in a source repository or artifact, the path is relative to the source or artifact root. If the template is in an Amazon S3 bucket, the path is the template's Object URL value.

Examples:

./MyFolder/MyTemplate.json
MyFolder/MyTemplate.yml
https://MyBucket.s3.us-west-2.amazonaws.com/MyTemplate.yml

Note
You may need to add a prefix to the template's file path to indicate which artifact or source to find it in. For more information, see Referencing files in a source repository (p. 327) and Referencing files in an artifact (p. 289).

Corresponding UI: Configuration tab/Template

role-arn

(DeployCloudFormationStack/Configuration/role-arn)

(Required)

Specify the Amazon Resource Name (ARN) of the stack role. CloudFormation uses this role to access and modify resources in your stack. For example: arn:aws:iam::123456789012:role/StackRole.

Make sure the stack role includes:

- One or more permissions policies. The policies depend on the resources you have in your stack. For example, if your stack includes an AWS Lambda function, you need to add permissions that grant access to Lambda. If you followed the tutorial described in Tutorial: Deploy a serverless application using AWS CloudFormation (p. 165), it includes a procedure titled, To create a stack role (p. 171) that lists the permissions that the stack role needs if you're deploying a typical serverless application stack.

Warning
Limit the permissions to those required by the CloudFormation service to access resources in your stack. Using a role with broader permissions might pose a security risk.

- The following trust policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Optionally, associate this role with your account connection. To learn more about associating an IAM role with an account connection, see Adding IAM roles to account connections (p. 40). If you do not associate the stack role with the account connection, then the stack role will not appear in the Stack role drop-
down list in the visual editor; however, the role ARN can still be specified in the `role-arn` field using the YAML editor.

**Note**
You can specify the name of the `CodeCatalystWorkflowDevelopmentRole-spaceName` role here, if you'd like. For more information about this role, see [Creating the `CodeCatalystWorkflowDevelopmentRole-spaceName` role for your account and space](p. 517). Understand that the `CodeCatalystWorkflowDevelopmentRole-spaceName` role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/Stack role - optional

capabilities

(*DeployCloudFormationStack*/Configuration/capabilities)

(Required)
Specify a list of IAM capabilities that are required to allow AWS CloudFormation to create certain stacks. In most cases, you can leave capabilities with the default value of `CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND`.

If you see `##[error] requires capabilities: [capability-name]` in your *Deploy AWS CloudFormation stack* action's logs, see [How do I fix IAM capabilities errors?](p. 591) for information about how to fix the problem.

For more information about IAM capabilities, see [Acknowledging IAM resources in AWS CloudFormation templates](in the IAM User Guide).

Corresponding UI: Configuration tab/Advanced/Capabilities

parameter-overrides

(*DeployCloudFormationStack*/Configuration/parameter-overrides)

(Optional)
Specify parameters in your AWS CloudFormation or AWS SAM template that don't have default values, or for which you want to specify non-default values. For more information about parameters, see [Parameters](in the AWS CloudFormation User Guide).

The parameter-overrides property accepts:

- A JSON file containing the parameters and values.
- A comma-separate list of parameters and values.

**To specify a JSON file**

1. Make sure the JSON file uses one of the following syntaxes:

   ```json
   {
     "Parameters": {
       "Param1": "Value1",
       "Param2": "Value2",
       ...
   ```
Or...

```json
[
  {
    "ParameterKey": "Param1",
    "ParameterValue": "Value1"
  },
  ...
]
```

(There are other syntaxes, but they are not supported by CodeCatalyst at the time of writing.) For more information about specifying CloudFormation parameters in a JSON file, see [Supported JSON syntax](https://docs.aws.amazon.com/cli/latest/reference/cfn-stack.html) in the AWS CLI Command Reference.

2. Specify the path to the JSON file using one of the following formats:

- If your JSON file resides in an output artifact from a previous action, use:

  ```
  file:///artifacts/current-action-name/output-artifact-name/path-to-json-file
  ```
  
  See [Example 1](#example-1) for details.

- If your JSON file resides in your source repository, use:

  ```
  file:///sources/WorkflowSource/path-to-json-file
  ```
  
  See [Example 2](#example-2) for details.

**Example 1** – The JSON file resides in an output artifact

```yaml
### My workflow YAML
...
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ParamArtifact
          Files:
            - params.json
      Configuration:
  ...
MyDeployCFNStackAction:
  Identifier: aws/cfn-deploy@v1
  Configuration:
    parameter-overrides: file:///artifacts/MyDeployCFNStackAction/ParamArtifact/params.json
```

**Example 2** – The JSON file resides in your source repository, in a folder called `my/folder`

```yaml
### My workflow YAML
...
Actions:
  MyDeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
```
To use a comma-separate list of parameters

- Add parameter name-value pairs in the `parameter-overrides` property using the following format:

  \texttt{param-1=value-1,param-2=value-2}

For example, assuming the following AWS CloudFormation template:

```yaml
## My CloudFormation template

Description: My AWS CloudFormation template

Parameters:
  InstanceType:
    Description: Defines the Amazon EC2 compute for the production server.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t3.medium

Resources:
  ...
```

...you might set the `parameter-overrides` property as follows:

```yaml
## My workflow YAML

... Actions:
  ...
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: InstanceType=t3.medium,UseVPC=true
```

\textbf{Note}

You can specify a parameter name without a corresponding value using `undefined` as the value. For example:

```yaml
parameter-overrides: MyParameter=undefined
```

The effect is that during a stack update, CloudFormation uses the existing parameter value for the given parameter name.

**Corresponding UI:**

- Configuration tab/Advanced/\textbf{Parameter overrides}
- Configuration tab/Advanced/Parameter overrides/\textbf{Specify overrides using a file}
- Configuration tab/Advanced/Parameter overrides/\textbf{Specify overrides using a value set}

\textbf{no-execute-changeset}

\textit{(DeployCloudFormationStack/Configuration/no-execute-changeset)}
Specify whether you want CodeCatalyst to create the CloudFormation change set and then stop before running it. This gives you the opportunity to review the change set in the CloudFormation console. If you determine that the change set looks good, disable this option and then re-run the workflow so that CodeCatalyst can create and run the change set without stopping. The default is to create and run the change set without stopping. For more information, see the AWS CloudFormation `deploy` parameter in the AWS CLI Command Reference. For more information about viewing change sets, see Viewing a change set in the AWS CloudFormation User Guide.

Corresponding UI: Configuration tab/Advanced/No execute change set

fail-on-empty-changeset

(DeployCloudFormationStack/Configuration/fail-on-empty-changeset)

(Optional)

Specify whether you want CodeCatalyst to fail the Deploy AWS CloudFormation stack action if the CloudFormation change set is empty. (If a change set is empty, it means there were no changes made to the stack during the latest deployment.) The default is to allow the action to proceed if the change set is empty, and to return an UPDATE_COMPLETE message even though the stack was not updated.

For more information about this setting, see the AWS CloudFormation `deploy` parameter in the AWS CLI Command Reference. For more information about change sets, see Updating stacks using change sets in the AWS CloudFormation User Guide.

Corresponding UI: Configuration tab/Advanced/Fail on empty changeset

disable-rollback

(DeployCloudFormationStack/Configuration/disable-rollback)

(Optional)

Specify whether you want CodeCatalyst to roll back the stack deployment if it fails. The rollback returns the stack to the last known stable state. The default is to enable rollbacks. For more information about this setting, see the AWS CloudFormation `deploy` parameter in the AWS CLI Command Reference.

For more information about how the Deploy AWS CloudFormation stack action handles rollbacks, see Configuring rollbacks (p. 240).

For more information about rolling back a stack, see Stack failure options in the AWS CloudFormation User Guide.

Corresponding UI: Configuration tab/Advanced/Disable rollback

termination-protection

(DeployCloudFormationStack/Configuration/termination-protection)

(Optional)

Specify whether you want the Deploy AWS CloudFormation stack to add termination protection to the stack that it is deploying. If a user attempts to delete a stack with termination protection enabled, the deletion fails and the stack, including its status, remains unchanged. The default is to disable termination
protection. For more information, see Protecting a stack from being deleted in the AWS CloudFormation User Guide.

Corresponding UI: Configuration tab/Advanced/Termination protection

**timeout-in-minutes**

*(DeployCloudFormationStack/Configuration/timeout-in-minutes)*

(Optional)

Specify the amount of time, in minutes, that CloudFormation should allot before timing out stack creation operations and setting the stack status to CREATE_FAILED. If CloudFormation can't create the entire stack in the time allotted, it fails the stack creation due to timeout and rolls back the stack.

By default, there is no timeout for stack creation. However, individual resources may have their own timeouts based on the nature of the service they implement. For example, if an individual resource in your stack times out, stack creation also times out even if the timeout you specified for stack creation hasn't yet been reached.

Corresponding UI: Configuration tab/Advanced/CloudFormation timeout

**notification-arns**

*(DeployCloudFormationStack/Configuration/notification-arns)*

(Optional)

Specify the ARN of an Amazon SNS topic that you want CodeCatalyst to send notification messages to. For example, arn:aws:sns:us-east-1:111222333:MyTopic. When the Deploy AWS CloudFormation stack action runs, CodeCatalyst coordinates with CloudFormation to send one notification per AWS CloudFormation event that occurs during the stack creation or update process. (The events are visible in the AWS CloudFormation console's Events tab for the stack.) You can specify up to five topics. For more information, see What is Amazon SNS?

Corresponding UI: Configuration tab/Advanced/Notification ARNs

**monitor-alarm-arns**

*(DeployCloudFormationStack/Configuration/monitor-alarm-arns)*

(Optional)

Specify the Amazon Resource Name (ARN) of an Amazon CloudWatch alarm to use as a rollback trigger. For example, arn:aws:cloudwatch::123456789012:alarm/MyAlarm. You can have a maximum of five rollback triggers.

**Note**

If you specify a CloudWatch alarm ARN, you'll also need to configure additional permissions to enable the action to access CloudWatch. For more information, see Configuring rollbacks (p. 240).

Corresponding UI: Configuration tab/Advanced/Monitor alarm ARNs

**monitor-timeout-in-minutes**

*(DeployCloudFormationStack/Configuration/monitor-timeout-in-minutes)*
(Optional)

Specify an amount of time, from 0 to 180 minutes, during which CloudFormation monitors the specified alarms. Monitoring begins after all the stack resources have been deployed. If the alarm occurs within the specified monitoring time, then the deployment fails, and CloudFormation rolls back the entire stack operation.

Default: 0. CloudFormation only monitors alarms while the stack resources are being deployed, not after.

Corresponding UI: Configuration tab/Advanced/Monitoring time

tags

(DeployCloudFormationStack/Configuration/tags)

(Optional)

Specify tags to attach to your CloudFormation stack. Tags are arbitrary key-value pairs that you can use to identify your stack for purposes such as cost allocation. For more information about what tags are and how they can be used, see Tagging your resources in the Amazon EC2 User Guide for Linux Instances. For more information about tagging in CloudFormation, see Setting AWS CloudFormation stack options in the AWS CloudFormation User Guide.

A key can have alphanumeric characters or spaces, and can have up to 127 characters. A value can have alphanumeric characters or spaces, and can have up to 255 characters.

You can add up to 50 unique tags for each stack.

Corresponding UI: Configuration tab/Advanced/Tags

"Deploy to Amazon ECS" action reference

The following is the action definition YAML reference for the Deploy to Amazon ECS action. To learn how to use this action, see Adding the "Deploy to Amazon ECS" action (p. 230).

Note

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
ECSDeployAction_00 (p. 425):
  Identifier (p. 425): aws/ecs-deploy@v1
  DependsOn (p. 425):
    - build-action
  Compute (p. 426):
    Type (p. 426): EC2 | Lambda
    Fleet (p. 426): fleet-name
  Timeout (p. 426): timeout-minutes
  Environment (p. 427):
    Name (p. 427): environment-name
  Connections (p. 427):
    - Name (p. 427): account-connection-name
    Role (p. 427): DeployToECS
### Inputs (p. 429):
# Specify a source or an artifact, but not both.

**Sources (p. 430):**
- `source-name-1`

**Artifacts (p. 430):**
- `task-definition-artifact`

**Configuration (p. 430):**
- `region (p. 430): us-east-1`
- `cluster (p. 430): ecs-cluster`
- `service (p. 431): ecs-service`
- `task-definition (p. 431): task-definition-path`
- `force-new-deployment (p. 431): false|true`
- `codedeploy-appspec (p. 431): app-spec-file-path`
- `codedeploy-application (p. 432): application-name`
- `codedeploy-deployment-group (p. 432): deployment-group-name`
- `codedeploy-deployment-description (p. 432): deployment-description`

### ECSDeployAction

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: ECSDeployAction_nn.

Corresponding UI: Configuration tab/Action display name

### Identifier

**(ECSDeployAction/Identifier)**

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see Working with action versions (p. 284).

Default: aws/ecs-deploy@v1.

Corresponding UI: Workflow diagram/ECSDeployAction_nn/aws/ecs-deploy@v1 label

### DependsOn

**(ECSDeployAction/DependsOn)**

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the 'depends on' functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: Inputs tab/Depends on - optional

### Compute

**(ECSDeployAction/Compute)**
(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see Sharing compute across actions (p. 298).

Corresponding UI: none

**Type**

**(ECSDeployAction)/Compute/Type**

(Required if Compute (p. 425) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.
- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).

Corresponding UI: Configuration tab/Advanced - optional/Compute type

**Fleet**

**(ECSDeployAction)/Compute/Fleet**

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Advanced - optional/Compute fleet

**Timeout**

**(ECSDeployAction)/Timeout**

(Optional)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is
described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

**Environment**

*(ECSDeployAction/Environment)*

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role/Environment

**Name**

*(ECSDeployAction/Environment/Name)*

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/account/role/Environment

**Connections**

*(ECSDeployAction/Environment/Connections)*

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection

**Name**

*(ECSDeployAction/Environment/Connections/Name)*

(Required)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection

**Role**

*(ECSDeployAction/Environment/Connections/Role)*
Specify the name of the IAM role that the Deploy to Amazon ECS action uses to access AWS. Make sure that this role includes the following policies:

- The following permissions policy:

  **Warning**
  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [{
      "Action": [
        "ecs:DescribeServices",
        "ecs:CreateTaskSet",
        "ecs:DeleteTaskSet",
        "ecs:ListClusters",
        "ecs:RegisterTaskDefinition",
        "ecs:UpdateServicePrimaryTaskSet",
        "ecs:UpdateService",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeListeners",
        "elasticloadbalancing:ModifyListener",
        "elasticloadbalancing:DescribeRules",
        "elasticloadbalancing:ModifyRule",
        "lambda:InvokeFunction",
        "lambda:ListFunctions",
        "cloudwatch:DescribeAlarms",
        "sns:Publish",
        "sns:ListTopics",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "codedeploy:CreateApplication",
        "codedeploy:CreateDeployment",
        "codedeploy:CreateDeploymentGroup",
        "codedeploy:GetApplication",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentGroup",
        "codedeploy:GetDeploymentGroups",
        "codedeploy:GetDeployments",
        "codedeploy:StopDeployment",
        "codedeploy:GetDeploymentTarget",
        "codedeploy:GetDeploymentTargets",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision",
        "codedeploy:BatchGetApplicationRevisions",
        "codedeploy:BatchGetDeploymentGroups",
        "codedeploy:BatchGetDeployments",
        "codedeploy:BatchGetApplications",
        "codedeploy:ListApplicationRevisions",
        "codedeploy:ListDeploymentConfigs",
        "codedeploy:ContinueDeployment"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }, {
      "Action": ["iam:PassRole"]
    }]
  }
```
Note
The first time the role is used, use the following wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

• The following custom trust policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "codecatalyst-runner.amazonaws.com",
               "codecatalyst.amazonaws.com"
            ],
            "Action": "sts:AssumeRole"
         }
      }
   ]
}
```

Make sure that this role is added to your account connection. To learn more about adding an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

Note
You can specify the name of the CodeCatalystWorkflowDevelopmentRole-spaceName role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-spaceName role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

Corresponding UI: Configuration tab/Environment/account/role/Role

**Inputs**

*(ECSDeployAction/Inputs)*

(Optional)

The Inputs section defines the data that the ECSDeployAction needs during a workflow run.

Note
Only one input (either a source or an artifact) is allowed per Deploy to Amazon ECS action.

Corresponding UI: Inputs tab
Sources

(ECSDeployAction/Inputs/Sources)

(Required if your task definition file is stored in a source repository)

If your task definition file is stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If your task definition file is not contained within a source repository, it must reside in an artifact generated by another action.

For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

Artifacts - input

(ECSDeployAction/Inputs/Artifacts)

(Required if your task definition file is stored in an output artifact (p. 286) from a previous action)

If the task definition file that you want to deploy is contained in an artifact generated by a previous action, specify that artifact here. If your task definition file is not contained within an artifact, it must reside in your source repository.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Configuration tab/Artifacts - optional

Configuration

(ECSDeployAction/Configuration)

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

region

(Configuration/region)

(Required)

Specify the AWS Region where your Amazon ECS cluster and service reside. For a list of Region codes, see Regional endpoints in the AWS General Reference.

Corresponding UI: Configuration tab/Region

cluster

(ECSDeployAction/Configuration/cluster)

(Required)
Specify the name of an existing Amazon ECS cluster. The **Deploy to Amazon ECS** action will deploy your containerized application as a task into this cluster. For more information about Amazon ECS clusters, see [Clusters](https://docs.aws.amazon.com/AmazonECS/latest/userguide/clusters.html) in the *Amazon Elastic Container Service Developer Guide*.

Corresponding UI: Configuration tab/Cluster

**service**

(*ECSDeployAction*/Configuration/service)

(Required)

Specify the name of an existing Amazon ECS service that will instantiate the task definition file. This service must reside under the cluster specified in the `cluster` field. For more information about Amazon ECS services, see [Amazon ECS services](https://docs.aws.amazon.com/AmazonECS/latest/userguide/services.html) in the *Amazon Elastic Container Service Developer Guide*.

Corresponding UI: Configuration tab/Service

**task-definition**

(*ECSDeployAction*/Configuration/task-definition)

(Required)

Specify the path to an existing task definition file. If the file resides in your source repository, the path is relative to the source repository root folder. If your file resides in an artifact from a previous workflow action, the path is relative to the artifact root folder. For more information about task definition files, see [Task definitions](https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-definition.html) in the *Amazon Elastic Container Service Developer Guide*.

Corresponding UI: Configuration tab/Task definition

**force-new-deployment**

(*ECSDeployAction*/Configuration/force-new-deployment)

(Required)

If enabled, the Amazon ECS service is able to start new deployments without service definition changes. Forcing a deployment causes the service to stop all currently running tasks and launch new tasks. For more information about forcing new deployments, see [Updating a service](https://docs.aws.amazon.com/AmazonECS/latest/userguide/services.html) in the *Amazon Elastic Container Service Developer Guide*.

Default: false

Corresponding UI: Configuration tab/Force a new deployment of the service

**codedeploy-appspec**

(*ECSDeployAction*/Configuration/codedeploy-appspec)

(Required if you have configured your Amazon ECS service to use blue/green deployments, otherwise, omit)

Specify the name and path to an existing CodeDeploy application specification (AppSpec) file. This file must reside in the root of your CodeCatalyst source repository. For more information about AppSpec files, see [CodeDeploy application specification (AppSpec) files](https://docs.aws.amazon.com/CodeDeploy/latest/userguide/codeploy-apispec.html) in the *AWS CodeDeploy User Guide*.

**Note**

Only supply CodeDeploy information if you have configured your Amazon ECS service to perform blue/green deployments. For rolling update deployments (the default), omit
CodeDeploy information. For more information about Amazon ECS deployments, see Amazon ECS deployment types in the Amazon Elastic Container Service Developer Guide.

**Note**
The CodeDeploy fields may be hidden in the visual editor. To get them to appear, see Why are CodeDeploy fields missing from the visual editor? (p. 591).

Corresponding UI: Configuration tab/CodeDeploy AppSpec
codedeploy-application

*(ECSDeployAction/Configuration/codedeploy-application)*

(Required if codedeploy-appspec is included)

Specify the name of an existing CodeDeploy application. For more information about CodeDeploy applications, see Working with applications in CodeDeploy in the AWS CodeDeploy User Guide.

Corresponding UI: Configuration tab/CodeDeploy application
codedeploy-deployment-group

*(ECSDeployAction/Configuration/codedeploy-deployment-group)*

(Required if codedeploy-appspec is included)

Specify the name of an existing CodeDeploy deployment group. For more information about CodeDeploy deployment groups, see Working with deployment groups in CodeDeploy in the AWS CodeDeploy User Guide.

Corresponding UI: Configuration tab/CodeDeploy deployment group
codedeploy-deployment-description

*(ECSDeployAction/Configuration/codedeploy-deployment-description)*

(Optional)

Specify a description of the deployment that this action will create. For more information, see Working with deployments in CodeDeploy in the AWS CodeDeploy User Guide.

Corresponding UI: Configuration tab/CodeDeploy deployment description

"Deploy to Kubernetes cluster" action reference

The following is the action definition YAML reference for the Deploy to Kubernetes cluster action. To learn how to use this action, see Adding the "Deploy to Kubernetes cluster" action (p. 231).

**Note**
Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```yaml
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:
```
# The action definition starts here.

**DeployToKubernetesCluster**

**Identifier** (DeployToKubernetesCluster/Identifier)

*(Required)*

Identifies the action. Do not change this property unless you want to change the version. For more information, see [Working with action versions](#). Default: `aws/kubernetes-deploy@v1`.

**Corresponding UI:** Workflow diagram/DeployToKubernetesCluster_nn/aws/kubernetes-deploy@v1 label

**DependsOn**

*(DeployToKubernetesCluster/DependsOn)*

*(Optional)*

Specify an action or action group that must run successfully in order for this action to run.

**Compute**

*(p. 434)*

- **Type** *(EC2 | Lambda)*
- **Fleet** *(fleet-name)*

**Timeout** *(timeout-minutes)*

**Environment**

*(p. 435)*

- **Name** *(environment-name)*
- **Connections**
  - **Name** *(account-connection-name)*
  - **Role** *(DeployToEKS)*

**Inputs** *(p. 437)*

# Specify a source or an artifact, but not both.

**Sources** *(source-name-1)*

**Artifacts** *(manifest-artifact)*

**Configuration** *(p. 438)*

- **Namespace** *(namespace)*
- **Region** *(us-east-1)*
- **Cluster** *(eks-cluster)*
- **Manifests** *(manifest-path)*

## DeployToKubernetesCluster

*(Required)*

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: `DeployToKubernetesCluster_nn`.

**Corresponding UI:** Configuration tab/Action display name
For more information about the 'depends on' functionality, see Configuring actions to depend on other actions (p. 280).

Corresponding UI: Inputs tab/Depends on - optional

**Compute**

*(DeployToKubernetesCluster/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see Sharing compute across actions (p. 298).

Corresponding UI: none

**Type**

*(DeployToKubernetesCluster/Compute/Type)*

(Required if Compute (p. 434) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.
- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see About compute types (p. 294).

Corresponding UI: Configuration tab/Advanced - optional/Compute type

**Fleet**

*(DeployToKubernetesCluster/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Advanced - optional/Compute fleet
Timeout

(\texttt{DeployToKubernetesCluster}/\texttt{Timeout})

(Optional)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in \textit{Quotas for workflows in CodeCatalyst (p. 457)}. The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/\texttt{Timeout - optional}

Environment

(\texttt{DeployToKubernetesCluster}/\texttt{Environment})

(Required)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see \textit{Working with environments (p. 307)} and \textit{Creating an environment (p. 308)}.

Corresponding UI: Configuration tab/\texttt{Environment/account/role}/\texttt{Environment}

Name

(\texttt{DeployToKubernetesCluster}/\texttt{Environment}/\texttt{Name})

(Required)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/\texttt{Environment/account/role}/\texttt{Environment}

Connections

(\texttt{DeployToKubernetesCluster}/\texttt{Environment}/\texttt{Connections})

(Required)

Specify the account connection to associate with the action.

For more information about account connections, see \textit{Managing AWS accounts (p. 36)}. For information about how to associate an account connection with your environment, see \textit{Creating an environment (p. 308)}.

Corresponding UI: Configuration tab/\texttt{Environment/account/role}/\texttt{AWS account connection}

Name

(\texttt{DeployToKubernetesCluster}/\texttt{Environment/Connections/Name})

(Required)

Specify the name of the account connection.
Role

(\texttt{DeployToKubernetesCluster/Environment/Connections/Role})

(Required)

Specify the name of the IAM role that the \textit{Deploy to Kubernetes cluster} action uses to access AWS. Make sure that this role includes the following policies:

- The following permissions policy:

  \textbf{Warning}
  
  Limit the permissions to those shown in the following policy. Using a role with broader permissions might pose a security risk.

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "eks:DescribeCluster",
          "eks:ListClusters"
        ],
        "Resource": "*"
      }
    ]
  }
  ```

  \textbf{Note}
  
  The first time the role is used, use the following wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

- The following custom trust policy:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": [
            "codecatalyst-runner.amazonaws.com",
            "codecatalyst.amazonaws.com"
          ]
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  ```

Make sure that this role is added to:

- Your account connection. To learn more about adding an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).
• Your Kubernetes ConfigMap. To learn more about adding an IAM role to a ConfigMap, see Manage IAM users and roles in the eksctl documentation.

Tip
See also Tutorial: Deploy an application to Amazon EKS (p. 207) for instructions on adding an IAM role to an account connection and ConfigMap.

Note
You can specify the name of the CodeCatalystWorkflowDevelopmentRole-<spaceName> role here, if you'd like. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-<spaceName> role for your account and space (p. 517). Understand that the CodeCatalystWorkflow DevelopmentRole-<spaceName> role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern. If you decide to use the CodeCatalystWorkflowDevelopmentRole-<spaceName> role, make sure to add it to your ConfigMap file, following the instructions in Manage IAM users and roles in the eksctl documentation.

Corresponding UI: Configuration tab/'Environment/account/role'/Role

Inputs
(DeployToKubernetesCluster/Inputs)

(Optional)

The Inputs section defines the data that the DeployToKubernetesCluster needs during a workflow run.

Note
Only one input (either a source or an artifact) is allowed per Deploy to Amazon EKS action.

Corresponding UI: Inputs tab

Sources
(DeployToKubernetesCluster/Inputs/Sources)

(Required if your manifest file is stored in a source repository)

If your Kubernetes manifest file or files are stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If your manifest files are not contained within a source repository, they must reside in an artifact generated by another action.

For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

Artifacts - input
(DeployToKubernetesCluster/Inputs/Artifacts)

(Required if your manifest file is stored in an output artifact (p. 286) from a previous action)

If the Kubernetes manifest file or files are contained in an artifact generated by a previous action, specify that artifact here. If your manifest files are not contained within an artifact, they must reside in your source repository.
For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Configuration tab/Artifacts - optional

Configuration

(DeployToKubernetesCluster/Configuration)

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

Namespace

(Configuration/Namespace)

(Optional)

Specify the Kubernetes namespace into which your Kubernetes application will be deployed. Use default if you are not using namespaces with your cluster. For more information on namespaces, see Subdividing your cluster using Kubernetes namespaces in the Kubernetes documentation.

If you omit the namespace, a value of default is used.

Corresponding UI: Configuration tab/Namespaces

Region

(Configuration/Region)

(Required)

Specify the AWS Region where your Amazon EKS cluster and service reside. For a list of Region codes, see Regional endpoints in the AWS General Reference.

Corresponding UI: Configuration tab/Region

Cluster

(DeployToKubernetesCluster/Configuration/Cluster)

(Required)

Specify the name of an existing Amazon EKS cluster. The Deploy to Kubernetes cluster action will deploy your containerized application into this cluster. For more information about Amazon EKS clusters, see Clusters in the Amazon EKS User Guide.

Corresponding UI: Configuration tab/Cluster

Manifests

(DeployToKubernetesCluster/Configuration/Manifests)

(Required)

Specify the path to your YAML-formatted Kubernetes manifest file(s), which are called configuration files, config files, or simply, configurations in the Kubernetes documentation.
If you're using multiple manifest files, place them in a single folder and reference that folder. Manifest files are processed alphanumerically by Kubernetes, so make sure to prefix file names with increasing numbers or letters to control the processing order. For example:

00-namespace.yaml
01-deployment.yaml

If your manifest files reside in your source repository, the path is relative to the source repository root folder. If the files reside in an artifact from a previous workflow action, the path is relative to the artifact root folder.

Examples:

Manifets/
deployment.yaml
my-deployment.yml

Do not use wildcards (*).

Note

Helm charts and kustomization files are not supported.

For more information about manifest files, see Organizing resource configurations in the Kubernetes documentation.

Corresponding UI: Configuration tab/Manifests

"GitHub Actions" action reference

The following is the action definition YAML reference for the GitHub Actions action.

Choose a YAML property in the following code to see a description if it.

Note

Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```yaml
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
action-name (p. 440):
  Identifier (p. 441): aws/github-actions-runner@v1
  DependsOn (p. 441):
    - dependent-action-name-1
  Compute (p. 441):
    Fleet (p. 441): fleet-name
  Timeout (p. 442): timeout-minutes
  Environment (p. 442):
    Name (p. 442): environment-name
    Connections (p. 442):
      - Name (p. 442): account-connection-name
      Role (p. 443): iam-role-name
  Inputs (p. 443):
    Sources (p. 443):
```

439
- source-name-1
- source-name-2

Artifacts: (p. 444)
- artifact-name

Variables (p. 444):
- Name: variable-name-1
  Value: variable-value-1
- Name: variable-name-2
  Value: variable-value-2

Outputs (p. 444):
Artifacts (p. 444):
- Name (p. 445): output-artifact-1
  Files (p. 445):
    - github-output/artifact-1.jar
    - "github-output/build*"
  - Name (p. 445): output-artifact-2
    Files (p. 445):
    - github-output/artifact-2.1.jar
    - github-output/artifact-2.2.jar

Variables (p. 445):
- variable-name-1
- variable-name-2

AutoDiscoverReports (p. 446):
Enabled (p. 446): true | false
ReportNamePrefix (p. 446): AutoDiscovered
IncludePaths (p. 447):
- "**/*"
ExcludePaths (p. 447):
- node_modules/cdk/junit.xml

SuccessCriteria (p. 448):
  PassRate (p. 448): percent
  LineCoverage (p. 448): percent
  BranchCoverage (p. 449): percent
  Vulnerabilities (p. 449):
    Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
    Number: whole-number

Reports (p. 450):
report-name-1 (p. 450):
  Format (p. 450): format
IncludePaths (p. 447):
- "*.xml"
ExcludePaths (p. 447):
  - report2.xml
  - report3.xml
SuccessCriteria (p. 448):
  PassRate (p. 448): percent
  LineCoverage (p. 448): percent
  BranchCoverage (p. 449): percent
  Vulnerabilities (p. 449):
    Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
    Number: whole-number

Configuration (p. 451)
Steps (p. 451):
- github-actions-code

**action-name**

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Corresponding UI: Configuration tab/`action-name`
Identifier

*(action-name/Identifier)*

Identifies the action. Do not change this property unless you want to change the version. For more information, see [Working with action versions (p. 284)](https://aws.github.io/aws-codebuild-documentation/latest/user-guide/working-with-action-versions.html).

Use `aws/github-actions-runner@v1` for GitHub Actions actions.

Corresponding UI: Workflow diagram/*action-name/aws/github-actions-runner@v1* label

DependsOn

*(action-name/DependsOn)*

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the 'depends on' functionality, see [Configuring actions to depend on other actions (p. 280)](https://aws.github.io/aws-codebuild-documentation/latest/user-guide/configuring-actions-to-depend-on-other-actions.html).

Corresponding UI: Inputs tab/Depends on - optional

Compute

*(action-name/Compute)*

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see [Sharing compute across actions (p. 298)](https://aws.github.io/aws-codebuild-documentation/latest/user-guide/sharing-compute-across-actions.html).

Corresponding UI: none

Fleet

*(action-name/Compute/Fleet)*

(Optional)

Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: `Linux.x86-64.Large`, `Linux.x86-64.XLarge`. For more information about on-demand fleets, see [On-demand fleet properties (p. 295)](https://aws.github.io/aws-codebuild-documentation/latest/user-guide/on-demand-fleet-properties.html).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see [Provisioned fleet properties (p. 295)](https://aws.github.io/aws-codebuild-documentation/latest/user-guide/provisioned-fleet-properties.html).

If Fleet is omitted, the default is `Linux.x86-64.Large`.

Corresponding UI: Configuration tab/Compute fleet - optional
**Timeout**

(*action-name*/Timeout)

(Optional)

Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

**Environment**

(*action-name*/Environment)

(Optional)

Specify the CodeCatalyst environment to use with the action.

For more information about environments, see Working with environments (p. 307) and Creating an environment (p. 308).

Corresponding UI: Configuration tab/Environment/account/role

**Name**

(*action-name*/Environment/Name)

(Optional)

Specify the name of an existing environment that you want to associate with the action.

Corresponding UI: Configuration tab/Environment/account/role/Environment

**Connections**

(*action-name*/Environment/Connections)

(Optional)

Specify the account connection to associate with the action.

For more information about account connections, see Managing AWS accounts (p. 36). For information about how to associate an account connection with your environment, see Creating an environment (p. 308).

Corresponding UI: none

**Name**

(*action-name*/Environment/Connections/Name)

(Optional)

Specify the name of the account connection.

Corresponding UI: Configuration tab/Environment/account/role/AWS account connection
Role

(*action-name*/Environment/Connections/Role)

(Optional)

Specify the name of the IAM role that this action uses in order to access and operate in AWS services such as Amazon S3 and Amazon ECR. Make sure this role is added to your account connection. To add an IAM role to an account connection, see Adding IAM roles to account connections (p. 40).

**Note**

You may be able to specify the name of the CodeCatalystWorkflowDevelopmentRole-*spaceName* role here, provided it has sufficient permissions. For more information about this role, see Creating the CodeCatalystWorkflowDevelopmentRole-*spaceName* role for your account and space (p. 517). Understand that the CodeCatalystWorkflowDevelopmentRole-*spaceName* role has very broad permissions which may pose a security risk. We recommend that you only use this role in tutorials and scenarios where security is less of a concern.

**Warning**

Limit the permissions to those required by the GitHub Action action. Using a role with broader permissions might pose a security risk.

Corresponding UI: Configuration tab/Environment/account/role/Role

Inputs

(*action-name*/Inputs)

(Optional)

The Inputs section defines the data that an action needs during a workflow run.

**Note**

A maximum of four inputs (one source and three artifacts) are allowed per GitHub Actions action. Variables do not count towards this total.

If you need to refer to files residing in different inputs (say a source and an artifact), the source input is the primary input, and the artifact is the secondary input. References to files in secondary inputs take a special prefix to distinguish them from the primary. For details, see Example: Referencing files in multiple artifacts (p. 292).

Corresponding UI: Inputs tab

Sources

(*action-name*/Inputs/Sources)

(Optional)

Specify the labels that represent the source repositories that will be needed by the action. Currently, the only supported label is WorkflowSource, which represents the source repository where your workflow definition file is stored.

If you omit a source, then you must specify at least one input artifact under *action-name*/Inputs/Artifacts.

For more information about sources, see Working with sources (p. 325).
Artifacts - input

*(action-name*/Inputs/Artifacts)*

(Optional)

Specify artifacts from previous actions that you want to provide as input to this action. These artifacts must already be defined as output artifacts in previous actions.

If you do not specify any input artifacts, then you must specify at least one source repository under *action-name*/Inputs/Sources.

For more information about artifacts, including examples, see [Working with artifacts (p. 286)](#).

**Note**

If the Artifacts - optional drop-down list is unavailable (visual editor), or if you get errors in when you validate your YAML (YAML editor), it might be because the action only supports one input. In this case, try removing the source input.

Variables - input

*(action-name*/Inputs/Variables)*

(Optional)

Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

For more information about variables, including examples, see [Working with variables (p. 337)](#).

Outputs

*(action-name*/Outputs)*

(Optional)

Defines the data that is output by the action during a workflow run.

Corresponding UI: Outputs tab

Artifacts - output

*(action-name*/Outputs/Artifacts)*

(Optional)

Specify the name of an artifact generated by the action. Artifact names must be unique within a workflow, and are limited to alphanumeric characters (a-z, A-Z, 0-9) and underscores (_). Spaces, hyphens (-), and other special characters are not allowed. You cannot use quotation marks to enable spaces, hyphens, and other special characters in output artifact names.

For more information about artifacts, including examples, see [Working with artifacts (p. 286)](#).
**Name**

*(action-name/Outputs/Artifacts/Name)*

(Required if Artifacts - output (p. 444) is included)

Specify the name of an artifact generated by the action. Artifact names must be unique within a workflow, and are limited to alphanumeric characters (a-z, A-Z, 0-9) and underscores (_). Spaces, hyphens (‐), and other special characters are not allowed. You cannot use quotation marks to enable spaces, hyphens, and other special characters in output artifact names.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

**Files**

*(action-name/Outputs/Artifacts/Files)*

(Required if Artifacts - output (p. 444) is included)

Specify the files that CodeCatalyst includes in the artifact that is output by the action. These files are generated by the workflow action when it runs, and are also available in your source repository. File paths can reside in a source repository or an artifact from a previous action, and are relative to the source repository or artifact root. You can use glob patterns to specify paths. Examples:

- To specify a single file that is in the root of your build location or source repository location, use *my-file.jar*.
- To specify a single file in a subdirectory, use *directory/my-file.jar* or *directory/subdirectory/my-file.jar*.
- To specify all files, use "**/**". The ** glob pattern indicates to match any number of subdirectories.
- To specify all files and directories in a directory named *directory*, use "directory/**/". The ** glob pattern indicates to match any number of subdirectories.
- To specify all files in a directory named *directory*, but not any of its subdirectories, use "directory/*".

**Note**

If your file path includes one or more asterisks (*) or other special character, enclose the path with double quotation marks (""). For more information about special characters, see Syntax guidelines and conventions (p. 355).

For more information about artifacts, including examples, see Working with artifacts (p. 286).

**Note**

You may need to add a prefix to the file path to indicate which artifact or source to find it in. For more information, see Referencing files in a source repository (p. 327) and Referencing files in an artifact (p. 289).

**Variables - output**

*(action-name/Outputs/Variables)*

(Optional)
Specify the variables that you want the action to export so that they are available for use by subsequent actions.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Outputs tab/Variables/Add variable

variable-name-1

(action-name/Outputs/Variables)

(Optional)

Specify the name of a variable that you want the action to export. This variable must already be defined in the Inputs or Steps section of the same action.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Outputs tab/Variables/Add variable/Name

AutoDiscoverReports

(action-name/Outputs/AutoDiscoverReports)

(Optional)

Defines the configuration for the auto-discovery feature.

When you enable auto-discovery, CodeCatalyst searches all Inputs passed into the action as well as all files generated by the action itself, looking for test, code coverage, and software composition analysis (SCA) reports. For each report that is found, CodeCatalyst transforms it into a CodeCatalyst report. A CodeCatalyst report is a report that is fully integrated into the CodeCatalyst service and can be viewed and manipulated through the CodeCatalyst console.

Note

By default, the auto-discover feature inspects all files. You can limit which files are inspected using the IncludePaths (p. 447) or ExcludePaths (p. 447) properties.

Corresponding UI: none

Enabled

(action-name/Outputs/AutoDiscoverReports/Enabled)

(Optional)

Enable or disable the auto-discovery feature.

Valid values are true or false.

If Enabled is omitted, the default is true.

Corresponding UI: Outputs tab/Reports/Automatically discover reports

ReportNamePrefix

(action-name/Outputs/AutoDiscoverReports/ReportNamePrefix)

(Required if AutoDiscoverReports (p. 446) is included and enabled)
Specify a prefix that CodeCatalyst prepends to all the reports it finds in order to name their associated CodeCatalyst reports. For example, if you specify a prefix of `AutoDiscovered`, and CodeCatalyst auto-discovers two test reports, `TestSuiteOne.xml` and `TestSuiteTwo.xml`, then the associated CodeCatalyst reports will be named `AutoDiscoveredTestSuiteOne` and `AutoDiscoveredTestSuiteTwo`.

Corresponding UI: Outputs tab/Reports/Automatically discover reports/Report prefix

**IncludePaths**

```
(action-name/Outputs/AutoDiscoverReports/IncludePaths)
```

Or

```
(action-name/Outputs/Reports/report-name-1/IncludePaths)
```

(Required if AutoDiscoverReports (p. 446) is included and enabled, or if Reports (p. 450) is included)

Specify the files and file paths that CodeCatalyst includes when searching for raw reports. For example, if you specify `"/test/report/*"`, CodeCatalyst searches the entire build image (p. 302) used by the action looking for the `/test/report/*` directory. When it finds that directory, CodeCatalyst then looks for reports in that directory.

**Note**

If your file path includes one or more asterisks (`*`) or other special characters, enclose the path with double quotation marks (`""`). For more information about special characters, see Syntax guidelines and conventions (p. 355).

If this property is omitted, the default is `"**/**"`, meaning the search includes all files at all paths.

**Note**

For manually configured reports, IncludePaths must be a glob pattern that matches a single file.

Corresponding UI:

- Outputs tab/Reports/Automatically discover reports/Include/exclude paths/Include paths
- Outputs tab/Reports/Manually configure reports/report-name-1/Include/exclude paths/Include paths

**ExcludePaths**

```
(action-name/Outputs/AutoDiscoverReports/ExcludePaths)
```

Or

```
(Outputs/Reports/report-name-1/ExcludePaths)
```

(Optional)

Specify the files and file paths that CodeCatalyst excludes when searching for raw reports. For example, if you specify `"/test/my-reports/**/*"`, CodeCatalyst will not search for files in the `/test/my-reports/` directory. To ignore all files in a directory, use the `**/*` glob pattern.

**Note**

If your file path includes one or more asterisks (`*`) or other special characters, enclose the path with double quotation marks (`""`). For more information about special characters, see Syntax guidelines and conventions (p. 355).

Corresponding UI:
• Outputs tab/Reports/Automatically discover reports/'Include/exclude paths'/Exclude paths
• Outputs tab/Reports/Manually configure reports/report-name-1/'Include/exclude paths'/Exclude paths

SuccessCriteria

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria)
Or
(action-name/Outputs/Reports/report-name-1/SuccessCriteria)
(Optional)
Specify the success criteria for the test, code coverage, software composition analysis (SCA), and static analysis (SA) reports.
For more information, see Configuring success criteria for reports (p. 149).
Corresponding UI:
• Outputs tab/Reports/Automatically discover reports/Success criteria
• Outputs tab/Reports/Manually configure reports/report-name-1/Success criteria

PassRate

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)
Or
(action-name/Outputs/Reports/report-name-1/SuccessCriteria/PassRate)
(Optional)
Specify the percentage of tests in a test report that must pass for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. The pass rate criteria are applied only to test reports. For more information about test reports, see Test reports (p. 147).
Corresponding UI:
• Outputs tab/Reports/Automatically discover reports/Pass rate
• Outputs tab/Reports/Manually configure reports/report-name-1/Success criteria/Pass rate

LineCoverage

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)
Or
(action-name/Outputs/Reports/report-name-1/SuccessCriteria/LineCoverage)
(Optional)
Specify the percentage of lines in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 448
60.5. Line coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see Code coverage reports (p. 147).

Corresponding UI:
- Outputs tab/Reports/Automatically discover reports/Success criteria/Line coverage
- Outputs tab/Reports/Manually configure reports/report-name-1/Success criteria/Line coverage

BranchCoverage

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)  
Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/BranchCoverage)

(Optional)

Specify the percentage of branches in a code coverage report that must be covered for the associated CodeCatalyst report to be marked as passed. Valid values include decimal numbers. For example: 50, 60.5. Branch coverage criteria are applied only to code coverage reports. For more information about code coverage reports, see Code coverage reports (p. 147).

Corresponding UI:
- Outputs tab/Reports/Automatically discover reports/Success criteria/Branch coverage
- Outputs tab/Reports/Manually configure reports/report-name-1/Success criteria/Branch coverage

Vulnerabilities

(action-name/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)  
Or

(action-name/Outputs/Reports/report-name-1/SuccessCriteria/Vulnerabilities)

(Optional)

Specify the maximum number and severity of vulnerabilities permitted in the SCA report for the associated CodeCatalyst report to be marked as passed. To specify vulnerabilities, you must specify:

- The minimum severity of the vulnerabilities you want to include in the count. Valid values, from most to least severe, are: CRITICAL, HIGH, MEDIUM, LOW, INFORMATIONAL.
  
  For example, if you choose HIGH, then HIGH and CRITICAL vulnerabilities will be tallied.
  
- The maximum number of vulnerabilities of the specified severity you want permit. Exceeding this number causes the CodeCatalyst report to be marked as failed. Valid values are whole numbers.

Vulnerabilities criteria are applied only to SCA reports. For more information about SCA reports, see Software composition analysis reports (p. 148).

To specify the minimum severity, use the Severity property. To specify the maximum number of vulnerabilities, use the Number property.

For more information about SCA reports, see Test report types (p. 147).
Corresponding UI:

- Outputs tab/Reports/Automatically discover reports/Success criteria/Vulnerabilities
- Outputs tab/Reports/Manually configure reports/report-name-1/Success criteria/Vulnerabilities

## Reports

*(action-name)/Outputs/Reports*

(Optional)

A section that specifies the configuration for test reports.

Corresponding UI: Outputs tab/Reports

### report-name-1

*(action-name)/Outputs/Reports/report-name-1*

(Required if Reports (p. 450) is included)

The name you want to give to the CodeCatalyst report that will be generated from your raw reports.

Corresponding UI: Outputs tab/Reports/Manually configure reports/Report name

## Format

*(action-name)/Outputs/Reports/report-name-1/Format*

(Required if Reports (p. 450) is included)

Specify the file format that you're using for your reports. Possible values are as follows.

- For test reports:
  - For Cucumber JSON, specify Cucumber (visual editor) or CUCUMBERJSON (YAML editor).
  - For JUnit XML, specify JUnit (visual editor) or JUNITXML (YAML editor).
  - For NUnit XML, specify NUnit (visual editor) or NUNITXML (YAML editor).
  - For NUnit 3 XML, specify NUnit3 (visual editor) or NUNIT3XML (YAML editor).
  - For Visual Studio TRX, specify Visual Studio TRX (visual editor) or VISUALSTUDIO TRX (YAML editor).
  - For TestNG XML, specify TestNG (visual editor) or TESTNGXML (YAML editor).
- For code coverage reports:
  - For Clover XML, specify Clover (visual editor) or CLOVERXML (YAML editor).
  - For Cobertura XML, specify Cobertura (visual editor) or COBERTURAXML (YAML editor).
  - For JaCoCo XML, specify JaCoCo (visual editor) or JACOCOXML (YAML editor).
  - For SimpleCov JSON generated by simplecov, not simplecov-json, specify Simplecov (visual editor) or SIMPL ECOV (YAML editor).
  - For software composition analysis (SCA) reports:
    - For SARIF, specify SARIF (visual editor) or SARIFSCA (YAML editor).

Corresponding UI: Outputs tab/Reports/Manually configure reports/Add report/report-name-1/Report type and Report format
**Configuration**

(*action-name*/Configuration)

(Required) A section where you can define the configuration properties of the action.

Corresponding UI: **Configuration** tab

**Steps**

(*action-name*/Configuration/Steps)

(Required)

Specify your GitHub Action code as it appears on the action's details page in [GitHub Marketplace](https://github.com). Add the code following these guidelines:

1. Paste the code from the GitHub Action's `steps:` section into the `Steps:` section of the CodeCatalyst workflow. The code starts with a dash (-) and looks similar to the following.

   **GitHub code to paste:**

   ```yaml
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: master
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

2. Review the code you just pasted and modify it as necessary so that it conforms to CodeCatalyst standards. For example, with the preceding code block, you might remove the code in *red italics*, and add the code in **bold**.

   **CodeCatalyst workflow yaml:**

   ```yaml
   Steps:
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: master
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

3. For additional code that's included with the GitHub Action but does not exist inside the `steps:` section, add it to the CodeCatalyst workflow using CodeCatalyst-equivalent code. You can review the Workflow definition reference (p. 354) to gain insight into how you might port your GitHub code to CodeCatalyst. Detailed migration steps are outside the scope of this guide.

Here is an example of how to specify file paths in a **GitHub Actions** action:

```yaml
Steps:
- name: Lint Code Base
  uses: github/super-linter@v4
  ...
- run: cd /sources/WorkflowSource/MyFolder/ && cat file.txt
- run: cd /artifacts/MyGitHubAction/MyArtifact/MyFolder/ && cat file2.txt
```

For more information about specifying file paths, see [Referencing files in a source repository](https://example.com) (p. 327) and [Referencing files in an artifact](https://example.com) (p. 289).
Amazon CodeCatalyst User Guide
"Render task definition" action reference

Corresponding UI: Configuration tab/GitHub Actions YAML

"Render Amazon ECS task definition" action reference

The following is the action definition YAML reference for the Render Amazon ECS task definition action. To learn how to use this action, see Adding the "Render Amazon ECS task definition" action (p. 260).

Note
Most of the YAML properties that follow have corresponding UI elements in the visual editor. To look up a UI element, use Ctrl+F. The element will be listed with its associated YAML property.

```yaml
# The workflow definition starts here.
# See Top-level properties for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
ECSRenderTaskDefinition_nn (p. 452):
  Identifier (p. 453): aws/ecs-render-task-definition@v1
  DependsOn (p. 453):
    - build-action
  Compute (p. 455):
    Type (p. 453): EC2 | Lambda
    Fleet (p. 455): fleet-name
  Timeout (p. 456): timeout-minutes
  Inputs (p. 454):
    # Specify a source or an artifact, but not both.
    Sources (p. 456):
      - source-name-1
    Artifacts: (p. 455)
      - task-definition-artifact
    Variables: (p. 455)
      - Name: variable-name-1
        Value: variable-value-1
      - Name: variable-name-2
        Value: variable-value-2
  Configuration (p. 455)
    task-definition (p. 455): task-definition-path
    container-definition-name (p. 455): container-definition-name
    image (p. 456): docker-image-name
    environment-variables (p. 456):
      - variable-name-1=variable-value-1
      - variable-name-2=variable-value-2
  Outputs: (p. 456)
    Artifacts (p. 457):
      - Name (p. 457): TaskDefArtifact
        Files (p. 457): "task-definition*"
    Variables (p. 457):
      - task-definition
```

ECSRenderTaskDefinition

(Required)

Specify the name of the action. All action names must be unique within the workflow. Action names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in action names.

Default: ECSRenderTaskDefinition_nn.
**Identifier**

(\texttt{ECSRenderTaskDefinition/Identifier})

(Required)

Identifies the action. Do not change this property unless you want to change the version. For more information, see [Working with action versions](#).

Default: \texttt{aws/ecs-render-task-definition@v1}.

**DependsOn**

(\texttt{ECSRenderTaskDefinition/DependsOn})

(Optional)

Specify an action or action group that must run successfully in order for this action to run.

For more information about the ‘depends on’ functionality, see [Configuring actions to depend on other actions](#).

**Compute**

(\texttt{ECSRenderTaskDefinition/Compute})

(Optional)

The computing engine used to run your workflow actions. You can specify compute either at the workflow level or at the action level, but not both. When specified at the workflow level, the compute configuration applies to all actions defined in the workflow. At the workflow level, you can also run multiple actions on the same instance. For more information, see [Sharing compute across actions](#).

**Type**

(\texttt{ECSRenderTaskDefinition/Compute/Type})

(Required if Compute (p. 453) is included)

The type of compute engine. You can use one of the following values:

- **EC2** (visual editor) or EC2 (YAML editor)
  
  Optimized for flexibility during action runs.

- **Lambda** (visual editor) or Lambda (YAML editor)
  
  Optimized action start-up speeds.

For more information about compute types, see [About compute types](#).
Fleet

(ECSRenderTaskDefinition/Compute/Fleet)

(Optional)
Specify the machine or fleet that will run your workflow or workflow actions. With on-demand fleets, when an action starts, the workflow provisions the resources it needs, and the machines are destroyed when the action finishes. Examples of on-demand fleets: Linux.x86-64.Large, Linux.x86-64.XLarge. For more information about on-demand fleets, see On-demand fleet properties (p. 295).

With provisioned fleets, you configure a set of dedicated machines to run your workflow actions. These machines remain idle, ready to process actions immediately. For more information about provisioned fleets, see Provisioned fleet properties (p. 295).

If Fleet is omitted, the default is Linux.x86-64.Large.

Corresponding UI: Configuration tab/Compute fleet

Timeout

(ECSRenderTaskDefinition/Timeout)

(Optional)
Specify the amount of time in minutes (YAML editor), or hours and minutes (visual editor), that the action can run before CodeCatalyst ends the action. The minimum is 5 minutes and the maximum is described in Quotas for workflows in CodeCatalyst (p. 457). The default timeout is the same as the maximum timeout.

Corresponding UI: Configuration tab/Timeout - optional

Inputs

(ECSRenderTaskDefinition/Inputs)

(Optional)
The Inputs section defines the data that the ECSRenderTaskDefinition needs during a workflow run.

Note
Only one input (either a source or an artifact) is allowed per Render Amazon ECS task definition action. Variables do not count towards this total.

Corresponding UI: Inputs tab

Sources

(ECSRenderTaskDefinition/Inputs/Sources)

(Required if your task definition file is stored in a source repository)
If your task definition file is stored in a source repository, specify the label of that source repository. Currently, the only supported label is WorkflowSource.

If your task definition file is not contained within a source repository, it must reside in an artifact generated by another action.
For more information about sources, see Working with sources (p. 325).

Corresponding UI: Inputs tab/Sources - optional

Artifacts - input

(ECSRenderTaskDefinition/Inputs/Artifacts)

(Required if your task definition file is stored in an output artifact (p. 286) from a previous action)

If the task definition file that you want to deploy is contained in an artifact generated by a previous action, specify that artifact here. If your task definition file is not contained within an artifact, it must reside in your source repository.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Configuration tab/Artifacts - optional

Variables - input

(ECSRenderTaskDefinition/Inputs/Variables)

(Required)

Specify a sequence of name/value pairs that define the input variables that you want to make available to the action. Variable names are limited to alphanumeric characters (a-z, A-Z, 0-9), hyphens (-), and underscores (_). Spaces are not allowed. You cannot use quotation marks to enable special characters and spaces in variable names.

For more information about variables, including examples, see Working with variables (p. 337).

Corresponding UI: Inputs tab/Variables - optional

Configuration

(ECSRenderTaskDefinition/Configuration)

(Required)

A section where you can define the configuration properties of the action.

Corresponding UI: Configuration tab

task-definition

(ECSRenderTaskDefinition/Configuration/task-definition)

(Required)

Specify the path to an existing task definition file. If the file resides in your source repository, the path is relative to the source repository root folder. If your file resides in an artifact from a previous workflow action, the path is relative to the artifact root folder. For more information about task definition files, see Task definitions in the Amazon Elastic Container Service Developer Guide.

Corresponding UI: Configuration tab/Task definition

container-definition-name

(ECSRenderTaskDefinition/Configuration/container-definition-name)
(Required)

Specify the name of the container where your Docker image will run. You can find this name in the containerDefinitions, name field in your task definition file. For more information, see Name in the Amazon Elastic Container Service Developer Guide.

Corresponding UI: Configuration tab/Container name

image

(ECSRenderTaskDefinition/Configuration/image)

(Required)

Specify the name of the Docker image that you want the Render Amazon ECS task definition action to add to your task definition file. The action adds this name to the containerDefinitions, image field in your task definition file. If a value already exists in the image field, then the action overwrites it. You can include variables in the image name.

Examples:

If you specify MyDockerImage:${WorkflowSource.CommitId}, the action adds MyDockerImage:commit-id to the task definition file, where commit-id is a commit ID generated at runtime by the workflow.

If you specify my-ecr-repo/image-repo:$(date +%m-%d-%y-%H-%m-%s), the action adds my-ecr-repo/image-repo:timestamp to the task definition file, where my-ecr-repo is the URI of an Amazon Elastic Container Registry (ECR) and timestamp is a timestamp in the format month-day-year-hour-minute-second generated at runtime by the workflow.

For more information about the image field, see Image in the Amazon Elastic Container Service Developer Guide. For more information about variables, see Working with variables (p. 337).

Corresponding UI: Configuration tab/Image name

environment-variables

(ECSRenderTaskDefinition/Configuration/environment-variables)

(Required)

Specify environment variables that you want the Render Amazon ECS task definition action to add to your task definition file. The action adds the variables to the containerDefinitions, environment field in your task definition file. If variables already exist in the file, the action overwrites the values of existing variables and adds any new variables. For more information about Amazon ECS environment variables, see Specifying environment variables in the Amazon Elastic Container Service Developer Guide.

Corresponding UI: Configuration tab/Environment variables - optional

Outputs

(ECSRenderTaskDefinition/Outputs)

(Required)

Defines the data that is output by the action during a workflow run.

Corresponding UI: Outputs tab
Artifacts

\((ECSRenderTaskDefinition/Outputs/Artifacts)\)

(Required)

Specify the artifacts generated by the action. You can reference these artifacts as input in other actions.

For more information about artifacts, including examples, see Working with artifacts (p. 286).

Corresponding UI: Outputs tab/Artifacts

Name

\((ECSRenderTaskDefinition/Outputs/Artifacts/Name)\)

(Required)

Specify the name of the artifact that will contain the updated task definition file. The default value is MyTaskDefinitionArtifact. You must then specify this artifact as input into the Deploy to Amazon ECS action. To understand how to add this artifact as input to the Deploy to Amazon ECS action, see Example workflow (p. 261).

Corresponding UI: Outputs tab/Artifacts/Name

Files

\((ECSRenderTaskDefinition/Outputs/Artifacts/Files)\)

(Required)

Specify the files to include in the artifact. You must specify task-definition-* so that the updated task definition file, which starts with task-definition-, will be included.

Corresponding UI: Outputs tab/Artifacts/Files

Variables

\((ECSRenderTaskDefinition/Outputs/Variables)\)

(Required)

Specify the name of a variable to be set by the render action. The render action will set this variable's value to the name of the updated task definition file (for example, task-definition-random-string.json). You must then specify this variable in the Deploy to Amazon ECS action's Task definition (visual editor) or task-definition (yaml editor) property. To understand how to add this variable to the Deploy to Amazon ECS action, see Example workflow (p. 261).

Default: task-definition

Corresponding UI: Outputs tab/Variables/Name field

Quotas for workflows in CodeCatalyst

The following table describes quotas and limits for workflows in Amazon CodeCatalyst.

For more information about quotas in Amazon CodeCatalyst, see Quotas for CodeCatalyst (p. 620).
### Quotas for workflows

<table>
<thead>
<tr>
<th>Quota Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of workflows per space</td>
<td>800</td>
</tr>
<tr>
<td>Maximum workflow definition file size</td>
<td>256 KB</td>
</tr>
<tr>
<td>Maximum number of workflow files processed in a single source event</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of files processed in a single source event</td>
<td>4,000</td>
</tr>
<tr>
<td>Maximum number of active fleets per space</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of active compute instances per fleet</td>
<td>20</td>
</tr>
<tr>
<td>Maximum number of input artifacts per action</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of output artifacts per action</td>
<td>10</td>
</tr>
<tr>
<td>Maximum total size of a single action's output variables</td>
<td>120 KB</td>
</tr>
<tr>
<td>Maximum length of an output variable value</td>
<td>500 characters or more, depending on the action that emits the value.</td>
</tr>
<tr>
<td>Note: Values may be truncated if they exceed the action's limit.</td>
<td></td>
</tr>
<tr>
<td>Maximum number of days to keep artifacts generated during a workflow run</td>
<td>30</td>
</tr>
<tr>
<td>Maximum number of reports per action</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of test cases per test report</td>
<td>20,000</td>
</tr>
<tr>
<td>Maximum number of files per code coverage report</td>
<td>20,000</td>
</tr>
<tr>
<td>Maximum number of software composition analysis findings per report</td>
<td>20,000</td>
</tr>
<tr>
<td>Maximum number of files per static analysis report</td>
<td>20,000</td>
</tr>
<tr>
<td>Maximum number of concurrent workflow runs per space</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of actions running concurrently per workflow</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of actions running concurrently per space</td>
<td>200</td>
</tr>
<tr>
<td>Maximum amount of time an action can run</td>
<td>For the build and test actions, the timeout is 8 hours. For all other actions, the timeout is 1 hour.</td>
</tr>
<tr>
<td>Maximum number of secrets per space</td>
<td>500,000</td>
</tr>
</tbody>
</table>
Issues in CodeCatalyst

In CodeCatalyst, you can monitor features, tasks, bugs, and any other work involved in your project. Each piece of work is kept in a distinct record called an issue. Each issue can have a description, assignee, status, and other properties, which you can search for, group and filter on. You can view your issues using the default views, or you can create your own views with custom filtering, sorting, or grouping. For more information about concepts related to issues, see Issues concepts (p. 460). To learn how to create your first issue, see Creating an issue in CodeCatalyst (p. 461).

Here is one possible workflow for a team using issues:

Jorge Souza is a developer working in a project. He and his fellow project members Li Juan, Mateo Jackson, and Wang Xiulan collaborate to determine what work needs to be done. Every day, he and his fellow developers hold a sync-up meeting, led by Wang Xiulan. They pull up the board by navigating to one of their teams views of the board. By creating views, users and teams can save filters, groupings, and sorting of issues to easily view issues that meet their specified criteria. Their view contains issues grouped by Assignee and sorted by Priority to show the most important issues and status of the issues for each developer. As Jorge is assigned tasks to complete, he plans his work by creating an issue for each task. When creating issues, Jorge can choose the appropriate Status, Priority, and work Estimation effort. Jorge creates his issues with a draft status, such as backlog, as he doesn't plan to start on them immediately. Issues in a draft status appear on the Drafts view where they are to be planned and prioritized. Once Jorge is ready to start the work, he moves the corresponding issue to the board by updating its status to a status in another category (Not Started, Started, or Completed). As each task is being worked on, the team can filter by the title, status, assignee, label, priority, and estimation to find a specific issue or similar issues that match the specified parameter. Using the board, he tracks the day-to-day progress by dragging each issue from one status to the next until the task is complete. As the project progresses, finished issues accumulate in the Completed status. Wang Xiulan decides to remove them from view by archiving them using the quick archive button, so that the developers can focus on the issues that are related to current and upcoming work.

When planning their work, the developers working on the project choose Sort by and Group by to find the issues they want to move from the backlog to the board. They might choose to add issues to the board based on the highest priority customer requests, so they group the board by a Customer request label and sort by Priority. They might also sort by estimate to ensure that they're taking on a volume of work they can achieve. The project manager, Saanvi Sarkar, regularly reviews and grooms the backlog to help ensure that the priority accurately reflects the importance of each issue to the success of the project.

Topics
- Issues concepts (p. 460)
- Creating an issue in CodeCatalyst (p. 461)
- Editing and collaborating on issues in CodeCatalyst (p. 462)
- Finding and viewing issues (p. 468)
- Exporting issues (p. 470)
- Configuring issues settings (p. 470)
- Markdown tips and tricks (p. 474)
- Quotas for issues in CodeCatalyst (p. 475)
Issues concepts

Creating an issue is a quick and efficient way to track work being done within a project. You can use issues to help you discuss work in daily sync-up meetings, prioritize work, and more.

This page includes a list of concepts that will help you effectively use issues in CodeCatalyst.

Active issues

Active issues are issues that are any issues that are not in a Draft status or archived. In other words, active issues are issues with a status in any of the following status categories: Not started, Started, and Completed. For more information about statuses and status categories, see Status and status categories (p. 461).

You can view all of the active issues in your project from the default Active issues view.

Archived issues

An archived issue is an issue that is no longer relevant to your project. For example, you can archive an issue (p. 467) if it is completed and you no longer need to see it in the Done column, or if it was created in error. Archived issues can be unarchived if needed.

Assignee

The assignee is the person the issue is assigned to. If the person doesn't appear in the list when you search for them, they have not been added to your project. To add them, see Inviting a user to your project (p. 63). To enable multiple assignees to an issue, see Enabling or disabling multiple assignees (p. 471). Issues with multiple assignees will appear on your board with different colored avatars, each representing one of the assignees.

Custom fields

Custom fields allow you to customize different attributes of an issue according to your needs for tracking and maintaining issues within a project. For example, you can add a field for roadmapping, a specific due date, or a requester field.

Estimate

In agile development, the estimate is known as story points. Use the estimate to include the amount of work required, in addition to the ambiguity and complexity of the issue. Use higher estimates for issues with greater risk, difficulty, and unknowns. You can change the estimation type for your project. For more information about the estimation types and how to configure them, see Configuring issue effort estimation (p. 471).

Issue

An issue is a record that tracks the work related to your project. You can create an issue for a feature, a task, a bug, or any other body of work related to your project. If you're using agile development, an issue can also describe an epic or user story.

Label

The label is used to group, sort and filter issues. You can enter a new label name or choose one of the labels from the populated list. This list consists of recently used labels in the project. An issue can have multiple labels, and a label can be removed from an issue. To customize labels, see Labels (p. 473).
Priority

Priority refers to the level of importance of the issue. There are four options: Low, Medium, High, and No priority.

Status and status categories

The status is the current state of the issue and is used to quickly check the progress of an issue through its lifecycle, from inception to completion. All issues must have a status, and each status belongs to a status category. Status categories are used to help organize your statuses and populate the default issue views.

There are five default statuses and four status categories in CodeCatalyst. You can create other statuses, but you cannot create other status categories. The following list contains the default statuses and their status categories in parenthesis: Backlog (Draft), To do (Not started), In progress (Started), In review (Started), and Done (Completed).

For more information on working with statuses, see Statuses (p. 471).

Views

Issues in your CodeCatalyst project are displayed in views. Views can either be grid views that show issues in list format or board views that show issues as tiles in columns organized by issue status. There are four default views, and you can create your own views with custom grouping, filtering, and sorting (p. 470). The following list contains details about the four default views.

- The Drafts view is a grid view that shows issues not currently being worked on. Any issue created with a status in the Draft status category shows up in this view. This view can be used by teams to see which issues are still being defined or are waiting to be assigned and worked on.
- The Active issues view is a board view of all issues that are currently being worked on. Any issue with a status in the Not started, Started, or Completed status categories will show up in this view.
- The All issues view is a grid view that shows all the issues in the project, both drafts and active issues.
- The Archived view shows all archived issues.

Creating an issue in CodeCatalyst

You can create issues within a project based on your needs. For example, you could create an issue to track updating a variable in your code.

Follow these instructions to create an issue in CodeCatalyst.

To create an issue

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to the project where you want to create an issue.
3. On the project home page, choose Create issue. Alternatively, in the navigation pane, choose Issues.
4. Choose Create issue.

Note
You can also add issues inline when using a grid view.

5. Enter a title for the issue.
6. (Optional) Enter a Description. You can use Markdown to add formatting. For more information, see Markdown tips and tricks (p. 474).
7. (Optional) Choose a **Status, Priority, Estimation** for the issue.
8. (Optional) Add an existing label or create a new label and add it by choosing + **Add label**.
   a. To add an existing label, choose the label from the list. You can enter a search term in the field to search all labels containing that term in the project.
   b. To create a new label and add it, enter the name of the label you want to create in the search field and press enter.
9. (Optional) Add an assignee by choosing + **Add an assignee**. You can quickly add yourself as the assignee by choosing + **Add me**.
10. (Optional) Add an existing **custom field** or create a new custom field. Issues can have multiple custom fields.
    a. To add an existing custom field, choose the custom field from the list. You can enter a search term in the field to search all custom fields containing that term in the project.
    b. To create a new custom field and add it, enter the name of the custom field you want to create in the search field and press enter. Then choose the type of custom field you want to create and set a value.
11. Choose **Create issue**. A notification appears in the lower right corner: If the issue was created successfully, a confirmation message appears saying the issue was successfully created. If the issue was not created successfully, an error message with the reason for the failure appears. You can then choose **Retry** to edit and retry creating the issue, or choose **Discard** to discard the issue. Both options will dismiss the notification.

    **Note**
    You cannot link a pull request to an issue when you create it. However, you can edit it (p. 462) after you create it to add links to pull requests.

## Editing and collaborating on issues in CodeCatalyst

### Contents
- Editing an issue (p. 462)
- Working with attachments (p. 463)
- Marking an issue as blocked or unblocked (p. 464)
- Adding, editing, or deleting comments (p. 465)
  - Using mentions in a comment (p. 465)
- Progressing an issue (p. 466)
  - Between the backlog and board (p. 466)
  - Progress an issue through lifecycle stages on the board (p. 466)
  - Moving issues between groups (p. 467)
- Archiving an issue (p. 467)

### Editing an issue

Follow these steps to edit the title, description, status, assignee, priority, estimate, or labels of an issue.

#### To edit an issue

1. Choose the issue that you want to edit to view the issue details. For help on finding your issue, see Finding and viewing issues (p. 468).
2. To edit the issue title, choose the title.
3. Edit the title and press **Enter**.

4. To edit the description, choose the description. You can use Markdown to add formatting. For more information, see [Markdown tips and tricks](p. 474).

5. Edit the description and press **Enter**.

6. To edit the **Status**, **Estimate**, or **Priority**, choose an option from the respective dropdown menus.

7. In **Labels**, you can add an existing label, create a new label, or remove a label.
   a. To add an existing label, choose + **Add label** and choose the label from the list. You can enter a search term in the field to search all labels containing that term in the project.
   b. To create a new label and add it, choose + **Add label** enter the name of the label you want to create in the search field and press enter.
   c. To remove a label, choose the X icon next to the label you want to remove. If you remove a label from all issues, the label will appear in the **Unused labels** section in the **Labels** section of issue **settings**. Unused labels appear at the end of the list of labels when using filters or adding labels to an issue. You can find an overview of all labels (used and unused) and issues that have them in the issue **settings**.

8. To assign an issue, choose + **Add an assignee** in the **Assignee** section, then search and choose the assignee from the list. You can choose + **Add me** to quickly add yourself as the assignee.

9. In **Attachments**, you can add, download, or remove attachments. For more information, see [Working with attachments](p. 463).

10. To link a pull request, choose **Link pull request**, and then either choose a pull request from the list or enter its URL or ID. To unlink a pull request, choose the unlink icon.
    
    **Tip**
    After you add a link to a pull request to an issue, you can quickly navigate to it by choosing its ID in the list of linked pull requests. You can use the URL of a pull request to link pull requests that are in different projects than the issue board, but only users that are members of that project will be able to view or navigate to that pull request.

11. (Optional) Add and set an existing custom field, create a new custom field, or remove a custom field. Issues can have multiple custom fields.
    a. To add an existing custom field, choose the custom field from the list. You can enter a search term in the field to search all custom fields containing that term in the project.
    b. To create a new custom field and add it, enter the name of the custom field you want to create in the search field and press enter. Then choose the type of custom field you want to create and set a value.
    c. To remove a custom field, choose the X icon next to the custom field you want to remove. If you remove a custom field from all issues, the custom field will be deleted and you will no longer see it when filtering.

---

**Working with attachments**

You can add attachments to issues in CodeCatalyst to make related files easily accessible. Use the following procedure to manage attachments for an issue.

The size of attachments added to issues counts towards your space's storage quotas. For information about viewing and managing attachments for your project, see [Viewing and managing attachments](p. 474).

**Important**
Attachments to issues are not scanned or analyzed by Amazon CodeCatalyst. Any user could add an attachment to an issue that might potentially contain malicious code or content. Make sure that users are aware of best practices when it comes to managing attachments and guarding against malicious code, content, or viruses.
To add, download, or remove attachments

1. Choose the issue for which you want to manage attachments. For help on finding your issue, see Finding and viewing issues (p. 468).
2. To add an attachment, choose Upload file. Navigate to the file in your operating system's file explorer and select it. Choose Open to add it as an attachment. For quota information, such as maximum attachment size, see Quotas for issues in CodeCatalyst (p. 475).

   Note the following restrictions to attachment file names and content types:
   - The following characters are not permitted in file names:
     - Control characters: 0x00–0x1f and 0x80–0x9f
     - Reserved characters: /, ?, <, >, \, :, *, |, and "
     - Unix reserved filenames: . and ..
     - Trailing periods and spaces
     - Windows reserved filenames: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9
   - The content type of the attachment must adhere to the following media-type pattern:

     ```
     media-type = type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
     ```

     For example, text/html; charset=UTF-8.

3. To download an attachment, choose the ellipses menu next to the attachment you want to download and choose Download.
4. To copy an attachment's URL, choose the ellipses menu next to the attachment of which you want to copy the URL and choose Copy URL.
5. To remove an attachment, choose the ellipses menu next to the attachment you want to remove and choose Delete.

Marking an issue as blocked or unblocked

If something is preventing you from working on an issue, you might want to mark it as blocked. For example, your issue might be blocked if it relies on a change to another part of your code base that hasn't been merged yet.

When you mark an issue as blocked, CodeCatalyst adds a red Blocked label to the issue, making it highly visible in your backlog or archive, or on your board.

You can unblock the issue when outside circumstances are resolved.

To mark an issue as blocked

1. Open the issue you want to mark as blocked. For help with finding your issue, see Finding and viewing issues (p. 468).
2. Choose Actions, and then choose Mark as blocked.

To unblock an issue

1. Open the issue that you want to unblock. For help with finding your issue, see Finding and viewing issues (p. 468).
2. Choose Actions, and then choose Mark as unblocked.
Adding, editing, or deleting comments

You can leave a comment on an issue. In the comments you can tag other space members, other projects in the space, related issues, and code.

To add a comment to an issue

1. Navigate to your project.
2. In the navigation bar choose Issues.
3. Choose the issue where you want to add the comment. For help on finding your issue, see Finding and viewing issues (p. 468).
4. Enter the comment in the Comments field. For information on formatting comments, see Markdown tips and tricks (p. 474).
5. Choose Send.

To edit a comment

You can edit comments you make on issues. You can only edit comment that you authored.

1. Navigate to your project.
2. In the navigation bar choose Issues.
3. Choose the issue where you want to edit a comment. For help on finding your issue, see Finding and viewing issues (p. 468).
4. To edit a comment, find the comment you want to edit.
   
   **Tip**
   
   You can sort comments by oldest or newest first. Comments are loaded 10 at a time.
5. Choose the ellipsis icon, then choose Edit.
6. Edit the comment. For information on formatting comments, see Markdown tips and tricks (p. 474).
7. Choose Save. The comment is now updated.

To delete a comment

You can delete comments you make on issues. You can only delete comments that you authored. When a comment is deleted, your user name will show, but with the words This comment has been deleted in place of the original comment text.

1. Navigate to your project.
2. In the navigation bar choose Issues.
3. Choose the issue where you want to delete a comment. For help on finding your issue, see Finding and viewing issues (p. 468).
4. Choose the ellipsis icon, choose Delete, and then choose Confirm.

Using mentions in a comment

You can mention space members, other projects in the space, related issues, and code in comments. Doing so creates a quick link to the user or resource you mention.

To @mention in a comment

1. Navigate to your project.
2. In the navigation bar choose **Issues**.
3. Choose the issue that you want to edit to view the issue details. For help on finding your issue, see *Finding and viewing issues (p. 468).*
4. Choose the **Add a comment** textbox.
5. Type `@user_name` to mention another user.
6. Type `@project_name` to mention a project.
7. Type `@issue_name` or `@issue_number` to mention another issue.
8. Type `@file_name` to mention specific files or code in a source repository.

**Note**
A list of the top 5 items (users, source repositories, projects, etc) containing the terms that match your @mention will populate as you type.

9. Choose the desired item you would like to mention. The pathway showing where the item is located will populate in the comment textbox.
10. Finish your comment and choose **Send**.

**Progressing an issue**

Every issue has a lifecycle. In CodeCatalyst, issues typically start as a draft in the backlog. When work for that issue is to be started, it is moved into another status category and moves through various statuses until it is complete, and then it is archived. You can move or progress an issue through its lifecycle in the following ways:

- You can move an issue between the backlog and the board.
- You can move in-progress issue through various completion stages.
- You can archive an issue that is completed.

**Between the backlog and board**

You can move an issue from the backlog to the board once you begin to work on the issue. You can also move an issue back to the backlog if the work is postponed.

**To move an issue between the backlog and the board**

1. Navigate to your project.
2. In the navigation pane, choose **Issues**. The default view is the **Board**.
3. To move an issue from the board to the backlog:
   a. Choose the issue that you want to move. For help with finding your issue, see *Finding and viewing issues (p. 468).*
   b. Choose **Backlog** from the dropdown **Status** menu.
4. To move an issue from the backlog to the board:
   a. To navigate to the backlog, choose **Board** and choose **Backlog**.
   b. Choose the issue that you want to move. For help with finding your issue, see *Finding and viewing issues (p. 468).*
   c. Choose **Add to board**, or choose a **Status** other than **Backlog**.

**Progress an issue through lifecycle stages on the board**

You can move an issue within a board through different statuses until completion.
To move an issue within the board

1. In the navigation pane, choose Issues. The default view is the Board.
2. Do one of the following:
   - Drag and drop an issue to another status.
   - Choose an issue, and then choose a status from the Status dropdown menu.
   - Choose an issue, and then choose Move to: next-status.

For information on archiving an issue, see Archiving an issue (p. 467).

Moving issues between groups

You can group issues (p. 469) in the All issues and Board views by various parameters. If the issues are grouped, you can move issues from one group to another. Moving an issue from one group to another will automatically edit the field that the issues are grouped on to match the target group.

As an example scenario, assume there is a company using CodeCatalyst that has issues assigned to two people, Wang Xiulan and Saanvi Sarkar. The board is grouped by Assignee, and there are two groups, one for each assignee. Moving an issue from the Wang Xiulan group to the Saanvi Sarkar group will update the issue’s assignee to Saanvi Sarkar.

Archiving an issue

Note
Issues are not deleted within a project, they are archived. To delete issues, you must delete the project.

You can archive an issue when it is no longer needed in your project. When you archive an issue, CodeCatalyst removes it from all views that filter out archived issues. Archived issues can be viewed in the Archived issues default view, where they can be unarchived if needed.

You archive an issue if:

- You have completed the issue and no longer need it in the Done column.
- You have no plans to work on it.
- You created it in error.
- You have reached the maximum number of active issues.

To archive an issue

1. Open the issue you want to archive. For help with finding your issue, see Finding and viewing issues (p. 468).
2. Choose Actions, and then choose Move to archive.
3. (Optional) To quickly archive multiple issues with a Completed status, choose the vertical ellipsis at the top of any Completed status on the board and choose Archive issues.

To unarchive an issue

1. Open the issue that you want to unarchive. You can view a list of archived issues by opening the Archived issues view from the issues view switcher dropdown menu. For help with finding your issue, see Finding and viewing issues (p. 468).
2. Choose Unarchive.
Finding and viewing issues

The following sections describe how to effectively search for and view issues within a CodeCatalyst project.

Issues in your CodeCatalyst project are displayed in views. Views can either be grid views that show issues in list format or board views that show issues as tiles in columns organized by issue status. There are four default views, and you can create your own views with custom grouping, filtering, and sorting (p. 470). The following list contains details about the four default views.

- The **Drafts** view is a grid view that shows issues not currently being worked on. Any issue created with a status in the **Draft** status category shows up in this view. This view can be used by teams to see which issues are still being defined or are waiting to be assigned and worked on.
- The **Active issues** view is a board view of all issues that are currently being worked on. Any issue with a status in the **Not started**, **Started**, or **Completed** status categories will show up in this view.
- The **All issues** view is a grid view that shows all the issues in the project, both drafts and active issues.
- The **Archived** view shows all archived issues.

### Searching for an issue

You can find an issue by searching for specific parameters. For more information about refining your search, see Search in CodeCatalyst (p. 576).

**To search for an issue**

1. Navigate to your project.
2. Use the search bar to search for issues or information related to issues. You can use query parameters to refine your search. For more information, see Search in CodeCatalyst (p. 576).

### Sorting issues

By default, issues in CodeCatalyst are sorted by **Manual order**. Manual order displays issues in the order they are moved to by users. You can drag and drop issues when sorted in Manual order to change their order. This sorting option is helpful when grooming the issues backlog and prioritizing issues.

The following table shows how issues can be sorted in both grid and board views.

<table>
<thead>
<tr>
<th>Grid view sorting options</th>
<th>Board view sorting options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual order</td>
<td>Manual order</td>
</tr>
<tr>
<td>Last updated</td>
<td>Last updated</td>
</tr>
<tr>
<td>Priority</td>
<td>Priority</td>
</tr>
<tr>
<td>Estimate</td>
<td>Estimate</td>
</tr>
<tr>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Blocked</td>
<td></td>
</tr>
</tbody>
</table>
Use the following procedure to change how issues are sorted.

**To sort issues**

1. Navigate to your project.
2. In the navigation pane, choose **Issues**. The default view is the **Board**.
3. (Optional) Choose **Active issues** to open the **issues view switcher** dropdown menu to navigate to a different issues view.
4. To sort a grid view, there are two options:
   a. Choose the **header** of the field you want to sort by. Choosing the **header** will cycle between ascending and descending order.
   b. Choose the **Sort by** dropdown menu and choose a parameter to sort by. Issues will be sorted in ascending order.
5. To sort a board view, choose the **Sort by** dropdown menu and choose a parameter to sort by. Issues will be sorted in ascending order.

**Grouping issues**

Grouping is used to organize issues on the board by multiple parameters, such as assignee, labels, and priority.

**To group issues**

1. Navigate to your project.
2. In the navigation pane, choose **Issues**. The default view is the **Board**.
3. (Optional) Choose **Active issues** to open the **issues view switcher** dropdown menu to navigate to a different issues view.
4. Choose **Group**.
5. In **Group by**, choose a parameter to group by:
   - If you choose **Assignee** or **Priority**, choose the **Group order**.
   - If you choose **Label**, choose the labels and then choose **Group order**.
6. (Optional) Choose the **Show empty groups** toggle to show or hide groups that have no issues currently assigned to them.
7. The view updates as you make your choices. An issue only appears in the group that matches the configured parameters.

**Filtering issues**

Use filtering to find issues that contain a specified name, priority, label, custom fields, or assignee.

**To filter issues**

1. Navigate to your project.
2. In the navigation pane, choose **Issues**.
3. (Optional) Choose **Active issues** to open the **issues view switcher** dropdown menu to navigate to a different issues view.

   **Note**
   To filter based on a string in the issue name or description, enter the string into the issues search bar.

4. Choose **Filter**, then choose **+ Add filter**.
5. Choose the parameters to filter for. You can choose multiple filters and parameters. You can configure filters to show issues that match every filter or any individual filter by selecting **and** or **or**. The view will update to show the issues that match the filter.

### Creating an issues view

You can create [views](p. 461) to quickly view issues that match a particular set of filters. This can help you save time and quickly view issues you have previously filtered, grouped, or sorted by.

**To create an issues view**

1. In the navigation pane, choose **Issues**.
2. (Optional) Depending on your use case, you may want to create a view from an existing view. To navigate to a different view, choose **Active issues** to open the **issues view switcher** dropdown menu and choose the view.
3. (Optional) Configure filters, grouping, and sorting before you create your view. You can add these while creating a view, but if you do it before, you can preview what is shown in the view before creating it.
4. Open the **issues view switcher** dropdown menu from the header bar. To create a board view where issues are viewed in columns based on status, choose the **+** in the **Board** column. To create a grid view where issues are viewed in a list, choose the **+** in the **Grid** column. You can change the type of view before it is created if you change your mind.
5. In the **Create view** dialog box, enter a **Name** for the view.
6. The **Filters**, **Group issues by**, and **Sort issues by** fields are filled based on the settings of the current view. Update them if necessary.
7. Choose **Create view** to create the view and be switched to it.

### Exporting issues

You can export issues in your current view into a .xlsx file. To export issues, perform the following steps.

**To export issues**

1. Navigate to your project.
2. In the navigation bar choose **Issues**.
3. Choose **Active issues** to open the **issues view switcher** dropdown menu and navigate to the view containing the issues you want to export. Only issues shown in the view will be exported.
4. Choose the ellipses menu and choose **Export to Excel**.
5. The .xlsx file downloads. By default, it is titled the name of the project and the date the export was completed.

### Configuring issues settings

The following topics detail how to configure settings for issues in CodeCatalyst.
Enabling or disabling multiple assignees

Follow these steps to configure the setting for multiple assignees for issues in CodeCatalyst.

To enable or disable multiple assignees

1. In the navigation pane, choose Issues.
2. Choose Active issues to open the issues view switcher dropdown menu and choose Settings.
3. In Assignee in the Basic settings section, toggle the indicator to enable multiple assignees to be assigned to the same issue. An issue can have up to 10 assignees. If you do not enable this option, you will only be able to assign one assignee to an issue.

Configuring issue effort estimation

Follow these steps to configure the setting for effort estimations for issues in CodeCatalyst.

To configure effort estimation for issues

1. In the navigation pane, choose Issues.
2. Choose Active issues to open the issues view switcher dropdown menu and choose Settings.
3. In Estimation in the Basic settings section, choose how the estimation values will be displayed. The types of estimates available are T-shirt sizing, Fibonacci sequencing, or Hide estimates. When the estimation type is updated, no data will be lost and the estimation value of all issues will be converted automatically. The conversion mapping is shown in the following table.

<table>
<thead>
<tr>
<th>T-shirt size</th>
<th>Fibonacci sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS</td>
<td>1</td>
</tr>
<tr>
<td>XS</td>
<td>2</td>
</tr>
<tr>
<td>S</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>5</td>
</tr>
<tr>
<td>L</td>
<td>8</td>
</tr>
<tr>
<td>XL</td>
<td>13</td>
</tr>
</tbody>
</table>

Statuses

You can add custom statuses on your board. Each custom status must belong to one of the following categories: Draft, Not started, Started, or Completed. Status categories are used to help organize
statuses and populate default views. For more information about statuses and status categories, see Status and status categories (p. 461) and for more information about views, see Finding and viewing issues (p. 468).

**To create a status**

1. In the navigation pane, choose **Issues**.
2. Choose **Active issues** to open the issues view switcher dropdown menu and choose **Settings**.
3. In **Statuses**, choose the plus icon next to the category you want the status to be in.
4. Name the status, then choose check mark icon.
   **Note**
   Choose the X icon to cancel adding a status.

   The custom status is now visible on your board and shows as an option when creating an issue.

**To edit a status**

1. In the navigation pane, choose **Issues**.
2. Choose **Active issues** to open the issues view switcher dropdown menu and choose **Settings**.
3. In **Statuses**, choose the edit icon next to the status you want to edit or change.
4. Edit the status, then choose the check mark icon.

   The edited status is now visible on your board.

**To move a status**

1. In the navigation pane, choose **Issues**.
2. Choose the ellipsis icon and choose **Settings**.
3. In **Statuses**, choose a status you want to move.
4. Drag and drop the status where you want it to be.
   **Note**
   You can only move a status within its designated category.

   The statuses are now reordered on your board.

**To deactivate a status**

1. In the navigation pane, choose **Issues**.
2. Choose **Active issues** to open the issues view switcher dropdown menu and choose **Settings**.
3. In **Statuses**, choose a status you want to deactivate.
4. On the status you want to deactivate, choose the toggle on the status. The status is now grayed out.
   **Note**
   The deactivated status appears on the board until all issues are moved out of it. Issues cannot be added to a deactivated status.
5. To reactivate a deactivated status, choose the toggle on the status. The status is no longer grayed out.
   **Note**
   There must be at least one active status in each category. If there is only one status in the category, you cannot deactivate it.
Labels

You can customize labels for issues. This includes editing the label and changing the color. Labels can help you categorize and organize your work.

Creating a label

In CodeCatalyst, you create labels by either adding them when you create a new issue or when you edit an existing issue. For more information, see Creating an issue in CodeCatalyst (p. 461) and Editing and collaborating on issues in CodeCatalyst (p. 462).

Editing a label

Use the following procedure to change the name or color of an existing label.

To edit a label

1. In the navigation pane, choose Issues.
2. Choose Active issues to open the issues view switcher dropdown menu and choose Settings.
3. On the Labels tile is a list of the labels used in the project. Choose the edit icon next to the label that you want to edit. Do one or more of the following:
   a. Edit the name of the label.
   b. To change the color, choose the color wheel. Use the picker to choose a new color.
4. To save the changes you made to the label, choose the check mark icon.
5. The changed label is now visible in your list of available labels. You can also see how many issues are using that label.

   Note
   You can choose the number displayed next to each label to navigate to the All issues page and see all issues that contain that label.

Deleting a label

You cannot currently delete an issues label in CodeCatalyst. If you remove a label from all issues, the label will appear in the Unused labels section in the Labels section of issue settings. Unused labels appear at the end of the list of labels when using filters or adding labels to an issue. You can find an overview of all labels (used and unused) and issues that have them in the issue settings.

Custom fields

You can create custom fields to help organize and view the work for your project. Custom fields are added to the list of available filters in Filter so you can filter issues by custom fields. Custom fields are name and value pairs. You filter by the name of the custom field, and then the value of that custom field.

An issue can have multiple custom fields.

Creating a custom field

In CodeCatalyst, you create custom fields by either adding them when you create an issue or when you edit an existing issue. For more information, see Creating an issue in CodeCatalyst (p. 461) and Editing and collaborating on issues in CodeCatalyst (p. 462).
Deleting a custom field

Do delete a custom field, you must remove the custom field from each issue it is added to. When a custom field is deleted, you will no longer see the custom field in Filter. You can use filters to view all issues with a custom field, and remove them by editing the issues. For more information, see Finding and viewing issues (p. 468) and Editing an issue (p. 462)

Viewing and managing attachments

You can view a table with every attachment added to issues in your project in issue settings. This table includes details of each attachment, including information such as content type, when it was added, the issue it's added to and its status, and the file size.

This table can be used to easily identify large attachments on issues that are completed or archived to remove them to free up space storage.

Important
Attachments to issues are not scanned or analyzed by Amazon CodeCatalyst. Any user could add an attachment to an issue that might potentially contain malicious code or content. Make sure that users are aware of best practices when it comes to managing attachments and guarding against malicious code, content, or viruses.

To view and manage all issue attachments in a project

1. In the navigation pane, choose Issues.
2. Choose the ellipsis icon and choose Settings.
3. Choose the Attachments tab.

Markdown tips and tricks

When you create an issue, you can format your description and comments using Markdown, which is a plaintext markup language. The following list describes the Markdown syntax for common formatting like headings, bullets, and links.

- To add a heading, use hashtags (#), where the number of hashtags corresponds to the level of the heading.

```markdown
# This is a heading 1
## This is a heading 2
### This is a heading 3
```

- To add bullets, use dashes (-).

```markdown
- Item 1
- Item 2
- Item 3
```

- To add numbered lists, use numbers (n.).

```markdown
1. Item 1
2. Item 2
3. Item 3
```

- To add check boxes, use square brackets ([ ]).

```markdown
[ ] To do
```
2. [x] Done
3. [x] Also done

- To add italics or bold, use single or double asterisks (*), respectively.

  *This is italic*
  **This is bold**

- To add inline code or a code block, use single or triple backticks (´), respectively.

  `This is inline code`
  ```
  ```This is a code block```

- To add a link with clickable text, use one of the following options:

  <http://www.example.com>
  [Example website](https://www.example.com)

- To add an image, use the following syntax:

  ![alt text](https://example.com/images/icon.png "on hover text")

## Quotas for issues in CodeCatalyst

The following table describes quotas and limits for issues in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see [Quotas for CodeCatalyst (p. 620)](#).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active issues</td>
<td>Maximum of 1,000 per project.</td>
</tr>
<tr>
<td>Attachment size</td>
<td>Maximum of 500MB per attachment.</td>
</tr>
<tr>
<td>Total number of issues (active and archived)</td>
<td>Maximum of 100,000 per project.</td>
</tr>
<tr>
<td>Saved views</td>
<td>Maximum of 50 saved issue views per project.</td>
</tr>
<tr>
<td>Number of pull requests you can link to an issue</td>
<td>Maximum of 50 pull requests per issue.</td>
</tr>
<tr>
<td>Statuses</td>
<td>Maximum of 50 per project.</td>
</tr>
<tr>
<td>Labels (per project)</td>
<td>Maximum of 200 per project.</td>
</tr>
<tr>
<td>Labels (per issue)</td>
<td>Maximum of 50 per issue.</td>
</tr>
<tr>
<td>Custom fields (per issue)</td>
<td>Maximum of 50 per issue.</td>
</tr>
<tr>
<td>Assignees</td>
<td>Maximum of 10 per issue.</td>
</tr>
<tr>
<td>Comments</td>
<td>Maximum of 1,000 per issue.</td>
</tr>
</tbody>
</table>
Identity, permissions, and access in CodeCatalyst

When you sign in to Amazon CodeCatalyst for the first time, you create an AWS Builder ID. AWS Builder IDs do not exist in AWS Identity and Access Management. The user name that you choose during your first sign-in becomes your unique user ID for your identity.

In CodeCatalyst, you can sign in for the first time in one of two ways:

- As part of creating a space.
- As part of accepting an invitation to a project or space in CodeCatalyst.

The role or roles associated with your identity determine the actions you can perform in CodeCatalyst. Project roles, such as Project administrator and Contributor, are specific to a project, so you can have one role in one project and a different role in another project. If you create a space, CodeCatalyst automatically assigns you the Space administrator role. When users accept invitations to a project, CodeCatalyst adds those identities to the space and assigns them the Limited access role. When you invite users to projects, you choose the role you want them to have in the project, which determines what actions they can and cannot take within the project. Most users working on a project only need the Contributor role to perform their tasks. For more information, see Working with roles in Amazon CodeCatalyst (p. 476).

In addition to a project role, users in a project need a personal access token (PAT) to access source repositories for a project when using Git clients or integrated development environments (IDEs). Project members can use this PAT with third-party applications as an application-specific password associated with their CodeCatalyst identity. For example, when you clone a source repository to a local computer, you must provide a PAT as well as your CodeCatalyst user name.

You can configure access between CodeCatalyst and AWS resources by using a service role (p. 516) to perform actions such as accessing AWS CloudFormation stacks and resources when you deploy actions in workflows. You must configure access between CodeCatalyst and AWS resources for the workflow actions that are included with the project templates to run.

Topics
- Working with roles in Amazon CodeCatalyst (p. 476)
- Managing personal access tokens in Amazon CodeCatalyst (p. 497)
- Multi-factor authentication (MFA) in Amazon CodeCatalyst (p. 500)
- Security in Amazon CodeCatalyst (p. 503)
- Monitoring in Amazon CodeCatalyst (p. 545)
- Quotas for identity, permission, and access in CodeCatalyst (p. 553)
- Troubleshooting (p. 554)

Working with roles in Amazon CodeCatalyst

In Amazon CodeCatalyst, you can assign roles to users at both the project level and the space level. In a project, a role specifies what a user is allowed to do in a project with the resources for that project. Users gain membership in a space when they join a project. You can add or remove users as administrators of a space. The Space administrator role has the broadest permissions of any role in CodeCatalyst. As a best practice, assign users the narrowest permissions necessary to perform their jobs.
You can assign roles to users in the space. You can also assign roles to users in the projects where they are members. Each user can only have one role in a project or space, but users can have different roles in each project and space. For example, a user might have the Project administrator role in one project and the Contributor role in another project.

There are three roles available for a space:

- **Space administrator** (p. 477)
- **Power user** (p. 483)
- **Limited access** (p. 484)

Users who accept an invitation to a project have the **Limited access** role automatically assigned to them in the space that contains the project.

There are four roles available for members in a project:

- **Project administrator** (p. 485)
- **Contributor** (p. 488)
- **Reviewer** (p. 492)
- **Read only** (p. 495)

When you add a user to a project, Amazon CodeCatalyst automatically gives them the **Limited access** role. If you remove a user from all projects, Amazon CodeCatalyst automatically removes the **Limited access** role from that user.

**Topics**

- **Space administrator role** (p. 477)
- **Power user role** (p. 483)
- **Limited access role** (p. 484)
- **Project administrator role** (p. 485)
- **Contributor role** (p. 488)
- **Reviewer role** (p. 492)
- **Read only role** (p. 495)
- **Viewing and changing user roles** (p. 496)

## Space administrator role

The **Space administrator** role is the most powerful role in Amazon CodeCatalyst. Only assign the **Space administrator** role to users who need to administer every aspect of a space, because this role has all permissions in CodeCatalyst. Users with the **Space administrator** role are the only users who can add or remove other users from the **Space administrator** role and delete the space.

When you create a space, CodeCatalyst automatically assigns you the **Space administrator** role. As a best practice, we recommend that you add this role to at least one other user who can act in this role in case the original space creator is unavailable.

### Space administrator role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>Create space</td>
<td>None</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td>View space</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Edit space billing details</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Delete space</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Invite others to the space</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Add other users to the <strong>Space administrator</strong> role</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Remove other users from the <strong>Space administrator</strong> role</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>View logs of activity in the space</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Create projects</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Associate an AWS account connection with the space</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Update an AWS account connection</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Disassociate an AWS account connection from the space</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Delete an AWS account connection and remove it from the space</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>Install extensions</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Update extensions</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Delete extensions</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>View extensions</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View configuration details for installed extensions</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Connect a GitHub account</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Disconnect a GitHub account</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Connect a Jira site</td>
<td>Power user</td>
</tr>
<tr>
<td>Extensions</td>
<td>Disconnect a Jira site</td>
<td>Power user</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Project</td>
<td>View project</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View project activity</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Edit project settings</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete project</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Invite users to a project</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Change roles of users in a project</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Remove users from a project</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View blueprints</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Source repository</td>
<td>Create repositories</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Link repositories</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Unlink repositories</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete repositories</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Edit repository settings</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View repositories</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View repository settings</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Clone repositories</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create branches</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Change default branch</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete branches</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Merge branches</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View branches</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create folders</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete folders</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit folders</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View folders</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create files</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Delete files</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit files</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View files</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create and push commits</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View commits</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create pull requests</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update pull requests</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View pull requests</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Close pull requests</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Approve pull requests</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Comment on pull requests</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Link an issue to a pull request</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Unlink an issue from a pull request</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>Create your own Dev Environment</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Stop your own Dev Environment</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Stop Dev Environments created by other users</td>
<td>Power user, Project administrator</td>
</tr>
<tr>
<td></td>
<td>Resume your own Dev Environment</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View your own Dev Environments</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View Dev Environments created by other users</td>
<td>Power user, Project administrator</td>
</tr>
<tr>
<td></td>
<td>Edit your own Dev Environment</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Edit Dev Environments created by other users</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Delete your own Dev Environment</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td>Delete Dev Environments created by other users</td>
<td>Power user</td>
</tr>
<tr>
<td></td>
<td>Create a devfile for Dev Environments</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit a devfile for Dev Environments</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete a devfile for Dev Environments</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View a devfile for Dev Environments</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Workflows</td>
<td>Create workflow</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update workflow</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View workflow</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Delete workflow</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Start workflow</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Stop workflow</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View workflow runs</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View workflow run results</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View build action logs</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View test action logs</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Track a commit in a workflow</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Manage workflow resources in other accounts</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Create workflow secrets</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update workflow secrets</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete workflow secrets</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View workflow secrets</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create fleets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Update fleets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete fleets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View fleets</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Issues</td>
<td>Create issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Update issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View issues</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Archive an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Update issues created by other users</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View comments on an issue</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create a comment on an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a comment on an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Create a label</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a label</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View labels</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Add a label to an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Remove a label from an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Create a custom status for issues</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a custom status</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View a custom status</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Move a custom status</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Deactivate a custom status</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Add an attachment to an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View an issue attachment</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Remove an attachment from an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Link a pull request to an issue</td>
<td>Project administrator, Contributor</td>
</tr>
</tbody>
</table>
### Power user role

The **Power user** role is the second-most powerful role in Amazon CodeCatalyst spaces, but it has no access to projects in a space. It is designed for users who need to be able to create projects in a space and help manage the users and resources for the space. Assign the **Power user** role to users who are team leaders or managers who need the ability to create projects and manage users in the space as part of their work.

#### Power user role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Invite others to the space</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>View logs of activity in the space</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Create projects</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Associate an AWS account connection with the space</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Update an AWS account connection</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Disasssociate an AWS account connection from the space</td>
<td>None</td>
</tr>
</tbody>
</table>

---

**Amazon CodeCatalyst User Guide**

**Power user role**

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlink a pull request from an issue</td>
<td>Project administrator, Contributor</td>
<td></td>
</tr>
<tr>
<td>Link a Jira project</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Unlink a Jira project</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Unlink a pull request from an issue</td>
<td>Project administrator, Contributor</td>
<td></td>
</tr>
<tr>
<td>Link a Jira project</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Unlink a Jira project</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Notifications</td>
<td>Configure notification channel</td>
<td>None</td>
</tr>
<tr>
<td>Remove notification channel</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Edit notification settings</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>View notification settings</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
<td></td>
</tr>
<tr>
<td>Configure email notifications for your associated email account</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
<td></td>
</tr>
<tr>
<td>Automatically receive notifications about CodeCatalyst incidents</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>Search inside a project</td>
<td>Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Search across the space</td>
<td>Power user, Project administrator, Contributor, Reviewer, Read only</td>
<td></td>
</tr>
</tbody>
</table>
Limited access role

The **Limited access** role is the role most users will have in Amazon CodeCatalyst spaces. It is the role automatically assigned to users when they accept an invitation to a project in a space. It provides the limited permissions they need to work within the space that contains that project. Assign the **Limited access** role to users you invite directly to the space unless their work requires that they manage some aspect of the space.

**Limited access role permissions**

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Power user, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
</tbody>
</table>
### Project administrator role

The **Project administrator** role is the most powerful role in an Amazon CodeCatalyst project. Only assign this role to users who need to administer every aspect of a project, including editing project settings, managing project permissions, and deleting projects.

Project roles do not have any permissions at the space level. Therefore, users with the **Project administrator** role cannot create additional projects. Only users with the **Space administrator** or **Power user** role can create projects.

**Note**
The **Space administrator** role has all permissions in CodeCatalyst and is therefore not listed in this table. **None** indicates that there are no other roles at the project level that have this permission.

### Project administrator role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Power user, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View extensions</td>
<td>Power user, Limited access, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Project</td>
<td>View project</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View project activity</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Edit project settings</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Delete project</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Invite users to a project</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Change roles of users in a project</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Remove users from a project</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>View blueprints</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Source repository</td>
<td>Create repositories</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Link repositories</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Unlink repositories</td>
<td>None</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Delete repositories</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Edit repository settings</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View repositories</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Clone repositories</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create branches</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Change default branch</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Delete branches</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Merge branches</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View branches</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create folders</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete folders</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit folders</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View folders</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create files</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete files</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit files</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View files</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create and push commits</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View commits</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create pull requests</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Update pull requests</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View pull requests</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Close pull requests</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Approve pull requests</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Comment on pull requests</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Link an issue to a pull request</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Unlink an issue from a pull request</td>
<td>Contributor</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>Create your own Dev Environment</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Stop your own Dev Environment</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Resume your own Dev Environment</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View your own Dev Environments</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>Edit your own Dev Environment</strong></td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td><strong>Delete your own Dev Environment</strong></td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td><strong>Create a devfile for Dev Environments</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Edit a devfile for Dev Environments</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Delete a devfile for Dev Environments</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>View a devfile for Dev Environments</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Workflows</td>
<td><strong>Create workflow</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Update workflow</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>View workflow</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>Delete workflow</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Start workflow</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Stop workflow</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>View workflow run results</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>View build action logs</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>View test action logs</strong></td>
<td>Contributor, Reviewer, Read only,</td>
</tr>
<tr>
<td></td>
<td><strong>Track a commit in a workflow</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>Manage workflow resources in other accounts</strong></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Create workflow secrets</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Update workflow secrets</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>Delete workflow secrets</strong></td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td><strong>View workflow secrets</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>Create fleets</strong></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Update fleets</strong></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>Delete fleets</strong></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td><strong>View fleets</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Issues</td>
<td><strong>Create issue</strong></td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td><strong>Update issue</strong></td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td><strong>View issues</strong></td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td><strong>Archive an issue</strong></td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td><strong>Update issues created by other users</strong></td>
<td>Contributor, Reviewer</td>
</tr>
</tbody>
</table>
### Contributor role

The **Contributor** role is intended for the majority of members in an Amazon CodeCatalyst project. Assign this role to users who need to be able to work with code, workflows, issues, and actions in a project.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>View comments on an issue</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create a comment on an issue</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a comment on an issue</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Create a label</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a label</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View labels</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Add a label to an issue</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Remove a label from an issue</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Create a custom status for issues</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Update a custom status</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>View a custom status</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Move a custom status</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Deactivate a custom status</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Add an attachment to an issue</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View an issue attachment</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Remove an attachment from an issue</td>
<td>Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Link a pull request to an issue</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Unlink a pull request from an issue</td>
<td>Contributor</td>
</tr>
<tr>
<td></td>
<td>Link a Jira project</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Unlink a Jira project</td>
<td>None</td>
</tr>
<tr>
<td>Notifications</td>
<td>Edit notification settings</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>View notification settings</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Configure email notifications for your associated email account</td>
<td>Power user, Limited access, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Search</td>
<td>Search inside a project</td>
<td>Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Search across the space</td>
<td>Power user, Contributor, Reviewer, Read only</td>
</tr>
</tbody>
</table>
**Note**
The **Space administrator** role has all permissions in CodeCatalyst and is therefore not listed in this table.

### Contributor role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View extensions</td>
<td>Power user, Limited access, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Project</td>
<td>View project</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View project activity</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View blueprints</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Source repository</td>
<td>Create repositories</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Link repositories</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Edit repository settings</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View repositories</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View repository settings</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Clone repositories</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create branches</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete branches</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Merge branches</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View branches</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create folders</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete folders</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Edit folders</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View folders</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Create files</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Delete files</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Edit files</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>View files</td>
<td>Project administrator, Reviewer, Read only</td>
<td></td>
</tr>
<tr>
<td>Create and push commits</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>View commits</td>
<td>Project administrator, Reviewer, Read only</td>
<td></td>
</tr>
<tr>
<td>Create pull requests</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Update pull requests</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>View pull requests</td>
<td>Project administrator, Reviewer, Read only</td>
<td></td>
</tr>
<tr>
<td>Close pull requests</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Approve pull requests</td>
<td>Project administrator, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Comment on pull requests</td>
<td>Project administrator, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Link an issue to a pull request</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Unlink an issue from a pull request</td>
<td>Project administrator</td>
<td></td>
</tr>
<tr>
<td>Dev Environments</td>
<td>Create your own Dev Environment</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Stop your own Dev Environment</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Resume your own Dev Environment</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View your own Dev Environments</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Edit your own Dev Environment</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Delete your own Dev Environment</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Create a devfile for Dev Environments</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Edit a devfile for Dev Environments</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete a devfile for Dev Environments</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View a devfile for Dev Environments</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Workflows</td>
<td>Create workflow</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Update workflow</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View workflow</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Delete workflow</td>
<td>Project administrator</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Start workflow</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Stop workflow</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View workflow runs</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View workflow run results</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View build action logs</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View test action logs</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Track a commit in a workflow</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create workflow secrets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Update workflow secrets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Delete workflow secrets</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View workflow secrets</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View fleets</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Issues</td>
<td>Create issue</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Update issue</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View issues</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Archive an issue</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Update issues created by other users</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View comments on an issue</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Create a comment on an issue</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Update a comment on an issue</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Create a label</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>Update a label</td>
<td>Project administrator</td>
</tr>
<tr>
<td></td>
<td>View labels</td>
<td>Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>Add a label to an issue</td>
<td>Project administrator, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Remove a label from an issue</td>
<td>Project administrator, Reviewer</td>
</tr>
</tbody>
</table>
## Reviewer role

The **Reviewer** role is intended for users who need to be able to interact with resources in a project, such as pull requests and issues, but not create and merge code, create workflows, or start or stop workflow runs in an Amazon CodeCatalyst project. Assign this role to users who need to be able to approve and comment on pull requests, create, update, resolve, and comment on issues, and view code and workflows in a project.

**Note**

The **Space administrator** role has all permissions in CodeCatalyst and is therefore not listed in this table.

### Reviewer role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View extensions</td>
<td>Power user, Limited access, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Project</td>
<td>View project</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View blueprints</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td>Source repository</td>
<td>View repositories</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Clone repositories</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View branches</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View folders</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View files</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View pull requests</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Approve pull requests</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Comment on pull requests</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>Create your own Dev Environment</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Stop your own Dev Environment</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Resume your own Dev Environment</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View your own Dev Environments</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Edit your own Dev Environment</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Delete your own Dev Environment</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View a devfile for Dev Environments</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td>Workflows</td>
<td>View workflow</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View workflow run results</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View build action logs</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td>Resource type</td>
<td>Permission</td>
<td>Other roles with this permission</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>View test action logs</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Track a commit in a workflow</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View workflow secrets</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View fleets</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td>Issues</td>
<td>Create issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View issues</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Archive an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Update issues created by other users</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View comments on an issue</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>View labels</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Add a label to an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Remove a label from an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View a custom status</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Move a custom status</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Deactivate a custom status</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>Add an attachment to an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td></td>
<td>View an issue attachment</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Remove an attachment from an issue</td>
<td>Project administrator, Contributor</td>
</tr>
<tr>
<td>Notifications</td>
<td>View notification settings</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Configure email notifications for your associated email account</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer, Read only</td>
</tr>
<tr>
<td>Search</td>
<td>Search inside a project</td>
<td>Project administrator, Contributor, Read only</td>
</tr>
<tr>
<td></td>
<td>Search across the space</td>
<td>Power user, Project administrator, Contributor, Read only</td>
</tr>
</tbody>
</table>
Read only role

The Read only role is intended for users who need to view the resources and status of resources but not interact with them or contribute directly to the project. Users with this role cannot create resources in CodeCatalyst, but they can view them and copy them, such as cloning repositories and downloading attachments to issues to a local computer. Assign this role to users who need to view resources and the state of the project, but not interact directly with it.

Note
The Space administrator role has all permissions in CodeCatalyst and is therefore not listed in this table.

Read only role permissions

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>View space</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View AWS account connections</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td></td>
<td>View incidents for CodeCatalyst</td>
<td>Power user, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Extensions</td>
<td>View extensions</td>
<td>Power user, Limited access, Project administrator, Reviewer, Read only</td>
</tr>
<tr>
<td>Project</td>
<td>View project</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View project activity</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View blueprints</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Source repository</td>
<td>View repositories</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>Clone repositories</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View branches</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View folders</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View files</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td></td>
<td>View pull requests</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Dev Environments</td>
<td>View a devfile for Dev Environments</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Workflows</td>
<td>View workflow</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
</tbody>
</table>
Viewing and changing user roles

You can view the role assigned to a user. This helps you understand what actions they can take in a project. You can also change their role if they need additional permissions.

**To view the role of a user in a project**

1. Navigate to the project where you want to view the roles associated with each project member.

---

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Permission</th>
<th>Other roles with this permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>View workflow runs</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View workflow run results</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View build action logs</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View test action logs</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Track a commit in a workflow</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View workflow secrets</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View fleets</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Issues</td>
<td>View issues</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>View comments on an issue</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View labels</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View a custom status</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>View an issue attachment</td>
<td>Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Notifications</td>
<td>View notification settings</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Configure email notifications for your associated email account</td>
<td>Power user, Limited access, Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>Search inside a project</td>
<td>Project administrator, Contributor, Reviewer</td>
</tr>
<tr>
<td>Search across the space</td>
<td>Power user, Project administrator, Contributor, Reviewer</td>
<td></td>
</tr>
</tbody>
</table>
Managing personal access tokens

Tip
You can choose which project to view in the top navigation bar.

1. In the navigation pane, choose **Project settings**.
2. On the **Members** tab, the role for each project member is displayed in **Role**.

### To change users' roles in a project

1. Navigate to the project where you want to change the roles associated with project members.

   **Tip**
   You can choose which project to view in the top navigation bar.

2. In the navigation pane, choose **Project settings**.
3. On the **Members** tab, in **Project members**, choose the users whose roles you want to change.
   Choose **Action**, and then choose **Edit role**.
4. In **Role**, choose the project role, and then choose **Confirm**.

### Viewing and changing roles in the space

All users who accept invitations to a project in CodeCatalyst become members of the project's space. You can view the list of space members. You can change users' roles from **Limited access** to **Space administrator** to better manage your space and its resources. The **Space administrator** role is the only role that allows users to create projects in CodeCatalyst.

**Warning**
The **Space administrator** role is the most powerful role in CodeCatalyst. Users with this role can perform any action in CodeCatalyst, including deleting the space. Only assign this role to users who require this level of access to your space. For more information, see [Space administrator role](p. 477).

### To change a user's role in the space

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to the space.

   **Tip**
   If you belong to more than one space, you can choose which space to view in the top navigation bar.
3. Choose the **Members** tab.
4. Choose the user whose role you want to change, and then choose **Change role**.
5. In **Change role**, choose the role you want to assign, and then choose **Confirm**.

---

Managing personal access tokens in Amazon CodeCatalyst

To access some CodeCatalyst resources, such as source repositories, on a local computer with a Git client or integrated development environment (IDE), you must enter an application-specific password. You can create a personal access token (PAT) to use for this purpose. PATs you create are associated with your user identity across all spaces and projects in CodeCatalyst. You can create more than one PAT for your CodeCatalyst identity.

You can view the names and expiration dates of the PATs you have created, and you can delete those you no longer need. You can only copy the PAT secret at the time you create it.
Note
By default, PATs expire in 1 year.

Creating PATs

PATs are associated with your user identity in CodeCatalyst. You can only copy a PAT secret at the time you create it.

Creating PATs (console)

You can use the console to create PATs in CodeCatalyst.

To create a personal access token (console)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the top menu bar, choose your profile badge, and then choose My settings. The CodeCatalyst My settings page opens.
   Tip
   You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.
3. Under Personal access tokens, choose Create.
   The Create PAT page displays.
4. In PAT name, enter a descriptive name for your PAT.
5. In Expiration date, keep the default date or choose the calendar icon to select a custom date. The expiration date defaults to 1 year from the current date.
6. Choose Create.
   Tip
   You can also create this token when you choose Clone repository for a source repository.
7. To copy the PAT secret, choose Copy. Store the PAT secret where you will be able to retrieve it.
   Important
   The PAT secret only displays once. You cannot retrieve it after you close the window. If you did not save the PAT secret in a secure location, you can create another one.

Creating PATs (CLI)

You can use the CLI to create PATs in CodeCatalyst.

To create a personal access token (AWS CLI)

1. At the terminal or command line, run the create-access-token command as follows.

   ```bash
   aws codecatalyst create-access-token
   ```

   If successful, the command returns information about the created PAT like the following example.

   ```json
   {
     "secret": "value",
     "name": "marymajor-22222EXAMPLE",
     "expiresTime": "2024-02-04T01:56:04.402000+00:00"
   }
   ```
You can only view the PAT secret once—when you create the PAT. If you’ve misplaced a PAT secret or you’re concerned that it’s not stored securely, you can create another one.

You can view the PATs associated with your user account by using the AWS CLI. You can only view information about the PAT, and not the value of the PAT secret itself.

**Note**
Make sure that you’re using a recent version of the AWS CLI to work with CodeCatalyst. Earlier versions might not contain the CodeCatalyst commands. You must configure your AWS CLI profile before you can use it with CodeCatalyst. For more information, see [Setting up to use the AWS CLI with CodeCatalyst](p. 15).

**Viewing PATs**

You can view PATs in CodeCatalyst. The list shows all of the PATs that you have associated with your user identity. Your PAT is associated with your user profile across all spaces and projects in CodeCatalyst. Expired PATs do not display because they're deleted after they expire.

**Viewing PATs (console)**

You can use the console to view PATs associated with your user identity in CodeCatalyst.

To view your personal access tokens (console)

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. In the top menu bar, choose your profile badge, and then choose **My settings**. The CodeCatalyst **My settings** page opens.
   
   **Tip**
   You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.
3. Under **Personal access tokens**, view the names and expiration dates of your current PATs.

**Viewing PATs (CLI)**

You can use the CLI to view PATs associated with your user identity in CodeCatalyst.

To view your personal access tokens (AWS CLI)

- At the terminal or command line, run the **list-access-tokens** command as follows.

```bash
aws codecatalyst list-access-tokens
```

If successful, the command returns information about the PATs associated with your user account like the following example.

```json
{
   "items": [
      {
         "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
         "name": "marymajor-22222EXAMPLE",
         "expiresTime": "2024-02-04T01:56:04.402000+00:00"
      },
      {
         "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbb",
```
Deleting PATs

You can delete PATs associated with your user identity in CodeCatalyst.

Deleting PATs (console)

You can use the console to delete PATs in CodeCatalyst.

To delete a personal access token (console)

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. In the top menu bar, choose your profile badge, and then choose My settings. The CodeCatalyst My settings page opens.
   Tip
   You can also find your user profile by going to the members page for a project or space and choosing your name from the members list.
3. Under Personal access tokens, choose the selector next to the PAT you want to delete, and then choose Delete.

   On the Delete PAT: <name>? page, to confirm deletion, type delete in the text field. Choose Delete.

Deleting PATs (CLI)

You can delete a PAT associated with your user identity by using the AWS CLI. To do this, you must supply the ID for the PAT, which you can view by using the delete-access-token command.

Note
Make sure that you're using a recent version of the AWS CLI to work with CodeCatalyst. Earlier versions might not contain the CodeCatalyst commands. For more information about using the AWS CLI with CodeCatalyst, see Setting up to use the AWS CLI with CodeCatalyst (p. 15).

To delete a personal access token (AWS CLI)

- At the terminal or command line, run the delete-access-token command, providing the ID for the PAT you want to delete. For example, run the following command to delete a PAT with an ID of 123EXAMPLE.

  ```bash
  aws codecatalyst delete-access-token --id a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb
  ```

  If successful, this command returns no response.

Multi-factor authentication (MFA) in Amazon CodeCatalyst

Whether you created your AWS Builder ID profile for personal use or professional use, we encourage configuring multi-factor authentication (MFA) as another layer of security. We especially recommend
configuring MFA if you're a member of a space and collaborate with others on projects. Because more than one person can have access to a project, more opportunities exist for security breaches.

When you enable MFA, you must sign in to Amazon CodeCatalyst with your email and password. This portion of signing in is the first factor, where you use something that you know. You then sign in with either a code or security key. This is the second factor, which is something that you have. The second factor could be an authentication code that is generated either by your mobile device or by tapping or pressing a security key connected to your computer. Taken together, these multiple factors provide increased security by preventing unauthorized access.

How to register a device for use with multi-factor authentication

Use the following procedure on My profile > Multi-factor authentication to register your new device for multi-factor authentication (MFA).

**Note**
We recommend that you first download the appropriate authenticator app onto your device before starting the steps in this procedure. For a list of apps that you can use for MFA devices, see Authenticator applications (p. 502).

To register your device for use with MFA

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose User profile. The CodeCatalyst Profile page opens.
3. On the profile page, choose Manage profile and security. The AWS Builder ID profile page opens.
4. On the left side of the page, choose Security.
5. On the Multi-factor authentication page, choose Register device.
6. On the Register MFA device page, choose one of the following MFA device types, and follow the instructions:
   - **Security key or Built-in authenticator**
     1. On the Register your user's security key page, follow the instructions given to you by your browser or platform.
     **Note**
     This experience varies based on your operating system and browser, so follow the instructions displayed by your browser or platform. After your device has been successfully registered, you will be given the option to associate a friendly display name to your newly enrolled device. If you want to change this, choose Rename, enter the new name, and then choose Save.
   - **Authenticator app**
     1. On the Set up the authenticator app page, you might notice configuration information for the new MFA device, including a QR code graphic. The graphic is a representation of the secret key that is available for manual entry on devices that do not support QR codes.
     2. Using the physical MFA device, do the following:
        a. Open a compatible MFA authenticator app. For a list of tested apps that you can use with MFA devices, see Tested authenticator apps (p. 502). If the MFA app supports multiple devices, choose the option to create a new MFA device.
        b. Determine whether the MFA app supports QR codes, and then do one of the following on the Set up the authenticator app page:
Authenticator applications

Authenticator apps are one-time password (OTP)–based third party-authenticators. Users can use an authenticator application installed on their mobile device or tablet as an authorized MFA device. The third-party authenticator application must be compliant with RFC 6238, which is a standards-based TOTP (time-based one-time password) algorithm capable of generating six-digit authentication codes.

When prompted for MFA, users must enter a valid code from their authenticator app within the input box presented. Each MFA device assigned to a user must be unique. Two authenticator apps can be registered for any given user.

Tested authenticator apps

Although any TOTP-compliant application will work with IAM Identity Center MFA, the following table lists well-known third-party authenticator apps to choose from.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Tested authenticator app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Authy, Duo Mobile, LastPass Authenticator, Microsoft Authenticator, Google Authenticator</td>
</tr>
<tr>
<td>iOS</td>
<td>Authy, Duo Mobile, LastPass Authenticator, Microsoft Authenticator, Google Authenticator</td>
</tr>
</tbody>
</table>

Changing your MFA devices

After you register an MFA device, you can change its name or delete it. We recommend always having at least one MFA device enabled for an extra layer of security. You can have up to five devices registered. To find out how to add more, see How to register a device for use with multi-factor authentication (p. 501).
Renaming an MFA device

To rename your MFA device

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose User profile. The CodeCatalyst Profile page opens.
3. On the profile page, choose Manage profile and security. The AWS Builder ID profile page opens.
4. Choose Multi-factor authentication on the left side of the page. You'll see that Rename is grayed out when you arrive at the page.
5. Select the MFA device that you want to change. Choose Rename. Then a modal pops up.
6. In the prompt that opens, enter the new name in MFA device name, and then choose Rename. The renamed device appears under Multi-factor authentication devices (MFA).

Deleting an MFA device

To delete an MFA device

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. At the top right, choose the arrow next to the icon with your first initial, and then choose User profile. The CodeCatalyst Profile page opens.
3. On the profile page, choose Manage profile and security. The AWS Builder ID profile page opens.
4. Choose Multi-factor authentication on the left side of the page. You'll see that Delete is grayed out when you arrive at the page.
5. Select the MFA device that you want to change. Choose Delete. A modal appears that says Delete MFA device? Follow the instructions to delete your device.
6. Choose Delete. The deleted device no longer appears under Multi-factor authentication devices (MFA).

Security in Amazon CodeCatalyst

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

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 CodeCatalyst, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS services 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 Amazon CodeCatalyst. It shows you how to configure CodeCatalyst to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your CodeCatalyst resources.
Data protection in Amazon CodeCatalyst

Security and Compliance is a shared responsibility between Amazon CodeCatalyst and the customer, just as the AWS shared responsibility model applies to your use of AWS resources used in a workflow. As described in this model, CodeCatalyst is responsible for protecting the global infrastructure for the service. You are responsible for maintaining control over your content that is hosted on this infrastructure. This shared responsibility model applies to data protection in CodeCatalyst.

For data protection purposes, we recommend that you protect your account credentials, and that you set up multi-factor authentication when signing in. For more information, see Multi-factor authentication (MFA) in Amazon CodeCatalyst (p. 500).

Do not enter confidential or sensitive information, such as your customers' email addresses, in tags or free-form fields such as a Name field. This includes resource names and any other identifiers you enter in CodeCatalyst in addition to any connected AWS accounts. For example, do not enter confidential or sensitive information as part of space, project, or deployment fleet names. Any data that you enter in tags, names, or free-form fields used for names might be used for billing or diagnostic logs or could be included in URL paths. This applies to using the console, API, AWS CLI, the CodeCatalyst Action Development Kit, or any AWS SDKs.

If you provide a URL to an external server, we strongly recommend that you do not include any security credentials information in the URL to validate your request to that server.

CodeCatalyst source repositories are automatically encrypted at rest. No customer action is required. CodeCatalyst also encrypts repository data in transit using the HTTPS protocol.

CodeCatalyst supports MFA. For more information, see Multi-factor authentication (MFA) in Amazon CodeCatalyst (p. 500).

Data encryption

CodeCatalyst securely stores and transfers data within the service. All data is encrypted in transit and at rest. Any data created or stored by the service, including any metadata for the service, is stored natively in the service and encrypted.

Note
While information about issues is stored securely within the service, information about open issues is also stored in the local cache of the browser where you viewed issue boards, backlogs, and individual issues. For optimal security, be sure to clear your browser cache to remove this information.

If you use resources linked to CodeCatalyst, such as an account connection to an AWS account or a linked repository in GitHub, data in transit from CodeCatalyst to that linked resource is encrypted, but the data handling in that linked resource is managed by that linked service. For more information,
see the documentation for the linked service and Best practices for workflow actions in Amazon CodeCatalyst (p. 543).

Key management

CodeCatalyst does not support key management.

Inter-network traffic privacy

When you create a space in CodeCatalyst, you choose the AWS Region where the data and resources will be stored for that space. Project data and metadata never leaves that AWS Region. However, to support navigation within CodeCatalyst, a limited set of space, project, and user metadata is replicated across all AWS Regions in the partition. It will not be replicated to AWS Regions outside of that partition. For example, if you choose US West (Oregon) as the AWS Region when you create your space, your data will not be replicated to Regions in China Regions or AWS GovCloud (US). For more information, see Managing AWS Regions, AWS Global Infrastructure, and AWS service endpoints.

Data replicated across AWS Regions inside a partition includes:

- An encrypted hash value that represents the name of the space in order to ensure the uniqueness of space names. This value is not human-readable and does not expose the actual names of spaces
- The unique ID of the space
- Metadata for the space that assists in the navigation across spaces
- The AWS Region where the space is located
- The unique IDs of all projects in the space
- The role ID that indicates a user's role in a space or project
- When signing up for CodeCatalyst, data and metadata about the signup process, including:
  - The unique ID of the AWS Builder ID
  - The display name for the user in their AWS Builder ID
  - The alias of the user in their AWS Builder ID
  - The email address used when the user signed up for their AWS Builder ID
  - The progress of the sign up process
  - If creating a space as part of the sign up process, the AWS account ID that is used as the billing account for the space

Space names are unique across CodeCatalyst. Be sure not to include sensitive data in the name of the space.

When working with linked resources and connected accounts such as a connection to an AWS account or a GitHub repository, we recommend configuring your source and destination locations with the highest level of security that each one supports. CodeCatalyst secures the connection between AWS accounts, AWS Regions, and Availability Zones by using Transport Layer Security (TLS) 1.2.

Identity and Access Management and Amazon CodeCatalyst

In Amazon CodeCatalyst, you create and use an AWS Builder ID in order to sign in and access your spaces and projects. An AWS Builder ID is not an identity in AWS Identity and Access Management (IAM) and does not exist in an AWS account. However, CodeCatalyst does integrate with IAM when verifying a space for billing purposes, and when connected to an AWS account to create and use resources in that AWS account.
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 resources. IAM is an AWS service that you can use with no additional charge.

When you create a space in Amazon CodeCatalyst, you must connect an AWS account as the billing account for your space. You must have administrator permissions in the AWS account to verify the CodeCatalyst space, or have the permission. You also have the option to add an IAM role for your space that CodeCatalyst can use to create and access resources in that connected AWS account. This is called a service role. You can choose to create connections to more than one AWS account and create service roles for CodeCatalyst in each of those accounts.

Note
Billing for CodeCatalyst takes place in the AWS account designated as the billing account. However, if you create a CodeCatalyst service role in that AWS account or in any other connected AWS account, resources created and used by the CodeCatalyst service role will be billed in that connected AWS account. For more information, see Billing (p. 608).

Topics

- Identity-based policies in IAM (p. 506)
- Policy actions in IAM (p. 506)
- Policy resources in IAM (p. 507)
- Policy condition keys in IAM (p. 507)
- Identity-based policy examples for CodeCatalyst connections (p. 507)
- Using tags to control access to account connection resources (p. 509)
- CodeCatalyst permissions reference (p. 510)
- AWS managed policies for Amazon CodeCatalyst (p. 512)
- IAM roles for Amazon CodeCatalyst access to AWS resources (p. 515)

Identity-based policies in IAM

Identity-based policies are JSON permissions policy documents that you can attach to an identity. That identity could be a user, a group of users, or a 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 CodeCatalyst

To view examples of CodeCatalyst identity-based policies, see Identity-based policy examples for CodeCatalyst connections (p. 507).

Policy actions in IAM

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform which 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.

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

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

Policy resources in IAM

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform which 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": "*"
```

Policy condition keys in IAM

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform which 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 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.

Identity-based policy examples for CodeCatalyst connections

In CodeCatalyst, AWS accounts are required to manage billing for a space and to access resources in project workflows. An account connection is used to authorize adding AWS accounts to a space. Identity-based polices are used in the connected AWS accounts.

By default, users and roles don't have permission to create or modify CodeCatalyst resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies for users that require them.
The following example IAM policies grant permissions for actions related to account connections. Use them to limit access for connecting accounts to CodeCatalyst.

**Example 1: Allow a user to accept connection requests in a single AWS Region**

The following permissions policy only allows users to view and accept requests for connections between CodeCatalyst and AWS accounts. In addition, the policy uses a condition to only allow the actions in the us-west-2 Region and not from other AWS Regions. To view and approve the request, the user signs in to the AWS Management Console with the same account as that specified in the request.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:GetPendingConnection"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-west-2"
        }
      }
    }
  ]
}
```

**Example 2: Allow managing connections in the console for a single AWS Region**

The following permissions policy allows users to manage connections between CodeCatalyst and AWS accounts in a single Region. The policy uses a condition to only allow the actions in the us-west-2 Region and not from other AWS Regions. After you create a connection, you can create the `CodeCatalystWorkflowDevelopmentRole-spaceName` role by choosing the option in the AWS Management Console. In the example policy, the condition for the `iam:PassRole` action includes the service principals for CodeCatalyst. Only roles with that access will be created in the AWS Management Console.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-west-2"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```
Example 3: Deny managing connections

The following permissions policy denies users any ability to manage connections between CodeCatalyst and AWS accounts.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": ["codecatalyst:*"],
      "Resource": "*"
    }
  ]
}
```

Using tags to control access to account connection resources

Tags can be attached to the resource or passed in the request to services that support tagging. Resources in policies can have tags, and some actions in policies can include tags. Tagging condition keys include the `aws:RequestTag` and `aws:ResourceTag` condition keys. When you create an IAM policy, you can use tag condition keys to control the following:

- Which users can perform actions on a connection resource, based on tags that it already has.
- Which tags can be passed in an action's request.
- Whether specific tag keys can be used in a request.

The following examples demonstrate how to specify tag conditions in policies for CodeCatalyst account connections users. For more information about condition keys, see [Policy condition keys in IAM](p. 507).

Example 1: Allow actions based on tags in the request

The following policy grants users permission to approve account connections.

To do that, it allows the `AcceptConnection` and `TagResource` actions if the request specifies a tag named `Project` with the value `ProjectA`. (The `aws:RequestTag` condition key is used to
control which tags can be passed in an IAM request.) The `aws:TagKeys` condition ensures tag key case sensitivity.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

**Example 2: Allow actions based on resource tags**

The following policy grants users permission to perform actions on, and get information about, account connection resources.

To do that, it allows specific actions if the connection has a tag named `Project` with the value `ProjectA` (The `aws:ResourceTag` condition key is used to control which tags can be passed in an IAM request.)

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:GetConnection",
        "codecatalyst:DeleteConnection",
        "codecatalyst:AssociateIamRoleToConnection",
        "codecatalyst:DisassociateIamRoleFromConnection",
        "codecatalyst:ListIamRolesForConnection",
        "codecatalyst:PutBillingAuthorization"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "ProjectA"
        }
      }
    }
  ]
}
```

**CodeCatalyst permissions reference**

This section provides a permissions reference for actions used with the account connection resource for AWS accounts that are connected to CodeCatalyst. The following section describes permissions-only actions that are related to connecting accounts.
## Required permissions for account connections

The following permissions are required for working with account connections.

<table>
<thead>
<tr>
<th>CodeCatalyst permissions for account connections</th>
<th>Required permissions</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptConnection</td>
<td>Required to accept a request to connect this account to a CodeCatalyst space. This is an IAM policy permission only, not an API action.</td>
<td>Supports only a wildcard (*) in the policy Resource element.</td>
</tr>
<tr>
<td>AssociateIamRoleToConnection</td>
<td>Required to associate an IAM role to an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
<tr>
<td>DeleteConnection</td>
<td>Required to delete an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
<tr>
<td>DisassociateIamRoleFromConnection</td>
<td>Required to disassociate an IAM role from an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
<tr>
<td>GetBillingAuthorization</td>
<td>Required to describe the billing authorization for an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
<tr>
<td>GetConnection</td>
<td>Required to get an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
<tr>
<td>GetPendingConnection</td>
<td>Required to get a pending request to connect this account to a CodeCatalyst space. This is an IAM policy permission only, not an API action.</td>
<td>Supports only a wildcard (*) in the policy Resource element.</td>
</tr>
<tr>
<td>ListConnections</td>
<td>Required to list account connections that are not pending. This is an IAM policy permission only, not an API action.</td>
<td>Supports only a wildcard (*) in the policy Resource element.</td>
</tr>
<tr>
<td>ListIamRolesForConnection</td>
<td>Required to list IAM roles associated with an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_ID:connections/region:connection_ID</td>
</tr>
</tbody>
</table>
### CodeCatalyst permissions for account connections

<table>
<thead>
<tr>
<th>CodeCatalyst permissions for account connections</th>
<th>Required permissions</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListTagsForResource</td>
<td>Required to list tags associated with an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_connections/connections/connection_ID</td>
</tr>
<tr>
<td>PutBillingAuthorization</td>
<td>Required to create or update the billing authorization for an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_connections/connections/connection_ID</td>
</tr>
<tr>
<td>RejectConnection</td>
<td>Required to reject a request to connect this account to a CodeCatalyst space. This is an IAM policy permission only, not an API action.</td>
<td>Supports only a wildcard (*) in the policy Resource element.</td>
</tr>
<tr>
<td>TagResource</td>
<td>Required to create or edit tags associated with an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_connections/connections/connection_ID</td>
</tr>
<tr>
<td>UntagResource</td>
<td>Required to remove tags associated with an account connection. This is an IAM policy permission only, not an API action.</td>
<td>arn:aws:codecatalyst:region:account_connections/connections/connection_ID</td>
</tr>
</tbody>
</table>

### AWS managed policies for Amazon CodeCatalyst

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the IAM User Guide.

### AWS managed policy: AmazonCodeCatalystSupportAccess
This is a policy that grants permissions for all space administrators and space members to utilize the Business or Enterprise premium support plan associated with the space billing account. These permissions allow space administrators and members to utilize the premium support plan for the resources they have permissions to within CodeCatalyst permissions policies.

Permissions details
This policy includes the following permissions.

- support – Grants permissions to allow users to search for, create, and resolve AWS Support cases. Also grants permissions to describe communications, severity levels, attachments, and related support case details.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "support:DescribeAttachment",
        "support:DescribeCaseAttributes",
        "support:DescribeCases",
        "support:DescribeCommunications",
        "support:DescribeIssueTypes",
        "support:DescribeServices",
        "support:DescribeSeverityLevels",
        "support:DescribeSupportLevel",
        "support:SearchForCases",
        "support:AddAttachmentsToSet",
        "support:AddCommunicationToCase",
        "support:CreateCase",
        "support:InitiateCallForCase",
        "support:InitiateChatForCase",
        "support:PutCaseAttributes",
        "support:RateCaseCommunication",
        "support:ResolveCase"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AmazonCodeCatalystFullAccess

This is a policy that grants permissions to manage your CodeCatalyst space and connected accounts in the Amazon CodeCatalyst Spaces page in the AWS Management Console. This application is used to configure AWS accounts that are connected to your space in CodeCatalyst.

Permissions details
This policy includes the following permissions.
Codecatalyst – Grants full permissions to the Amazon CodeCatalyst Spaces page in the AWS Management Console.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeCatalystResourceAccess",
      "Effect": "Allow",
      "Action": [
        "codecatalyst:*",
        "iam:ListRoles"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCatalystAssociateIAMRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "codecatalyst.amazonaws.com",
            "codecatalyst-runner.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS managed policy: AmazonCodeCatalystReadOnlyAccess

This is a policy that grants permissions to view and list information for spaces and connected accounts in the Amazon CodeCatalyst Spaces page in the AWS Management Console. This application is used to configure AWS accounts that are connected to your space in CodeCatalyst.

Permissions details

This policy includes the following permissions.

- codecatalyst – Grants read-only permissions to the Amazon CodeCatalyst Spaces page in the AWS Management Console.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:*",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```
CodeCatalyst updates to AWS managed policies

View details about updates to AWS managed policies for CodeCatalyst since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CodeCatalyst Document history (p. 621) page.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmazonCodeCatalystSupportAccess</td>
<td>CodeCatalyst added the policy.</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>– New policy</td>
<td>Grants permissions to allow CodeCatalyst users to search for, create, and resolve support cases, as well as viewing related communications and details.</td>
<td></td>
</tr>
<tr>
<td>AmazonCodeCatalystFullAccess</td>
<td>CodeCatalyst added the policy.</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>– New policy</td>
<td>Grants full access to CodeCatalyst.</td>
<td></td>
</tr>
<tr>
<td>AmazonCodeCatalystReadOnlyAccess</td>
<td>CodeCatalyst added the policy.</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>– New policy</td>
<td>Grants read-only access to CodeCatalyst.</td>
<td></td>
</tr>
<tr>
<td>CodeCatalyst started tracking changes</td>
<td>CodeCatalyst started tracking changes for its AWS managed policies.</td>
<td>April 20, 2023</td>
</tr>
</tbody>
</table>

IAM roles for Amazon CodeCatalyst access to AWS resources

CodeCatalyst can access AWS resources by connecting your AWS account to a CodeCatalyst space. You can then create the following service roles and associate them when you connect your account.

For more information about the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the IAM User Guide.

- To access resources in an AWS account for your CodeCatalyst projects and workflows, you must first grant permission for CodeCatalyst to access those resources on your behalf. To do so, you must create a service role in a connected AWS account that CodeCatalyst can assume on behalf of users and projects in the space. You can either choose to create and use the CodeCatalystWorkflowDevelopmentRole-spaceName service role, or you can create customized service roles and configure these IAM policies and roles manually. As a best practice, assign these roles the least amount of permissions necessary.
Note
For customized service roles, the CodeCatalyst service principal is required. For more information about the CodeCatalyst service principal and trust model, see CodeCatalyst trust model (p. 543).

- To manage support for a space through the connected AWS account, you can choose to create and use the AWSRoleForCodeCatalystSupport service role that allows CodeCatalyst users to access support. For more information about support for a CodeCatalyst space, see AWS Support for Amazon CodeCatalyst (p. 614).

Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role

You can add an IAM role for your space that CodeCatalyst can use to create and access resources in a connected AWS account. This is called a service role. The simplest way to create a service role is to add one when you create the space and to choose the CodeCatalystWorkflowDevelopmentRole-spaceName option for that role. This not only creates the service role with the AdministratorAccess attached, but it also creates the trust policy that allows CodeCatalyst to assume the role on behalf of users in projects in the space. The service role is scoped to the space, not to individual projects. To create this role, see Creating the CodeCatalystWorkflowDevelopmentRole-spaceName role for your account and space (p. 517). You can only create one role for each space in each account.

Note
This role is only recommended for use with development accounts and uses the AdministratorAccess AWS managed policy, giving it full access to create new policies and resources in this AWS account.

The policy attached to the CodeCatalystWorkflowDevelopmentRole-spaceName role is designed to work with projects created with blueprints in the space. It allows users in those projects to develop, build, test, and deploy code using resources in the connected AWS account. For more information, see Creating a role for an AWS service.

The policy attached to the CodeCatalystWorkflowDevelopmentRole-spaceName role is the AdministratorAccess managed policy in AWS. This is a policy that grants full access to all AWS actions and resources. To view the JSON policy document in the IAM console, see AdministratorAccess.

The following trust policy allows CodeCatalyst to assume the CodeCatalystWorkflowDevelopmentRole-spaceName role. For more information about the CodeCatalyst trust model, see CodeCatalyst trust model (p. 543).

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "codecatalyst-runner.amazonaws.com",
        "codecatalyst.amazonaws.com"
      ],
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:codecatalyst::space/spaceId/project/"
      }
    }
  }
]
```
Creating the **CodeCatalystWorkflowDevelopmentRole-spaceName** role for your account and space

Follow these steps to create the CodeCatalystWorkflowDevelopmentRole-spaceName role that will be used for workflows in your space. For each account that you want to have IAM roles for use in projects, to your space, you must add a role such as the developer role.

Before you begin, you must have administrative privileges for your AWS account or be able to work with your administrator. For more information about how AWS accounts and IAM roles are used in CodeCatalyst, see [Managing AWS accounts](p. 36).

**To create and add the CodeCatalyst CodeCatalystWorkflowDevelopmentRole-spaceName**

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.
2. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
3. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **AWS accounts**.
4. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.
5. Choose **Manage roles from AWS Management Console**.
   The **Add IAM role to Amazon CodeCatalyst space** page opens in the AWS Management Console. This is the [Amazon CodeCatalyst Spaces](https://aws.amazon.com/catalyst/) page. You might need to log in to access the page.
6. Choose **Create CodeCatalyst development administrator role in IAM**. This option creates a service role that contains the permissions policy and trust policy for the development role. The role will have a name CodeCatalystWorkflowDevelopmentRole-spaceName. For more information about the role and role policy, see [Understanding the CodeCatalystWorkflowDevelopmentRole-spaceName service role](p. 516).
   **Note**
   This role is only recommended for use with developer accounts and uses the AdministratorAccess AWS managed policy, giving it full access to create new policies and resources in this AWS account.
7. Choose **Create development role**.
8. On the connection page, under **IAM roles available to CodeCatalyst**, view the CodeCatalystWorkflowDevelopmentRole-spaceName role in the list of IAM roles added to your account.
9. To return to your space, choose **Go to Amazon CodeCatalyst**.

**Understanding the AWSRoleForCodeCatalystSupport service role**

You can add an IAM role for your space that CodeCatalyst users in a space can use to create and access support cases. This is called a **service role** for support. The simplest way to create a service role for support is to add one when you create the space and to choose the AWSRoleForCodeCatalystSupport option for that role. This not only creates the policy and the role, but it also creates the trust policy that allows CodeCatalyst to assume the role on behalf of users in projects in the space. The service role is scoped to the space, not to individual projects. To create this role, see [Creating the AWSRoleForCodeCatalystSupport role for your account and space](p. 518).

The policy attached to the AWSRoleForCodeCatalystSupport role is managed policy that provides access to support permissions. For more information, see [AWS managed policy: AmazonCodeCatalystSupportAccess](p. 512).
The trust role for the policy allows CodeCatalyst to assume the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": ["codecatalyst.amazonaws.com", "codecatalyst-runner.amazonaws.com"]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating the `AWSRoleForCodeCatalystSupport` role for your account and space

Follow these steps to create the `AWSRoleForCodeCatalystSupport` role that will be used for support cases in your space. The role must be added to the designated billing account for the space.

Before you begin, you must have administrative privileges for your AWS account or be able to work with your administrator. For more information about how AWS accounts and IAM roles are used in CodeCatalyst, see Managing AWS accounts (p. 36).

**To create and add the CodeCatalyst `AWSRoleForCodeCatalystSupport` role**

1. Before you start in the CodeCatalyst console, open the AWS Management Console, and then make sure you are logged in with the same AWS account for your space.
2. Navigate to your CodeCatalyst space. Choose *Settings*, and then choose *AWS accounts*.
3. Choose the link for the AWS account where you want to create the role. The AWS account details page displays.
4. Choose *Manage roles from AWS Management Console*.

   The *Add IAM role to Amazon CodeCatalyst space* page opens in the AWS Management Console. This is the Amazon CodeCatalyst Spaces page. You might need to sign in to access the page.
5. Under *CodeCatalyst space details*, choose *Add CodeCatalyst Support role*. This option creates a service role that contains the permissions policy and trust policy for the preview development role. The role will have a name `AWSRoleForCodeCatalystSupport` with a unique identifier appended. For more information about the role and role policy, see Understanding the `AWSRoleForCodeCatalystSupport` service role (p. 517).
6. On the *Add role for CodeCatalyst Support* page, leave the default selected, and then choose *Create role*.
7. Under *IAM roles available to CodeCatalyst*, view the CodeCatalystWorkflowDevelopmentRole-*spaceName* role in the list of IAM roles added to your account.
8. To return to your space, choose *Go to Amazon CodeCatalyst*.

Configuring IAM roles for workflow actions in CodeCatalyst

This section details IAM roles and policies that you can create to use with your CodeCatalyst account. For instructions to create example roles, see Creating roles manually for workflow actions (p. 529). After you create your IAM role, copy the role ARN to add the IAM role to your account connection and associate it with your project environment. To learn more, see Adding IAM roles to account connections (p. 40).
**CodeCatalyst build role for Amazon S3 access**

For CodeCatalyst workflow build actions, you can use the default `CodeCatalystWorkflowDevelopmentRole-spaceName` service role, or you can create an IAM role named `CodeCatalystBuildRoleforS3Access`. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on AWS CloudFormation resources in your AWS account.

This role gives permissions to do the following:

- Write to Amazon S3 buckets.
- Support building of resources with AWS CloudFormation. This requires Amazon S3 access.

This role uses the following policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [{
      "Action": [
         "s3:PutObject",
         "iam:PassRole"
      ],
      "Resource": "resource_ARN",
      "Effect": "Allow"
   }]
}
```

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*

**CodeCatalyst build role for AWS CloudFormation**

For CodeCatalyst workflow build actions, you can use the default `CodeCatalystWorkflowDevelopmentRole-spaceName` service role, or you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on AWS CloudFormation resources in your AWS account.

This role gives permissions to do the following:

- Support building of resources with AWS CloudFormation. This is required along with the CodeCatalyst build role for Amazon S3 access and the CodeCatalyst deploy role for AWS CloudFormation.

The following AWS managed policies should be attached to this role:

- AWSCloudFormationFullAccess
- IAMFullAccess
- AmazonS3FullAccess
- AmazonAPIGatewayAdministrator
- AWSSLambdaFullAccess

**CodeCatalyst build role for CDK**

For CodeCatalyst workflows that run CDK build actions, such as Modern three-tier web application, you can use the default `CodeCatalystWorkflowDevelopmentRole-spaceName` service role, or you can
create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to bootstrap and run CDK build commands for AWS CloudFormation resources in your AWS account.

This role gives permissions to do the following:

- Write to Amazon S3 buckets.
- Support building of CDK constructs and AWS CloudFormation resource stacks. This requires access to Amazon S3 for artifact storage, Amazon ECR for image repository support, and SSM for system governance and monitoring for virtual instances.

This role uses the following policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "ecr:*",
        "ssm:*",
        "s3:*",
        "iam:PassRole",
        "iam:GetRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:PutRolePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

**CodeCatalyst deploy role for AWS CloudFormation**

For CodeCatalyst workflow deploy actions that use AWS CloudFormation, you can use the default `CodeCatalystWorkflowDevelopmentRole-spaceName` service role, or you can use a policy with scoped permissions that CodeCatalyst needs to run tasks on AWS CloudFormation resources in your AWS account.

This role gives permissions to do the following:

- Allow CodeCatalyst to invoke a Λ function to perform blue/green deployment through AWS CloudFormation.
- Allow CodeCatalyst to create and update stacks and changesets in AWS CloudFormation.

This role uses the following policy:

```json
{"Action": [
  "cloudformation:CreateStack",
]}
```
"cloudformation:DeleteStack",
"cloudformation:DescribeStack",
"cloudformation:UpdateStack",
"cloudformation:CreateChangeSet",
"cloudformation:DeleteChangeSet",
"cloudformation:ExecuteChangeSet",
"cloudformation:SetStackPolicy",
"cloudformation:ValidateTemplate",
"cloudformation:ListStacks",
"iam:PassRole"
],
"Resource": "resource_ARN",
"Effect": "Allow"
}

Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "**"

CodeCatalyst deploy role for Amazon EC2

CodeCatalyst workflow deploy actions use an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Amazon EC2 resources in your AWS account. The default policy for the CodeCatalystWorkflowDevelopmentRole-spaceName role does not include permissions for Amazon EC2 or Amazon EC2 Auto Scaling.

This role gives permissions to do the following:

- Create Amazon EC2 deployments.
- Read the tags on an instance or identify an Amazon EC2 instance by Auto Scaling group names.
- Read, create, update, and delete Amazon EC2 Auto Scaling groups, lifecycle hooks, and scaling policies.
- Publish information to Amazon SNS topics.
- Retrieve information about CloudWatch alarms.
- Read and update Elastic Load Balancing.

This role uses the following policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:EnableMetricsCollection",
        "autoscaling:DescribePolicies",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeNotificationConfigurations",
        "autoscaling:SuspendProcesses",
        "autoscaling:ResumeProcesses",
```

521
"autoscaling:AttachLoadBalancers",
"autoscaling:AttachLoadBalancerTargetGroups",
"autoscaling:PutScalingPolicy",
"autoscaling:PutScheduledUpdateGroupAction",
"autoscaling:PutNotificationConfiguration",
"autoscaling:PutWarmPool",
"autoscaling:DescribeScalingActivities",
"autoscaling:DeleteAutoScalingGroup",
"ec2:DescribeInstances",
"ec2:DescribeInstanceStatus",
"ec2:TerminateInstances",
"tag:GetResources",
"sns:Publish",
"cloudwatch:DescribeAlarms",
"cloudwatch:PutMetricAlarm",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
"elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
"elasticloadbalancing:DescribeTargetGroups",
"elasticloadbalancing:DescribeTargetHealth",
"elasticloadbalancing:RegisterTargets",
"elasticloadbalancing:DeregisterTargets"
],
"Resource": "resource_ARN"
}]

Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

CodeCatalyst deploy role for Amazon ECS

For CodeCatalyst workflow actions, you can create an IAM role with the necessary permissions. You can use the default CodeCatalystWorkflowDevelopmentRole-spaceName service role, or you can create an IAM role for CodeCatalyst deploy actions to use for Lambda deployments. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Amazon ECS resources in your AWS account.

This role gives permissions to do the following:

- Initiate rolling Amazon ECS deployment on behalf of a CodeCatalyst user, in an account specified in the CodeCatalyst connection.
- Read, update, and delete Amazon ECS task sets.
- Update Elastic Load Balancing target groups, listeners, and rules.
- Invoke Lambda functions.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.
- Publish information to Amazon SNS topics.

This role uses the following policy:

```json
{
  "Version": "2012-10-17",
  "Statement": []
}
```
"Action": [
    "ecs:DescribeServices",
    "ecs:CreateTaskSet",
    "ecs:DeleteTaskSet",
    "ecs:ListClusters",
    "ecs:RegisterTaskDefinition",
    "ecs:UpdateServicePrimaryTaskSet",
    "ecs:UpdateService",
    "elasticloadbalancing:DescribeTargetGroups",
    "elasticloadbalancing:DescribeListeners",
    "elasticloadbalancing:ModifyListener",
    "elasticloadbalancing:DescribeRules",
    "elasticloadbalancing:ModifyRule",
    "lambda:InvokeFunction",
    "lambda:ListFunctions",
    "cloudwatch:DescribeAlarms",
    "sns:Publish",
    "sns:ListTopics",
    "s3:GetObject",
    "s3:GetObjectVersion",
    "codedeploy:CreateApplication",
    "codedeploy:CreateDeployment",
    "codedeploy:CreateDeploymentGroup",
    "codedeploy:GetApplication",
    "codedeploy:GetDeployment",
    "codedeploy:GetDeploymentGroup",
    "codedeploy:ListApplications",
    "codedeploy:ListDeploymentGroups",
    "codedeploy:ListDeployments",
    "codedeploy:StopDeployment",
    "codedeploy:GetDeploymentTarget",
    "codedeploy:ListDeploymentTargets",
    "codedeploy:GetDeploymentConfig",
    "codedeploy:GetApplicationRevision",
    "codedeploy:RegisterApplicationRevision",
    "codedeploy:BatchGetApplicationRevisions",
    "codedeploy:BatchGetDeploymentGroups",
    "codedeploy:BatchGetDeployments",
    "codedeploy:BatchGetApplications",
    "codedeploy:ListApplicationRevisions",
    "codedeploy:ListDeploymentConfigs",
    "codedeploy:ContinueDeployment"
],
"Resource": "*",
"Effect": "Allow"
],
{"Action": [
    "iam:PassRole"
],
"Effect": "Allow",
"Resource": "*",
"Condition": {
    "StringLike": {
        "iam:PassedToService": [
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
        ]
    }
}]

**Note**

The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"
CodeCatalyst deploy role for Lambda

For CodeCatalyst workflow actions, you can create an IAM role with the necessary permissions. You can use the default `CodeCatalystWorkflowDevelopmentRole-spaceName` service role, or you create an IAM role for CodeCatalyst deploy actions to use for Lambda deployments. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Lambda resources in your AWS account.

This role gives permissions to do the following:

- Read, update, and invoke Lambda functions and aliases.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch Events alarms.
- Publish information to Amazon SNS topics.

This role uses the following policy:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```
Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*

CodeCatalyst deploy role for Lambda

For CodeCatalyst workflow actions, you can use the default CodeCatalystWorkflowDevelopmentRole-spaceName service role, or you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Lambda resources in your AWS account.

This role gives permissions to do the following:

- Read, update, and invoke Lambda functions and aliases.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.
- Publish information to Amazon SNS topics.

This role uses the following policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "cloudwatch:DescribeAlarms",
            "lambda:UpdateAlias",
            "lambda:GetAlias",
            "lambda:GetProvisionedConcurrencyConfig",
            "sns:Publish"
        ],
         "Resource": "resource_ARN",
         "Effect": "Allow"
      },
      {
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
        ],
         "Resource": "arn:aws:s3:::/CodeDeploy/",
         "Effect": "Allow"
      },
      {
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
        ],
         "Resource": "",
         "Condition": {
            "StringEquals": {
               "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
            }
        },
         "Effect": "Allow"
      },
      {
         "Action": [
            "lambda:InvokeFunction"
        ],
         "Resource": "resource_ARN",
         "Effect": "Allow"
      }
   ]
}
```
Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

CodeCatalyst deploy role for AWS SAM

For CodeCatalyst workflow actions, you can use the default CodeCatalystWorkflowDevelopmentRole-spaceName service role, or you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on AWS SAM and AWS CloudFormation resources in your AWS account.

This role gives permissions to do the following:

- Allow CodeCatalyst to invoke a Lambda function to perform deployment of serverless and AWS SAM CLI applications.
- Allow CodeCatalyst to create and update stacks and changesets in AWS CloudFormation.

This role uses the following policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "iam:PassRole",
        "iam:DeleteRole",
        "iam:GetRole",
        "iam:TagRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "cloudformation:*",
        "lambda:*",
        "apigateway:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Note
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"
CodeCatalyst read only role for Amazon EC2

For CodeCatalyst workflow actions, you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Amazon EC2 resources in your AWS account. The CodeCatalystWorkflowDevelopmentRole-spaceName service role does not include permissions for Amazon EC2 or the described actions for Amazon CloudWatch.

This role gives permissions to do the following:

• Get status of Amazon EC2 instances.
• Get CloudWatch metrics for Amazon EC2 instances.

This role uses the following policy:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe",
            "Resource": "resource_ARN"
        }
    ]
}
```

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

CodeCatalyst read only role for Amazon ECS

For CodeCatalyst workflow actions, you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Amazon ECS resources in your AWS account.

This role gives permissions to do the following:

• Read Amazon ECS task sets.
- Retrieve information about CloudWatch alarms.

This role uses the following policy:

```json
{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeRules"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam:::role/ecsTaskExecutionRole",
                "arn:aws:iam:::role/ECSTaskExecution"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"
CodeCatalyst read only role for Lambda

For CodeCatalyst workflow actions, you can create an IAM role with the necessary permissions. This role uses a policy with scoped permissions that CodeCatalyst needs to run tasks on Lambda resources in your AWS account.

This role gives permissions for the following:

- Read Lambda functions and aliases.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.

This role uses the following policy.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:DescribeAlarms",
        "lambda:GetAlias",
        "lambda:GetProvisionedConcurrencyConfig"
      ],
      "Resource": "resource_ARN",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::/CodeDeploy/",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
        }
      },
      "Effect": "Allow"
    }
  ]
}
```

**Note**
The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

Creating roles manually for workflow actions

CodeCatalyst workflow actions use IAM roles that you create called the **build role**, the **deploy role**, and the **stack role**.
Follow these steps to create these roles in IAM.

**To create a deploy role**

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   b. Open the IAM console at https://console.aws.amazon.com/iam/.
   c. In the navigation pane, choose Policies.
   d. Choose Create policy.
   e. Choose the JSON tab.
   f. Delete the existing code.
   g. Paste the following code:

   ```json
   {
   "Version": "2012-10-17",
   "Statement": [
   { "Action": [
   "cloudformation:CreateStack",
   "cloudformation:DeleteStack",
   "cloudformation:Describe*",
   "cloudformation:UpdateStack",
   "cloudformation:CreateChangeSet",
   "cloudformation:DeleteChangeSet",
   "cloudformation:ExecuteChangeSet",
   "cloudformation:SetStackPolicy",
   "cloudformation:ValidateTemplate",
   "cloudformation:List*",
   "iam:PassRole"
   ],
   "Resource": "*",
   "Effect": "Allow"
   }],
   "Resource": "*",
   "Effect": "Allow"
   }
   
   Note
   The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

   "Resource": "*"
   
   h. Choose Next: Tags.
   i. Choose Next: Review.
   j. In Name, enter:

   **codecatalyst-deploy-policy**

   k. Choose Create policy.

   You have now created a permissions policy.

2. Create the deploy role, as follows:
   a. In the navigation pane, choose Roles, and then choose Create role.
   b. Choose Custom trust policy.
   c. Delete the existing custom trust policy.
   d. Add the following custom trust policy:
The code is shown in the image:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "codecatalyst-runner.amazonaws.com",
               "codecatalyst.amazonaws.com"
            ]
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

e. Choose **Next**.

f. In **Permissions policies**, search for `codecatalyst-deploy-policy` and select its check box.

g. Choose **Next**.

h. For **Role name**, enter:

   ```
   codecatalyst-deploy-role
   ```

i. For **Role description**, enter:

   ```
   CodeCatalyst deploy role
   ```

j. Choose **Create role**.

You have now created a deploy role with a trust policy and permissions policy.

3. Obtain the deploy role ARN, as follows:

   a. In the navigation pane, choose **Roles**.

   b. In the search box, enter the name of the role you just created (`codecatalyst-deploy-role`).

   c. Choose the role from the list.

   The role's **Summary** page appears.

   d. At the top, copy the **ARN** value.

You have now created the deploy role with the appropriate permissions, and obtained its ARN.

**To create a build role**

1. Create a policy for the role, as follows:

   a. Sign in to AWS.


   c. In the navigation pane, choose **Policies**.

   d. Choose **Create policy**.

   e. Choose the **JSON** tab.

   f. Delete the existing code.

   g. Paste the following code:
Note

The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

"Resource": "*"

h. Choose **Next: Tags**.

i. Choose **Next: Review**.

j. In **Name**, enter:

```
codecatalyst-build-policy
```

k. Choose **Create policy**.

You have now created a permissions policy.

2. Create the build role, as follows:

a. In the navigation pane, choose **Roles**, and then choose **Create role**.

b. Choose **Custom trust policy**.

c. Delete the existing custom trust policy.

d. Add the following custom trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codecatalyst-runner.amazonaws.com",
          "codecatalyst.amazonaws.com"
        ],
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

e. Choose **Next**.

f. In **Permissions policies**, search for **codecatalyst-build-policy** and select its check box.

 g. Choose **Next**.

h. For **Role name**, enter:
For **Role description**, enter:

**CodeCatalyst build role**

Choose **Create role**.

You have now created a build role with a trust policy and permissions policy.

3. **Obtain the build role ARN, as follows:**
   a. In the navigation pane, choose **Roles**.
   b. In the search box, enter the name of the role you just created (`codecatalyst-build-role`).
   c. Choose the role from the list.
   
   The role's **Summary** page appears.
   d. At the top, copy the **ARN** value.

You have now created the build role with the appropriate permissions, and obtained its ARN.

**To create a stack role**

**Note**
You don't have to create a stack role, although doing so is recommended for security reasons. If you don't create the stack role, you'll need to add the permissions policies described further on in this procedure to the deploy role.

1. Sign in to AWS using the account where you want to deploy your stack.
3. In the navigation pane, choose **Roles** and then choose **Create role**.
4. At the top, choose **AWS service**.
5. From the list of services, choose **CloudFormation**.
6. Choose **Next: Permissions**.
7. In the search box, add any policies that are required to access the resources in your stack. For example, if your stack includes an AWS Lambda function, you need to add a policy that grants access to Lambda.

   **Tip**
   If you're unsure which policies to add, you can omit them for now. When you test the action, if you don't have the right permissions, AWS CloudFormation generates errors that show which permissions you need to add.
8. Choose **Next: Tags**.
9. Choose **Next: Review**.
10. For **Role name**, enter:

    **codecatalyst-stack-role**

11. Choose **Create role**.
12. To obtain the stack role's ARN, do the following:

    a. In the navigation pane, choose **Roles**.
b. In the search box, enter the name of the role you just created (codecatalyst-stack-role).

c. Choose the role from the list.

d. On the **Summary** page, copy the **Role ARN** value.

**Using AWS CloudFormation to create policies and roles in IAM**

You can choose to create and use AWS CloudFormation templates to create the policies and roles you need to access resources in an AWS account for your CodeCatalyst projects and workflows. AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. If you intend to create roles in multiple AWS accounts, creating a template can help you perform this task more quickly.

The following example template creates a deploy action role and policy.

```json
Parameters:
  CodeCatalystAccountId:
    Type: String
    Description: Account ID from the connections page
  ExternalId:
    Type: String
    Description: External ID from the connections page

Resources:
  CrossAccountRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS:
                - !Ref CodeCatalystAccountId
            Action:
              - 'sts:AssumeRole'
            Condition:
              StringEquals:
                sts:ExternalId: !Ref ExternalId
            Path: /
      Policies:
        - PolicyName: CodeCatalyst-CloudFormation-action-policy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - 'cloudformation:CreateStack'
                  - 'cloudformation:DeleteStack'
                  - 'cloudformation:Describe*''
                  - 'cloudformation:UpdateStack'
                  - 'cloudformation:CreateChangeSet'
                  - 'cloudformation:DeleteChangeSet'
                  - 'cloudformation:ExecuteChangeSet'
                  - 'cloudformation:SetStackPolicy'
                  - 'cloudformation:ValidateTemplate'
                  - 'cloudformation:List**'
                Resource: '***'
```
Creating the role manually for the web application blueprint

The CodeCatalyst web application blueprint uses IAM roles that you create called the build role for CDK, the deploy role, and the stack role.

Follow these steps to create the role in IAM.

To create a build role

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   b. Open the IAM console at https://console.aws.amazon.com/iam/.
   c. In the navigation pane, choose Policies.
   d. Choose Create Policy.
   e. Choose the JSON tab.
   f. Delete the existing code.
   g. Paste the following code:

   ```json
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "cloudformation:*",
                   "ecr:*",
                   "ssm:*",
                   "s3:*",
                   "iam:PassRole",
                   "iam:GetRole",
                   "iam:CreateRole",
                   "iam:AttachRolePolicy",
                   "iam:PutRolePolicy"
               ],
               "Resource": "*"
           }
       ]
   }
   
   Note
   The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

   "Resource": "*"
   
   h. Choose Next: Tags.
   i. Choose Next: Review.
   j. In Name, enter:

   **codectalyst-webapp-build-policy**

   k. Choose Create policy.

   You have now created a permissions policy.

2. Create the build role, as follows:
a. In the navigation pane, choose Roles, and then choose Create role.
b. Choose Custom trust policy.
c. Delete the existing custom trust policy.
d. Add the following custom trust policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": 
         "Effect": "Allow",
         "Principal": [
            "codecatalyst-runner.amazonaws.com",
            "codecatalyst.amazonaws.com"
         ],
         "Action": "sts:AssumeRole"
      }
   ]
}
```
e. Choose Next.
f. Attach the permissions policy to the build role. On the Add permissions page, in the Permissions policies section, search for codecatalyst-webapp-build-policy and select its check box.
g. Choose Next.
h. For Role name, enter:

`codecatalyst-webapp-build-role`
i. For Role description, enter:

`CodeCatalyst Web app build role`
j. Choose Create role.

You have now created a build role with a trust policy and permissions policy.

3. Attach the permissions policy to the build role, as follows:

a. In the navigation pane, choose Roles, and then search for codecatalyst-webapp-build-role.
b. Choose codecatalyst-webapp-build-role to display its details.
c. In the Permissions tab, choose Add permissions, and then choose Attach policies.
d. Search for codecatalyst-webapp-build-policy, select its check box, and then choose Attach policies.

You have now attached the permissions policy to the build role. The build role now has two policies: a permissions policy and a trust policy.

4. Obtain the build role ARN, as follows:

a. In the navigation pane, choose Roles.
b. In the search box, enter the name of the role you just created (codecatalyst-webapp-build-role).
c. Choose the role from the list.
   The role's **Summary** page appears.

d. At the top, copy the **ARN** value.

You have now created the build role with the appropriate permissions, and obtained its ARN.

**Creating roles manually for the SAM blueprint**

The CodeCatalyst SAM blueprint uses IAM roles that you create called the **build role for CloudFormation** and the **deploy role for SAM**.

Follow these steps to create the roles in IAM.

**To create a build role for CloudFormation**

1. Create a policy for the role, as follows:
   a. Sign in to AWS.
   c. In the navigation pane, choose **Policies**.
   d. Choose **Create Policy**.
   e. Choose the **JSON** tab.
   f. Delete the existing code.
   g. Paste the following code:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "s3:*",
           "cloudformation:*"
         ],
         "Resource": "*"
       }
     ]
   }
   ```

   **Note**
   The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

   h. Choose **Next: Tags**.
   i. Choose **Next: Review**.
   j. In **Name**, enter:

   ```
   codecatalyst-SAM-build-policy
   ```

   k. Choose **Create policy**.
You have now created a permissions policy.

2. Create the build role, as follows:
   a. In the navigation pane, choose Roles, and then choose Create role.
   b. Choose Custom trust policy.
   c. Delete the existing custom trust policy.
   d. Add the following custom trust policy:

   ```json
   {
   "Version": "2012-10-17",
   "Statement": [
   {
   "Sid": "",
   "Effect": "Allow",
   "Principal": {
   "Service": [
   "codecatalyst-runner.amazonaws.com",
   "codecatalyst.amazonaws.com"
   ],
   "Action": "sts:AssumeRole"
   }
   ]
   }
   }
   ```
   e. Choose Next.
   f. Attach the permissions policy to the build role. On the Add permissions page, in the Permissions policies section, search for codecatalyst-SAM-build-policy and select its check box.
   g. Choose Next.
   h. For Role name, enter:

   ```text
   codecatalyst-SAM-build-role
   ```
   i. For Role description, enter:

   ```text
   CodeCatalyst SAM build role
   ```
   j. Choose Create role.

   You have now created a build role with a trust policy and permissions policy.

3. Attach the permissions policy to the build role, as follows:
   a. In the navigation pane, choose Roles, and then search for codecatalyst-SAM-build-role.
   b. Choose codecatalyst-SAM-build-role to display its details.
   c. In the Permissions tab, choose Add permissions, and then choose Attach policies.
   d. Search for codecatalyst-SAM-build-policy, select its check box, and then choose Attach policies.

   You have now attached the permissions policy to the build role. The build role now has two policies: a permissions policy and a trust policy.

4. Obtain the build role ARN, as follows:
   a. In the navigation pane, choose Roles.
b. In the search box, enter the name of the role you just created (codecatalyst-SAM-build-role).

c. Choose the role from the list.

The role's Summary page appears.

d. At the top, copy the ARN value.

You have now created the build role with the appropriate permissions, and obtained its ARN.

**To create a deploy role for SAM**

1. Create a policy for the role, as follows:

a. Sign in to AWS.


c. In the navigation pane, choose Policies.

d. Choose Create Policy.

e. Choose the JSON tab.

f. Delete the existing code.

g. Paste the following code:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "iam:PassRole",
        "iam:DeleteRole",
        "iam:GetRole",
        "iam:TagRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "cloudformation:*",
        "lambda:*",
        "apigateway:*"
      ],
      "Resource": "*"
    }
  ]
}
```

**Note**

The first time the role is used to run workflow actions, use the wildcard in the resource policy statement and then scope down the policy with the resource name after it is available.

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

h. Choose Next: Tags.

i. Choose Next: Review.

j. In Name, enter:
Choose Create policy.

You have now created a permissions policy.

2. Create the build role, as follows:
   a. In the navigation pane, choose Roles, and then choose Create role.
   b. Choose Custom trust policy.
   c. Delete the existing custom trust policy.
   d. Add the following custom trust policy:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": [
             "codecatalyst-runner.amazonaws.com",
             "codecatalyst.amazonaws.com"
           ],
           "Action": "sts:AssumeRole"
         }
       }
     ]
   }
   ```
   
   e. Choose Next.
   f. Attach the permissions policy to the build role. On the Add permissions page, in the Permissions policies section, search for codecatalyst-SAM-deploy-policy and select its check box.
   g. Choose Next.
   h. For Role name, enter:

   ```
   codecatalyst-SAM-deploy-role
   ```
   
   i. For Role description, enter:

   ```
   CodeCatalyst SAM deploy role
   ```
   
   j. Choose Create role.

   You have now created a build role with a trust policy and permissions policy.

3. Attach the permissions policy to the build role, as follows:
   a. In the navigation pane, choose Roles, and then search for codecatalyst-SAM-deploy-role.
   b. Choose codecatalyst-SAM-deploy-role to display its details.
   c. In the Permissions tab, choose Add permissions, and then choose Attach policies.
   d. Search for codecatalyst-SAM-deploy-policy, select its check box, and then choose Attach policies.

   You have now attached the permissions policy to the build role. The build role now has two policies: a permissions policy and a trust policy.
4. Obtain the build role ARN, as follows:
   a. In the navigation pane, choose Roles.
   b. In the search box, enter the name of the role you just created (codecatalyst-SAM-deploy-role).
   c. Choose the role from the list.
      The role's Summary page appears.
   d. At the top, copy the ARN value.

You have now created the build role with the appropriate permissions, and obtained its ARN.

Compliance validation for Amazon CodeCatalyst

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

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

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

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

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

- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the AWS Config Developer Guide – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- **AWS Audit Manager** – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon CodeCatalyst

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through 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. To learn more about what CodeCatalyst data is replicated across AWS Regions, see Data protection in Amazon CodeCatalyst (p. 504).

Infrastructure Security in Amazon CodeCatalyst

As a managed service, Amazon CodeCatalyst 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 CodeCatalyst 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.

Configuration and vulnerability analysis in Amazon CodeCatalyst

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model.

Your data and privacy in Amazon CodeCatalyst

Amazon CodeCatalyst takes your privacy seriously, and the security of your information is our top priority. You can review more about how we handle your information in the AWS Privacy Notice.

To request and view your data, see Requesting your data in the AWS General Reference.

Deleting your AWS Builder ID profile

Deleting your profile is a permanent action that cannot be reversed. The deletion process begins immediately after you choose Delete. Amazon CodeCatalyst starts deleting your profile and all associated personal information. This process may take up to 90 days to complete.

When your profile is deleted, you cannot access or recover your data in Amazon CodeCatalyst. This includes Personal Access Tokens, Roles, User Memberships, and any Amazon CodeCatalyst spaces of which you are the only member. You can no longer sign in to Amazon CodeCatalyst.

For information on how to delete your AWS Builder ID profile, see Deleting your AWS Builder ID in the AWS General Reference.
Best practices for workflow actions in Amazon CodeCatalyst

There are a number of security best practices to consider as you develop your workflows in CodeCatalyst. The following are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

Topics

- Sensitive information (p. 543)
- Licensing terms (p. 543)
- Untrusted code (p. 543)
- GitHub Actions (p. 543)

Sensitive information

Do not embed sensitive information in your YAML. Rather than embedding credentials, keys, or tokens in your YAML, we recommend you use CodeCatalyst secrets. Secrets provide an easy way to store and reference sensitive information from within your YAML.

Licensing terms

Make sure to pay attention to the licensing terms of the action you choose to use.

Untrusted code

Actions are generally self-contained, single purpose modules that can be shared across a project, space, or the broader community. Using code from others can be a great convenience and efficiency gain, but also introduces a new threat vector. Review the following sections to ensure you’re following best practices to keep your CI/CD workflows secure.

GitHub Actions

GitHub Actions are open source, built and maintained by the community. We follow the shared responsibility model and consider GitHub Actions source code as customer data for which you are responsible. GitHub Actions can be granted access to secrets, repository tokens, source code, account links, and your compute time. Make sure you are confident in the trustworthiness and security of the GitHub Actions you plan to run.

More specific guidance and security best practices for GitHub Actions:

- Security hardening
- Preventing pwn requests
- Untrusted input
- How to trust your building blocks

CodeCatalyst trust model

The Amazon CodeCatalyst trust model allows CodeCatalyst to assume the service role in the connected AWS account. The model connects the IAM role, the CodeCatalyst service principals, and the CodeCatalyst space. The trust policy uses the `aws:SourceArn` condition key to grant permissions to the
A trust policy is a JSON policy document in which you define the principals that you trust to assume the role. A role trust policy is a required resource-based policy that is attached to a role in IAM. For more information, see Terms and concepts in the IAM User Guide. For details about the service principals for CodeCatalyst, see Service principals for CodeCatalyst (p. 544).

In the following trust policy, the service principals listed in the Principal element are granted permissions from the resource-based policy, and the Condition block is used to limit access to the scoped-down resource.

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": [
        "codecatalyst-runner.amazonaws.com",
        "codecatalyst.amazonaws.com"
      ],
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/*"
      }
    }
  }
]
```

In the trust policy, the CodeCatalyst service principals are given access through the `aws:SourceArn` condition key, which contains the Amazon Resource Name (ARN) for the CodeCatalyst space ID. The ARN uses the following format:

```
arn:aws:codecatalyst:::space/spaceId/project/*
```

**Important**

Use the space ID only in condition keys, such as `aws:SourceArn`. Do not use the space ID in IAM policy statements as a resource ARN.

As a best practice, scope down permissions as much as possible in the policy.

- You can use the wildcard (*) in the `aws:SourceArn` condition key for specifying all projects in the space with `project/*`.
- You can specify resource-level permissions in the `aws:SourceArn` condition key for a specific project in the space with `project/projectId`.

**Service principals for CodeCatalyst**

You use the Principal element in a resource-based JSON policy to specify the principal that is allowed or denied access to a resource. The principals that you can specify in the trust policy include users, roles, accounts, and services. You cannot use the Principal element in an identity-based policy; similarly, you cannot identify a user group as a principal in a policy (such as a resource-based policy) because groups relate to permissions, not authentication, and principals are authenticated IAM entities.

In the trust policy, you can specify AWS services in the Principal element of a resource-based policy or in condition keys that support principals. Service principals are defined by the service. The following are the service principals defined for CodeCatalyst:
Monitoring in Amazon CodeCatalyst

In Amazon CodeCatalyst, management events for the space are collected by AWS CloudTrail and are logged in the trail for the billing account for the space. CloudTrail logging is the primary method to manage logging for CodeCatalyst events, and a secondary method is viewing event logging in CodeCatalyst.

Events in the account are logged with the trail and designated bucket that is set up for the AWS account.

The following diagram shows how all management events for the space are logged in CloudTrail for the billing account, while account connections/billing events and AWS resource events are logged in CloudTrail for the respective account.

The diagram illustrates the following steps:

1. When a space is created, an AWS account is connected to the space and is designated as the billing account. The trail used is the trail that was created in CloudTrail for the billing account, where space events are logged. CloudTrail captures API calls and related events made by or on behalf of a CodeCatalyst space and delivers the log files to an S3 bucket that you specify. If the billing account changes to another AWS account, then space events are logged in the trail and bucket for that account. For more information about CodeCatalyst management events that are logged by CloudTrail, see CodeCatalyst information in CloudTrail (p. 549).

2. Other accounts connected to the space, including the billing account, log a subset of events for account connections and billing events. CodeCatalyst workflows that generate account events...
for AWS resources deployed for that account are also logged in the trail and bucket for the AWS account. CloudTrail captures API calls and related events made by or on behalf of a CodeCatalyst space and delivers the log files to an S3 bucket that you specify. For more information about CodeCatalyst management events that are logged by CloudTrail, see Accessing logged events in CodeCatalyst (p. 546).

3. You can also monitor CodeCatalyst actions in your space within a specific time in the space with the list-event-logs command using the AWS CLI. For more information, see the Amazon CodeCatalyst API Reference Guide. You must have the Space administrator role to call the list of events for CodeCatalyst actions in your space. For more information, see Accessing logged events in CodeCatalyst (p. 546).

Note
ListEventLogs guarantees events for the last 30 days in a given space. You can also view and retrieve a list of management events over the last 90 days for CodeCatalyst in the AWS CloudTrail console by viewing Event history, or by creating a trail to create and maintain a record of events that extends past 90 days. For more information, see Working with CloudTrail Event history and Working with CloudTrail trails.

Note
AWS resources that are deployed into connected accounts for CodeCatalyst workflows, are not logged as part of CloudTrail logging for the CodeCatalyst space. For example, CodeCatalyst resources include a space or project. AWS resources include an Amazon ECS service or Lambda function. You must configure CloudTrail logging separately for each AWS account where resources are deployed into.

Here is one possible flow for event monitoring in CodeCatalyst.

Mary Major is a Space administrator for a CodeCatalyst space and views all management events in CodeCatalyst for space-level and project-level resources in the space that are logged in CloudTrail. See CodeCatalyst information in CloudTrail (p. 549) for example events that are logged in CloudTrail.

For resources that are created in CodeCatalyst, such as Dev Environments, Mary views the Event history in the billing account for the space and investigates events where Dev Environments were created by project members in CodeCatalyst. The event provides the identity store IAM identity type and credentials for the AWS Builder ID for the user who created the Dev Environment. For resources that are created in AWS when deployed by workflows in CodeCatalyst, such as a Lambda function for a serverless deployment, the AWS account owner can view the event history for the trail associated with the separate AWS account (which is also a connected account to CodeCatalyst) for the workflow deploy action.

To investigate further, Mary can also view events for all CodeCatalyst APIs in the space by using the list-event-logs command in the AWS CLI.

Topics
- Logging CodeCatalyst API calls in connected AWS accounts using AWS CloudTrail (p. 546)
- Accessing logged events in CodeCatalyst (p. 551)

Logging CodeCatalyst API calls in connected AWS accounts using AWS CloudTrail

Amazon CodeCatalyst is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures API calls made on behalf of CodeCatalyst in connected AWS accounts as events. If you create a trail, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for CodeCatalyst. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.
CodeCatalyst supports logging the following actions as events in CloudTrail log files:

- Management events for CodeCatalyst spaces will be logged in the AWS account that is the designated billing account for the space. For more information, see CodeCatalyst space events (p. 547).
  
  **Note**
  
  Data events for CodeCatalyst spaces are accessible by using the CLI as detailed in Accessing logged events in CodeCatalyst (p. 551).

- Events for resources that are used in CodeCatalyst workflow actions that occur in a connected AWS account will be logged as events in that AWS account. For more information, see CodeCatalyst account connections and billing events (p. 548).

**Important**

While multiple accounts can be associated with a space, CloudTrail logging for events in CodeCatalyst spaces and projects apply only for the billing account.

The space billing account is your AWS account that is charged for CodeCatalyst resources beyond the AWS Free tier. Multiple accounts can be connected to a space, while only one account can be the designated billing account. The billing account or additional connected accounts for the space can have IAM roles that are used for deploying AWS resources and infrastructure, such as an Amazon ECS cluster or S3 bucket, from CodeCatalyst workflows. You can use the workflow YAML to identify the AWS account that you deployed to.

**Note**

AWS resources that are deployed into connected accounts for CodeCatalyst workflows, are not logged as part of CloudTrail logging for the CodeCatalyst space. For example, CodeCatalyst resources include a space or project. AWS resources include an Amazon ECS service or Lambda function. CloudTrail logging must be configured separately for each AWS account where resources are deployed into.

CodeCatalyst logging in connected accounts includes the following considerations:

- Access to CloudTrail events is managed with IAM in the connected account and not in CodeCatalyst.

- Third-party connections, such as linking to a GitHub repository, will result in third-party resource names being recorded in CloudTrail logs.

**Note**

CloudTrail logging for CodeCatalyst events is at the space level and does not isolate events by project boundaries.

For more information about CloudTrail, see the AWS CloudTrail User Guide.

**Note**

This section describes CloudTrail logging for all events logged in a logged in a CodeCatalyst space and the AWS accounts that are connected to CodeCatalyst. Additionally, to review all events logged in a CodeCatalyst space, you can also use the AWS CLI and the aws codecatalyst list-event-logs command. For more information, see Accessing logged events in CodeCatalyst (p. 551).

**CodeCatalyst space events**

Actions in CodeCatalyst for managing space-level and project-level resources are logged in the billing account for the space. For CloudTrail logging for a CodeCatalyst space, events are logged with the following considerations:

- CloudTrail events apply across the entire space and are not scoped to any single project.
- When you connect an AWS account to a CodeCatalyst space, loggable events for account connections will be logged in that AWS account. After you enable this connection, you cannot disable it.
• When you connect an AWS account to a CodeCatalyst space and designate it as the billing account for the space, events will be logged in that AWS account. After you enable this connection, you cannot disable it.

Events for space-level and project-level resources are logged only in the billing account. To change the CloudTrail destination account, update the billing account in CodeCatalyst. At the beginning of the next monthly billing cycle, the change takes effect for the new billing account in CodeCatalyst. After that, the CloudTrail destination account is updated.

The following are examples of events in AWS that are related to actions in CodeCatalyst for managing space-level and project-level resources. The following APIs are released through the SDK and CLI. Events will be logged in the AWS account specified as the billing account for the CodeCatalyst space.

• CreateDevEnvironment
• CreateProject
• DeleteDevEnvironment
• GetDevEnvironment
• GetProject
• GetSpace
• GetSubscription
• ListDevEnvironments
• ListDevEnvironmentSessions
• ListEventLogs
• ListProjects
• ListSourceRepositories
• StartDevEnvironment
• StartDevEnvironmentSession
• StopDevEnvironment
• StopDevEnvironmentSession
• UpdateDevEnvironment

CodeCatalyst account connections and billing events

The following are examples of events in AWS that are related to actions in CodeCatalyst for account connections or billing:

• AcceptConnection
• AssociateIAMRoleToConnection
• DeleteConnection
• DissociateIAMRoleFromConnection
• GetBillingAuthorization
• GetConnection
• GetPendingConnection
• ListConnections
• ListIAMRolesForConnection
• PutBillingAuthorization
CodeCatalyst information in CloudTrail

CloudTrail is enabled on an AWS account when you create that account. When you connect that AWS account to a CodeCatalyst space, events for that space that occur in that AWS account are logged in CloudTrail logs in that AWS account. Loggable events in CodeCatalyst are recorded as CloudTrail events in CloudTrail logs in the connected account and in Event history in the CloudTrail console, along with other loggable AWS events in that account.

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 by a user with their AWS Builder ID.
- 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.

Accessing CloudTrail events

For an ongoing record of events in your AWS account, including events for CodeCatalyst activity in the AWS account, create a trail. A trail enables CloudTrail to deliver log files to an 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 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

A trail is a configuration that enables delivery of events as log files to an 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.

Example CodeCatalyst account connections event in AWS

The following example shows a CloudTrail log entry that demonstrates the ListConnections action. For an AWS account that is connected to the space, ListConnections is used to view all account connections to CodeCatalyst for this AWS account. The event will be logged in the AWS account specified in accountId, and the value of the arn will be the Amazon Resource Name (ARN) of the role used for the action.

```json
{}
```
Logging CodeCatalyst API calls to connected accounts in AWS CloudTrail

Example CodeCatalyst project resource event in AWS

The following example shows a CloudTrail log entry that demonstrates the `CreateDevEnvironment` action. An AWS account that is connected to the space and is the designated billing account for the space is used for project-level events in the space, such as creating a Dev Environment.

Under `userIdentity`, in the `accountId` field, this is the IAM Identity Center account ID (432677196278) that hosts the identity pool for all AWS Builder ID identities. This account ID contains the following information about the CodeCatalyst user for the event.

- The `type` field indicates the type of IAM entity for the request. For CodeCatalyst events for space and project resources, this value is `IdentityCenterUser`. The `accountId` field specifies the account that owns the entity that was used to get credentials.
- The `userId` field contains the AWS Builder ID identifier for the user.
- The `identityStoreArn` field contains the role ARN for the identity store account and user.

The `recipientAccountId` field contains the account ID for the billing account for the space, with an example value here of 111122223333.

For more information, see the CloudTrail `userIdentity` element.
Accessing logged events in CodeCatalyst

When users perform actions in Amazon CodeCatalyst, these actions are recorded as events. You can use the AWS CLI to view logs of events in a space in a specified timeframe. You can view these events to review actions taken in the space, including the date and time of the actions, the name of the user who performed the action, and the IP address where the user made the request.

Note
In certain events, the user agent may not be known. In this case, CodeCatalyst will provide a value of Unknown in the userAgent field in the CloudTrail event.

Querying your CodeCatalyst event trails

You can create and manage queries for your CloudTrail logs using a query table in Amazon Athena. For more information about creating a query, see Querying AWS CloudTrail logs in the Amazon Athena User Guide.

Accessing logged events in CodeCatalyst

| "eventVersion": "1.09",
| "userIdentity": {
| "type": "IdentityCenterUser",
| "accountId": "432677196278",
| "onBehalfOf": {
| "userId": "user-ID",
| "identityStoreArn": "arn:aws:identitystore::432677196278:identitystore/d-9067642ac7"
| },
| "credentialId": "ABCDefGhiJKLMn11Lmn_1AbCDEFGHIjKLmn0Pqrs11abEXAMPLE"
| },
| "eventTime": "2023-05-18T17:10:50Z",
| "eventSource": "codecatalyst.amazonaws.com",
| "eventName": "CreateDevEnvironment",
| "awsRegion": "us-west-2",
| "sourceIPAddress": "192.168.0.1",
| "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0",
| "requestParameters": {
| "spaceName": "MySpace",
| "projectName": "MyProject",
| "ides": [{
| "runtime": "public.ecr.aws/q6e8p2q0/cloud9-ide-runtime:2.5.1",
| "name": "Cloud9"
| }],
| "instanceType": "dev.standard1.small",
| "inactivityTimeoutMinutes": 15,
| "persistentStorage": {
| "sizeInGiB": 16
| }
| },
| "responseElements": {
| "spaceName": "MySpace",
| "projectName": "MyProject",
| "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
| },
| "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
| "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
| "readOnly": false,
| "eventType": "AwsApiCall",
| "managementEvent": true,
| "recipientAccountId": "111122223333",
| "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
| "eventCategory": "Management"

551
Note
Management events for a CodeCatalyst space are logged in CloudTrail for the connected billing account. For more information about CodeCatalyst management events that are logged by CloudTrail, see CodeCatalyst information in CloudTrail (p. 549).

In order to view a log of events for a space, you must have installed and configured the AWS CLI with a profile for CodeCatalyst, and you must have the Space administrator role for the space. For more information, see Setting up to use the AWS CLI with CodeCatalyst (p. 15) and Space administrator role (p. 477).

Note
To view logging for events that occur on behalf of CodeCatalyst in connected AWS accounts, or to view logging for events for space or project resources in the connected billing account, you can use AWS CloudTrail. For more information, see Logging CodeCatalyst API calls in connected AWS accounts using AWS CloudTrail (p. 546).

1. Open a terminal or command line and run the `aws codecatalyst list-event-logs` command, specifying:
   - The name of the space with the `--space-name` option.
   - The date and time when you want to start reviewing events, in coordinated universal time (UTC) timestamp format as specified in RFC 3339, with the `--start-time` option.
   - The date and time when you want to stop reviewing events, in coordinated universal time (UTC) timestamp format as specified in RFC 3339, with the `--end-time` option.
   - (Optional) The maximum number of results to return in a single response, with the `--max-results` option. If the number of results is larger than the number you specify, the response will include a nextToken element which you can use to return the next results.
   - (Optional) Limit the results to a specific event type you want returned, with the `--event-name` option.

This example returns logged events in the space named ExampleCorp from the time period 2022-11-30 to 2022-12-01, and that a maximum of 2 events be returned in the response.

```bash
aws codecatalyst list-event-logs --space-name ExampleCorp --start-time 2022-11-30 --end-time 2022-12-01 --event-name list-event-logs --max-results 2
```

2. If events occurred in this time frame, the command returns results similar to the following:

```
{
   "nextToken": "EXAMPLE",
   "items": [ {
      "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "eventName": "listEventLogs",
      "eventType": "AwsApiCall",
      "eventCategory": "MANAGEMENT",
      "eventSource": "manage",
      "eventTime": "2022-12-01T22:47:24.605000+00:00",
      "operationType": "READONLY",
      "userIdentity": { "userType": "USER", "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111" },
      "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "requestPayload": { "contentType": "application/json", "data": "{\"spaceName\":\"ExampleCorp\",\"startTime\":\"2022-12-01T00:00:00Z\",\"endTime\":\"2022-12-10T00:00:00Z\",\"maxResults\":\"2\"}
```
3. Run the `list-event-logs` command again with the `--next-token` option and the value of the returned token to retrieve the next set of logged events that match the request.

### Quotas for identity, permission, and access in CodeCatalyst

The following table describes quotas and limits for identity, permission, and access in Amazon CodeCatalyst. For more information about quotas in Amazon CodeCatalyst, see [Quotas for CodeCatalyst (p. 620)](#).

<table>
<thead>
<tr>
<th>Resource</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases in CodeCatalyst</td>
<td>Any combination of allowed characters between 3 and 100 characters in length and must start with a letter. Valid characters: A-Z, a-z, and 0-9. Aliases cannot:</td>
</tr>
<tr>
<td></td>
<td>- contain fewer than 3 characters</td>
</tr>
<tr>
<td></td>
<td>- contain spaces or any of the following characters: ? ^ * [ \ ~ :</td>
</tr>
</tbody>
</table>
### Resource Information

<table>
<thead>
<tr>
<th>Resource</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of invitations sent by a user per day</td>
<td>500</td>
</tr>
<tr>
<td>Maximum number of invitations sent to an email address per day</td>
<td>25</td>
</tr>
<tr>
<td>Maximum number of Personal Access Tokens (PAT) per user</td>
<td>100</td>
</tr>
<tr>
<td>Passwords in CodeCatalyst</td>
<td>Any combination of allowed characters between 8 and 64 characters in length. Valid characters: A-Z, a-z, and 0-9. Your password can include the following nonalphanumeric characters: (~ ! @ # $ % ^ &amp; * _ - + = `</td>
</tr>
<tr>
<td>PAT names in CodeCatalyst</td>
<td>Any combination of allowed characters between 1 and 100 characters</td>
</tr>
<tr>
<td>Time until a project member invitation expires</td>
<td>Expires after 24 hours</td>
</tr>
<tr>
<td>Time until a space member invitation expires</td>
<td>Expires after 24 hours</td>
</tr>
<tr>
<td>Time until an email address verification expires</td>
<td>Expires 10 minutes after sending</td>
</tr>
</tbody>
</table>

### Troubleshooting

This section can help you troubleshoot some common issues you might encounter while accessing your Amazon CodeCatalyst profile.

#### Problems signing up

You might encounter some issues while signing up. We've got some solutions.

**My email address is already in use**

If the email that you entered is already in use and you recognize it as your own, then you may already have a profile with us. Sign in with this existing identity. If you do not own the existing email, then sign up with a different, unused one.

**I can't complete email verification**

If you have not received your verification email

1. Check your spam, junk, and deleted items folder.

   **Note**
   This verification email comes from either the address no-reply@signin.aws or no-reply@login.awsapps.com. We recommend that you configure your mail system so that it accepts emails from these sender email addresses and does not handle them as junk or spam.

2. Wait 5 minutes and refresh your inbox. Check your spam, junk, and deleted items folder again.

3. If you still don't see your verification email, choose Resend code. If you exited that page already, then restart your workflow for signing up with Amazon CodeCatalyst.
My password doesn't meet minimum requirements

For your security, your password must include 8-20 characters, both uppercase and lowercase letters, and numbers.

Problems signing in

I forgot my password

Follow the steps in I forgot my password (p. 582).

My password isn’t working

You must follow these requirements whenever you set or change your password:

- Passwords are case-sensitive.
- Passwords must be between 8 and 64 characters in length with both uppercase and lowercase letters, numbers, and at least one non-alphanumeric character.
- You can't reuse the last three passwords.

I can't enable MFA

To enable MFA, add one or more MFA devices to your profile by following the steps in Multi-factor authentication (MFA) in Amazon CodeCatalyst (p. 500).

I can't add an MFA device

If you find that you can't add another MFA device, it may have reached the limit of MFA devices that you can register. You may need to remove an existing MFA device before adding a new one.

I can't remove an MFA device

If you intend to disable MFA, then proceed with removing your MFA device by following the steps in Deleting an MFA device (p. 503). However, if you want to keep MFA enabled, you should add another MFA device before attempting to delete an existing MFA device. For more information about adding another MFA device, see How to register a device for use with multi-factor authentication (p. 501).

Problems signing out

I can't find where to sign out

In the top right corner of the page, choose Sign out.

Sign out doesn't sign me out completely

The system is designed to sign out immediately, but full sign out may take up to an hour.

I get a role does not exist error for a failed workflow

Issue: After creating a project from the web application or serverless blueprint, the workflow fails with the following error:
I get a role error for a failed workflow

**Issue:** After creating a project from the web application or serverless blueprint, the workflow fails with the following error:

CLIENT_ERROR: Role not set up properly or does not exist

**Possible solution:** The space where the project was created might need to set up an AWS account connection or might need to complete an account connection request. If your space already has an active AWS account connection, create and add an IAM role with permissions to run workflow actions. Add the IAM role to your account connection as detailed in [Adding IAM roles to account connections](p. 40).

I need to update the IAM role in a project workflow

If the AWS account connection is set up completely, and the IAM role is created and added to the account connection, you can update the IAM role in your project workflow.

1. Choose the **CI/CD** option and choose your workflow. Choose the YAML button.
2. Choose **Edit**.
3. In the **ActionRoleArn** field, replace the IAM role ARN with the updated IAM role ARN. Choose **Validate**.
4. Choose **Commit**.

The workflow starts automatically if on the mainline branch. Otherwise, to rerun the workflow, choose **Run**.

How do I fill out a support form?

You can go to [Amazon CodeCatalyst](https://aws.amazon.com/codecatalyst/) or fill out a [Support Feedback form](https://aws.amazon.com/support/). In the **Request information** section, under **How can we help you**, include that you are an Amazon CodeCatalyst customer. Provide as much detail as possible so that we can most efficiently address your issue.
Extensions in CodeCatalyst

Amazon CodeCatalyst includes extensions that help you add functionality and integrate with products outside of CodeCatalyst. With extensions from the CodeCatalyst catalog, teams can customize their experiences in CodeCatalyst.

Topics
- Extensions concepts (p. 557)
- Installing and uninstalling extensions in CodeCatalyst (p. 557)
- Using GitHub repositories in CodeCatalyst (p. 558)
- Using Jira issues in CodeCatalyst (p. 568)

Extensions concepts

Here are some concepts and terms to know when working with extensions in CodeCatalyst.

Extensions

An extension is an add-on that you can install into your CodeCatalyst space to add new functionality to your projects and integrate with services outside of CodeCatalyst. Extensions can be browsed and installed from the CodeCatalyst catalog.

CodeCatalyst catalog

The CodeCatalyst catalog is a centralized listing of all the extensions available in CodeCatalyst. You can browse the CodeCatalyst catalog to find extensions that can improve your team's experiences in areas of CodeCatalyst such as source, workflows, and more.

Installing and uninstalling extensions in CodeCatalyst

An extension is an add-on that you can install into your CodeCatalyst space to add new functionality to your projects and integrate with services outside of CodeCatalyst. Extensions can be browsed and installed from the CodeCatalyst catalog.

Topics
- Installing an extension (p. 557)
- Uninstalling an extension (p. 558)

Installing an extension

You can install extensions for your CodeCatalyst space that add functionality to projects in that space.

You can view the CodeCatalyst catalog by choosing the Catalog icon.

557
Uninstalling an extension

Important
To install an extension, you must be signed in with an account that has the Space administrator role in the space.

To install an extension from the CodeCatalyst catalog

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Navigate to the CodeCatalyst catalog by choosing the Catalog icon in the top menu. You can search for extensions or filter extensions based on categories.
4. (Optional) Choose the name of the extension to see more details about the extension, such as the permissions the extension will have.
5. Choose Install. Review the permissions required by the extension, and if you want to continue, choose Install again.

After installing an extension, you will see the details page for the installed extension. Browse the tabs for more information about the extension. The details page is also where you will perform further configuration of the extension if needed.

Uninstalling an extension

You can uninstall extensions that were previously installed in your CodeCatalyst space. Uninstalling an extension may remove resources related to that extension from your CodeCatalyst space or projects.

Important
To uninstall an extension, you must be signed in with an account that has the Space administrator role in the space.

To uninstall an extension from your CodeCatalyst space

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view a list of the installed extensions for your space:
   a. Choose Settings, and then choose Installed extensions.
   b. Choose the Catalog icon in the top menu.
4. Choose Configure on the extension you want to uninstall.
5. Choose Uninstall on the extension details page.
6. Review the information in the Uninstall extension dialog box. Follow the instructions, and then choose Uninstall to uninstall the extension.

Using GitHub repositories in CodeCatalyst

GitHub is a cloud-based service that helps developers store and manage their code. The GitHub repositories extension lets you use linked GitHub repositories in Amazon CodeCatalyst projects. You can also link GitHub repositories when creating a new CodeCatalyst project. For more information, see Creating a project with a linked GitHub repository (p. 59).

Note
You can't use empty or archived GitHub repositories with CodeCatalyst projects. The GitHub repositories extension isn't compatible with GitHub Enterprise Server.
Once you install and configure the GitHub repositories extension, you will be able to:

- View your GitHub repositories in the list of source repositories in CodeCatalyst
- Store and manage workflow definition files in your GitHub repositories.
- Create, read, update, and delete files stored in linked GitHub repositories from CodeCatalyst Dev Environments
- Start CodeCatalyst workflow runs automatically when code is pushed to your GitHub repositories
- Use your linked GitHub repository source files in CodeCatalyst workflows
- Read and execute GitHub actions in CodeCatalyst workflows

The following topics explain how to install and configure the GitHub repositories extension and use your GitHub repositories once they are linked to your CodeCatalyst projects.

Topics

- Quickstart: Using GitHub repositories in CodeCatalyst (p. 559)
- Managing GitHub accounts in CodeCatalyst (p. 561)
- Managing GitHub repositories in CodeCatalyst (p. 563)
- Viewing linked GitHub repositories in CodeCatalyst (p. 565)
- Using GitHub repositories in CodeCatalyst workflows (p. 566)
- Using IP address access restrictions in GitHub Enterprise Cloud (p. 567)

Quickstart: Using GitHub repositories in CodeCatalyst

Perform the following steps to install the GitHub repositories extension, connect to your GitHub account, and link your GitHub repositories to your existing CodeCatalyst projects.

You can also install the GitHub repositories extension, connect to your GitHub account, link GitHub repositories when creating a new CodeCatalyst project. For more information, see Creating a project with a linked GitHub repository (p. 59).

Contents

- Step 1: Install the GitHub extension from the CodeCatalyst catalog (p. 559)
- Step 2: Connect your GitHub account to your CodeCatalyst space (p. 560)
- Step 3: Link your GitHub repository to your CodeCatalyst project (p. 560)
- Next steps (p. 561)

Step 1: Install the GitHub extension from the CodeCatalyst catalog

The first step to using GitHub repositories in CodeCatalyst is to install the GitHub repositories extension from the CodeCatalyst catalog. To install the extension, perform the following steps, choosing the GitHub repositories extension.

Important

As part of installing and configuring the GitHub repositories extension, you must install an extension into your GitHub account. To do this, you must be both the GitHub account administrator and the CodeCatalyst Space administrator.

To install an extension from the CodeCatalyst catalog

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.

3. Navigate to the CodeCatalyst CodeCatalyst catalog by choosing the Catalog icon in the top menu. You can search for GitHub repositories or filter extensions based on categories.

4. (Optional) To see more details about the extension, such as the permissions the extension will have, choose the GitHub repositories extension name.

5. Choose Install. Review the permissions required by the extension, and if you want to continue, choose Install again.

After installing the GitHub repositories extension, you are taken to the GitHub repositories extension details page, where you can view and manage connected GitHub accounts and linked GitHub repositories.

Step 2: Connect your GitHub account to your CodeCatalyst space

After you install the GitHub repositories extension, the next step is to connect your GitHub account to your CodeCatalyst space.

Important
To connect your GitHub account to your CodeCatalyst space, you must be both the GitHub account administrator and the CodeCatalyst Space administrator.

To connect your GitHub account to CodeCatalyst

1. In the Connected GitHub accounts tab, choose Connect GitHub account to go to the external site for GitHub.

2. Sign into your GitHub account using your GitHub credentials, and then choose the account where you want to install Amazon CodeCatalyst.

   Tip
   If you have previously connected your GitHub account to another space, you will not be prompted to re-authorize. You will instead see a dialog asking you where you would like to install the extension if you are a member or collaborator in more than one GitHub account, or the configuration page for the Amazon CodeCatalyst application if you only belong to one GitHub account. Configure the application for the repository access you want to allow, and then choose Save. If the Save button is not active in GitHub, make a change to the configuration, and then try again.

3. Choose whether you want to allow CodeCatalyst to access all current and future repositories, or choose the specific GitHub repositories you want to use in CodeCatalyst. The default option is all GitHub repositories in the GitHub account.

4. Review the permissions given to CodeCatalyst, and then choose Install.

After connecting to your GitHub account to CodeCatalyst, you can view the connected account in the GitHub accounts tab of the GitHub repositories extension details page.

Step 3: Link your GitHub repository to your CodeCatalyst project

The third and final step to using your GitHub repositories in CodeCatalyst is to link the repository to the CodeCatalyst project in which you want to use it.

Important
While you can link a GitHub repository as a Contributor, you can only unlink a GitHub repository as the Space administrator or the Project administrator. For more information, see Unlinking a GitHub repository from a CodeCatalyst project (p. 565).
To link a GitHub repository to a CodeCatalyst project from the GitHub repositories extension details page

1. In the Linked GitHub repositories tab, choose Link GitHub repository.
2. From the GitHub account drop-down menu, choose the GitHub account that contains the repository that you want to link.
3. From the GitHub repository drop-down menu, choose the repository you want to link to a CodeCatalyst project.
   
   **Tip**
   If the name of the repository is greyed out, you can't link that repository because it has already been linked to another project in the space.
4. From the CodeCatalyst project drop-down menu, choose the CodeCatalyst project you want to link the GitHub repository to.
5. Choose Link.

You can also link your GitHub repository to a project from Source repositories in Code. For more information, see Managing GitHub repositories in CodeCatalyst (p. 563).

Next steps

After installing the GitHub repositories extension, connecting your GitHub account, and linking your GitHub repositories to CodeCatalyst projects, you can use it in CodeCatalyst workflows and Dev Environments. For more information, see Using GitHub repositories in CodeCatalyst workflows (p. 566) and Creating a Dev Environment (p. 113).

Managing GitHub accounts in CodeCatalyst

To use GitHub repositories in CodeCatalyst, you must connect your GitHub account to your CodeCatalyst space. If you no longer want to use an account's repositories in CodeCatalyst, you can disconnect that GitHub account. When an account is disconnected, events in the account's repositories will not start workflow runs, and you will not be able to use those repositories with CodeCatalyst Dev Environments.

**Important**
To connect or disconnect your GitHub account and your CodeCatalyst space, you must be both the GitHub account administrator and the CodeCatalyst **Space administrator**.

Contents
- Connect a GitHub account to a CodeCatalyst space (p. 561)
- Connect a GitHub account to a CodeCatalyst space during project creation (p. 562)
- Disconnect a GitHub account from a CodeCatalyst space (p. 562)

Connect a GitHub account to a CodeCatalyst space

Perform the following steps to connect a GitHub account to your CodeCatalyst space.

**To connect your GitHub account to CodeCatalyst**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed **GitHub repositories** extension for your space:
   a. Choose **Settings**, and then choose **Installed extensions**.
b. Choose the Catalog icon in the top menu.

4. In GitHub repositories, choose Configure.

5. In the Connected GitHub accounts tab, choose Connect GitHub account to go to the external site for GitHub.

6. Sign into your GitHub account using your GitHub credentials, and then choose the account where you want to install Amazon CodeCatalyst.

**Tip**
If you have previously connected a GitHub account to the space, you will not be prompted to re-authorize. You will instead see a dialog asking you where you would like to install the extension if you are a member or collaborator in more than one GitHub space, or the configuration page for the Amazon CodeCatalyst application if you only belong to one GitHub space. Configure the application for the repository access you want to allow, and then choose Save. If the Save button is not active, make a change to the configuration, and then try again.

7. Choose whether you want to allow CodeCatalyst to access all current and future repositories, or choose the specific GitHub repositories you want to use in CodeCatalyst. The default option is all GitHub repositories in the GitHub space.

8. Review the permissions given to CodeCatalyst, and then choose Install.

After installing the GitHub repositories extension, you are taken to the GitHub repositories extension details page, where you can view and manage connected GitHub accounts and linked GitHub repositories.

**Connect a GitHub account to a CodeCatalyst space during project creation**

You can connect a GitHub account to your CodeCatalyst space when creating a new CodeCatalyst project. For more information, see *Creating a project with a linked GitHub repository* (p. 59).

**Disconnect a GitHub account from a CodeCatalyst space**

Perform the following steps to disconnect a GitHub account from your CodeCatalyst space. Once an account is disconnected, events in the account's repositories will not start workflow runs, and you will not be able to use those repositories with CodeCatalyst Dev Environments.

**Note**
To disconnect a GitHub account, you must first unlink all linked GitHub repositories from that account. For more information, see *Unlinking a GitHub repository from a CodeCatalyst project* (p. 565).

**To disconnect a GitHub account**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed GitHub repositories extension for your CodeCatalyst space:
   a. Choose Settings, and then choose Installed extensions.
   b. Choose the Catalog icon in the top menu.
4. In GitHub repositories, choose Configure.
5. In the **GitHub accounts** tab, choose the GitHub account you want to disconnect.
6. Choose **Disconnect GitHub account**.
7. In the **Disconnect** dialog box, review the effects of disconnecting the account.
8. Enter **disconnect** into the text input field, and then choose **Disconnect**.

**Managing GitHub repositories in CodeCatalyst**

To use GitHub repositories in CodeCatalyst, you must first link the repositories to your CodeCatalyst project. Before you can link a GitHub repository, you must connect the GitHub account that the repository belongs to with your CodeCatalyst space. For more information, see Connect a GitHub account to a CodeCatalyst space (p. 561).

If you no longer want to use GitHub repository in CodeCatalyst, you can unlink it from a CodeCatalyst project. When a repository is unlinked, events in that repository will not start workflow runs, and you will not be able to use that repository with CodeCatalyst Dev Environments.

**Contents**
- Link a GitHub repository to a CodeCatalyst project (p. 563)
- Link a GitHub repository to a CodeCatalyst project during project creation (p. 565)
- Unlinking a GitHub repository from a CodeCatalyst project (p. 565)

**Link a GitHub repository to a CodeCatalyst project**

You can use linked GitHub repositories in workflows, where events in the linked GitHub repository start workflows that might build, test, or deploy code, depending on the workflow configuration. Workflow configuration files for workflows that use linked GitHub repositories are stored in the linked GitHub repository. Linked GitHub repositories can also be used with Dev Environments to create, update, and delete files in the linked GitHub repository. You can link a GitHub repository to a CodeCatalyst project from either the details page of the GitHub repositories extension, or from the Source repositories view in Code in the project itself.

**Note**
A GitHub repository can only be linked to one CodeCatalyst project in a space. You can't link a repository that has the same name as another repository in the project.

**To link a GitHub repository to a CodeCatalyst project from the GitHub repositories extension details page**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed GitHub repositories extension for your space:
   a. Choose **Settings**, and then choose **Installed extensions**.
   b. Choose the Catalog icon in the top menu.
4. In GitHub repositories, choose **Configure**.
5. In the Linked GitHub repositories tab, choose **Link GitHub repository**.
6. From the GitHub account dropdown, choose the GitHub account that contains the repository that you want to link.
7. From the GitHub repository dropdown, choose the repository you want to link to a CodeCatalyst project.

563
Tip
If the name of the repository is greyed out, you can't link that repository because it has already been linked to another project in the space.

8. (Optional) If you don't see a GitHub repository in the list of repositories, it might not have been configured for repository access in the Amazon CodeCatalyst application in GitHub. You can configure which GitHub repositories can be used in CodeCatalyst in the connected account.
   a. Navigate to your GitHub account, choose Settings, and then choose Applications.
   b. In the Installed GitHub Apps tab, choose Configure for the Amazon CodeCatalyst application.
   c. Do one of the following to configure access of GitHub repositories you want to link in CodeCatalyst:
      - To provide access to all current and future repositories, choose All repositories.
      - To provide access to specific repositories, choose Only select repositories, choose the Select repositories dropdown, and then choose a repository you want to allow to link in CodeCatalyst.

Note
- You can't use empty or archived GitHub repositories with CodeCatalyst projects.
- You can't link a GitHub repository that has the same name as a repository in a CodeCatalyst project.
- The GitHub repositories extension isn't compatible with GitHub Enterprise Server repositories.

9. From the CodeCatalyst project dropdown menu, choose the CodeCatalyst project you want to link the GitHub repository to.

10. Choose Link.

To link a GitHub repository to a CodeCatalyst project from the source repositories page in a project
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst project.
3. In the navigation pane, choose Code, and then choose Source repositories.
4. Choose Add repository, and then choose Link repository.
5. From the Repository provider dropdown menu, choose GitHub.
6. From the GitHub account dropdown menu, choose the GitHub account that contains the repository that you want to link.
7. From the GitHub repository dropdown menu, choose the GitHub repository you want to link your CodeCatalyst project.

Tip
If the name of the repository is greyed out, you can't link that repository because it has already been linked to another project in the Amazon CodeCatalyst.

8. (Optional) If you don't see a GitHub repository in the GitHub repository dropdown, it might not have been configured for repository access in the space application in GitHub. You can configure which GitHub repositories can be used in CodeCatalyst in the connected account.
   a. Navigate to your GitHub account, choose Settings, and then choose Applications.
   b. In the Installed GitHub Apps tab, choose Configure for the Amazon CodeCatalyst application.
   c. Do one of the following to configure access of GitHub repositories you want to link in CodeCatalyst:
• To provide access to all current and future repositories, choose All repositories.
• To provide access to specific repositories, choose Only select repositories, choose the Select repositories dropdown, and then choose a repository you want to allow to link in CodeCatalyst.

Note
• You can't use empty or archived GitHub repositories with CodeCatalyst projects.
• You can't link a GitHub repository that has the same name as a repository in a CodeCatalyst project.
• The GitHub repositories extension isn't compatible with GitHub Enterprise Server repositories.

9. Choose Link.

Link a GitHub repository to a CodeCatalyst project during project creation

You can link a GitHub repository to a new CodeCatalyst project when creating the new CodeCatalyst project. For more information, see Creating a project with a linked GitHub repository (p. 59).

Unlinking a GitHub repository from a CodeCatalyst project

Unlinking a repository doesn't delete the GitHub repository or make any changes to it. It doesn't delete any workflow configuration files stored in that linked repository. However, once you unlink a GitHub repository, events in that repository will no longer start workflow runs, and you can't use the repository with Dev Environments.

Important
To unlink your GitHub repository from your CodeCatalyst project, you must be the Space administrator or the Project administrator.

To unlink a GitHub repository
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed GitHub repositories extension for your space:
   a. Choose Settings, and then choose Installed extensions.
   b. Choose the Catalog icon in the top menu.
4. In GitHub repositories, choose Configure.
5. In the Linked GitHub repositories tab, choose the GitHub repository you want to unlink.
6. Choose Unlink GitHub repository.
7. In the Unlink dialog box, review the effects of unlinking the repository.
8. Enter unlink into the text input field and choose Unlink.

Viewing linked GitHub repositories in CodeCatalyst

You can view the linked GitHub repositories in the list of source repositories for your project or from the GitHub repositories extension details page. Choosing them from the list of repositories doesn't open
them in CodeCatalyst. Instead, they open in GitHub, where you can view and work on the code in the linked repository.

**To view linked GitHub repositories in CodeCatalyst**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst project.
3. In the navigation pane, choose **Code**, and then choose **Source repositories**.

**To view linked GitHub repositories from you the extension details page**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space, and then choose the **Installed extensions** tab.
3. In **GitHub repositories**, choose **Configure**.
4. Choose the **Linked GitHub repositories** tab to view all GitHub repositories connected to CodeCatalyst projects in your CodeCatalyst space.

The GitHub repositories that are linked to your project are shown in the list. Choose the GitHub repository to view and edit files in GitHub.

**Note**
If a workflow uses a GitHub repository in a source action, changes you make to the workflow YAML in the visual editor or the YAML editor in CodeCatalyst will be automatically committed and pushed to the GitHub repository.

**Using GitHub repositories in CodeCatalyst workflows**

You can use a linked GitHub repository as the source for a workflow, where changes to a specified branch in a linked GitHub repository automatically start a workflow run.

A workflow is an automated procedure that describes how to build, test, and deploy your code as part of a continuous integration and continuous delivery (CI/CD) system. A workflow defines a series of steps, or **actions**, to take during a workflow run. A workflow also defines the events, or **triggers**, that cause the workflow to start. To set up a workflow, you create a **workflow definition file** using the CodeCatalyst console's **visual or YAML editor**.

**Tip**
For a quick look at how you might use workflows in a project, [create a project with a blueprint](#).
Each blueprint deploys a functioning workflow that you can review, run, and experiment with.

When you configure a workflow to use a linked GitHub repository, the workflow configuration file is stored in that GitHub repository. The workflow configuration is a YAML file that defines the workflow name, triggers, resources, artefacts, and actions. For more information about the workflow configuration file, see [Workflow definition reference](#).

The workflow configuration file must be in the `./codecatalyst/workflows/` directory in your GitHub repository.

You can use the workflow editor to create and configure workflows. For more information see [Getting started with workflows in CodeCatalyst](#) and [Working with sources](#).
Automatically start a workflow run after GitHub repository events

You can configure a CodeCatalyst workflow to automatically start a run when code is pushed to the specified branch of your GitHub repository. To start a workflow run automatically, add a trigger to the Triggers section of the workflow configuration file.

Example: A simple code push trigger

The following example shows a trigger that starts a workflow run whenever code is pushed to any branch in your source repository.

<table>
<thead>
<tr>
<th>Triggers:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Type: PUSH</td>
</tr>
</tbody>
</table>

Example: A simple pull request trigger

The following example shows a trigger that starts a workflow run whenever a pull request is created against any branch in your source repository.

<table>
<thead>
<tr>
<th>Triggers:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Type: PULLREQUEST</td>
</tr>
<tr>
<td>Events:</td>
</tr>
<tr>
<td>- OPEN</td>
</tr>
</tbody>
</table>

For more information, see Working with triggers (p. 327).

Using IP address access restrictions in GitHub Enterprise Cloud

The Amazon CodeCatalyst GitHub repositories extension is compatible with GitHub Enterprise Cloud IP access restrictions. When configuring a GitHub Enterprise Cloud organization to restrict access to specific IP addresses, you can also enable GitHub apps to configure the allow list, which will let CodeCatalyst register its IP addresses automatically with GitHub. Alternatively, you can manually add the CodeCatalyst IP addresses. For more information, see IP addresses used by GitHub repositories extension (p. 567).

If the CodeCatalyst IP addresses aren't in a GitHub organization's allow list, the Amazon CodeCatalyst GitHub app won't be able to access your GitHub repositories.

IP addresses used by GitHub repositories extension

The following IP addresses are used by the GitHub repositories extension to access your GitHub organization's resources:

<table>
<thead>
<tr>
<th>IP addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>us-west-2</td>
</tr>
<tr>
<td>52.32.242.246</td>
</tr>
<tr>
<td>54.148.176.49</td>
</tr>
<tr>
<td>35.164.118.94</td>
</tr>
<tr>
<td>eu-west-1</td>
</tr>
<tr>
<td>34.241.64.10</td>
</tr>
<tr>
<td>34.246.255.80</td>
</tr>
<tr>
<td>3.248.38.7</td>
</tr>
</tbody>
</table>
Using Jira issues in CodeCatalyst

Jira is a software application that helps agile development teams plan, assign, track, report, and manage work. The Jira Software extension lets you use Jira projects in Amazon CodeCatalyst projects.

Note
CodeCatalyst is only compatible with Jira Software Cloud.

Once you install and configure the Jira Software extension for an Amazon CodeCatalyst project, you will be able to:

1. Access Jira projects from CodeCatalyst by linking them to CodeCatalyst projects
2. Update Jira issues with CodeCatalyst pull requests
3. View status and workflow runs of linked CodeCatalyst pull requests in Jira issues

The following topics explain how to install and configure the Jira Software extension to use Jira in CodeCatalyst.

Topics
- Quickstart: Using Jira issues in CodeCatalyst (p. 568)
- Managing Jira sites in CodeCatalyst (p. 571)
- Managing Jira projects in CodeCatalyst (p. 572)
- Linking a Jira issue to a CodeCatalyst pull request (p. 574)
- Viewing CodeCatalyst events in Jira (p. 574)
- Searching Jira issues in CodeCatalyst (p. 575)

Quickstart: Using Jira issues in CodeCatalyst

Perform the following steps to install the Jira Software extension, connect your Jira sites to your CodeCatalyst spaces, and link your Jira projects to your CodeCatalyst projects.

Contents
- Step 1: Install the Jira Software extension from the CodeCatalyst catalog (p. 568)
- Step 2: Connect your Jira sites to your CodeCatalyst space (p. 569)
- Step 3: Link your Jira project to your CodeCatalyst project (p. 569)
- Step 4: Linking your Jira issues to a CodeCatalyst pull request (p. 570)
- Next steps (p. 571)

Step 1: Install the Jira Software extension from the CodeCatalyst catalog

The first step to managing Jira in CodeCatalyst is to install the Jira Software extension from the CodeCatalyst catalog. To install the extension, perform the following steps, choosing the Jira Software extension.

Important
As part of installing and configuring the Jira Software extension, you must have an Atlassian account and an existing Jira site, but a Jira project can be created at a later time. You must be the Jira site administrator and the CodeCatalyst Space administrator.
To install an extension from the CodeCatalyst catalog

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Navigate to the CodeCatalyst CodeCatalyst catalog by choosing the Catalog icon in the top menu. You can search for Jira Software or filter extensions based on categories.
4. (Optional) To see more details about the extension, such as the permissions the extension will have, choose the Jira Software extension name.
5. Choose Install. Review the permissions required by the extension, and if you want to continue, choose Install again.

After installing the Jira Software extension, you are taken to the Jira Software extension details page, where you can view and manage connected Jira sites and linked Jira projects.

Step 2: Connect your Jira sites to your CodeCatalyst space

After you install the Jira Software extension, the next step is to connect your Jira site to your CodeCatalyst space.

Important
To connect your Jira site to your CodeCatalyst space, you must be both the Jira site administrator and the CodeCatalyst Space administrator.

To connect your Jira site to CodeCatalyst

1. In the Connected Jira sites tab, choose Connect Jira site to go to the external site for Atlassian Marketplace.
2. Choose Get it now to get started with installing CodeCatalyst on your Jira site.
3. Depending on your role, do one of the following:
   1. If you are a Jira site administrator, from the site drop-down menu, choose the Jira site to install the CodeCatalyst application, and then choose Install app.
      
      Note
      If you have one Jira site, this step won't appear, and you'll automatically be directed to the next step.
   2. a. If you aren't a Jira administrator, from the site drop-down menu, choose the Jira site to install the CodeCatalyst application, and then choose Request app. For more information on installing Jira apps, see Who can install apps?
   
   b. Enter the reason you need to install CodeCatalyst into the input text field or keep the default text, and then choose Submit request.
4. Review the actions performed by CodeCatalyst when the application is installed, and then choose Get it now.
5. After the application is installed, choose Return to CodeCatalyst to return to CodeCatalyst.

After connecting your Jira site to CodeCatalyst, you can view the connected site in the Connected Jira sites tab of the Jira Software extension details page.

Step 3: Link your Jira project to your CodeCatalyst project

The third step to managing your Jira project in CodeCatalyst is to link the Jira project to the CodeCatalyst project in which you want to use it.
Note
A CodeCatalyst project can only be linked to one Jira project. A Jira project can be linked to multiple CodeCatalyst projects.

Important
To link your Jira project to your CodeCatalyst project, you must be the CodeCatalyst Space administrator or CodeCatalyst Project administrator.

To link a Jira project to a CodeCatalyst project from the Jira Software extension details page

1. In the Linked Jira projects tab, choose Link Jira project.
2. From the Jira site drop-down menu, choose the Jira site that contains the project that you want to link.
3. From the Jira project drop-down menu, choose the project you want to link to a CodeCatalyst project.
4. From the CodeCatalyst project drop-down menu, choose the CodeCatalyst project you want to link to a Jira project.
5. Choose Link.

Once a Jira project is linked to a CodeCatalyst project, access to CodeCatalyst issues is disabled entirely, and Issues in the CodeCatalyst navigation pane will be replaced with a Jira issues item that links to the Jira project.

Step 4: Linking your Jira issues to a CodeCatalyst pull request

After linking a Jira project to CodeCatalyst project, you can link CodeCatalyst issues by creating a pull request and have them displayed as a property of the pull request.

Note
You can't create a pull request without a source repository with two branches in your CodeCatalyst project. For more information on pull requests, see Working with pull requests in CodeCatalyst (p. 100).

To link Jira issues to a CodeCatalyst pull request

1. In the navigation pane, choose Code, and then choose Pull requests.
2. Choose Create pull request to enter the pull request details.
3. From the Source repository drop-down menu, choose the source repository in which you want to link a pull request.
4. From the Source branch drop-down menu, choose the branch that contains the changes you want reviewed.
5. From the Destination branch drop-down menu, choose the branch where you want to merge reviewed changes.
6. In the Pull request title text input field, enter the title of your pull requests.
7. Choose Link issues for the Jira issues - optional field, choose the drop-down, and search the Jira issues you want to add from the linked Jira project.
8. Select the Jira issues you want to add to the pull request.
9. Choose Create to create the pull request.

The summary status of the pull request and the status of associated CodeCatalyst workflow events are reflected in your Jira issue.
Next steps

After installing the **Jira Software** extension, connecting your Jira site, linking your Jira projects to your CodeCatalyst project, and linking a pull request, updates from CodeCatalyst are reflected in your Jira project. For more information on viewing CodeCatalyst events in Jira, see [Viewing CodeCatalyst events in Jira](p. 574).

Managing Jira sites in CodeCatalyst

To manage Jira projects in CodeCatalyst, you must connect your Jira sites to your CodeCatalyst space. If you no longer want to use a Jira site's projects in CodeCatalyst, you can disconnect that Jira site. When a Jira site is disconnected, Jira issues from the site's projects will not be available in the CodeCatalyst projects, and CodeCatalyst Issues will be the issue provider again.

**Important**

To connect or disconnect your Jira site and your CodeCatalyst space, you must be both the Jira site administrator and the CodeCatalyst Space administrator.

Contents

- Connecting a Jira site to a CodeCatalyst space (p. 571)
- Disconnecting a Jira site from a CodeCatalyst space (p. 572)

Connecting a Jira site to a CodeCatalyst space

Perform the following steps to connect a Jira site to your CodeCatalyst space.

**To connect your Jira site to CodeCatalyst**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed **Jira Software** extension for your space:
   a. Choose **Settings**, and then choose **Installed extensions**.
   b. Choose the Catalog icon in the top menu.
4. In **Jira Software**, choose **Configure**.
5. In the **Connected Jira sites** tab, choose **Connect Jira site** to go to the external site for Atlassian Marketplace.
6. Choose **Get it now** to get started with installing CodeCatalyst on your Jira site.
   
   **Note**
   
   If you previously installed CodeCatalyst to your Jira site, you will be notified. Choose **Get started** to be taken to the final step.
7. Depending on your role, do one of the following:

   1. If you are a Jira site administrator, from the site drop-down menu, choose the Jira site to install the CodeCatalyst application, and then choose **Install app**.
      
      **Note**
      
      If you have one Jira site, this step won't appear, and you'll automatically be directed to the next step.

   2. a. If you aren't a Jira administrator, from the site drop-down menu, choose the Jira site to install the CodeCatalyst application, and then choose **Request app**. For more information on installing Jira apps, see [Who can install apps?](p. 571).
b. Enter the reason you need to install CodeCatalyst into the input text field or keep the default text, and then choose **Submit request**.

8. Review the actions performed by CodeCatalyst when the application is installed, and then choose **Get it now**.

9. After the application is installed, choose **Return to CodeCatalyst** to return to CodeCatalyst.

After connecting your Jira site to CodeCatalyst, you can view the connected site in the **Connected Jira sites** tab of the **Jira Software** extension details page.

### Disconnecting a Jira site from a CodeCatalyst space

Perform the following steps to disconnect a Jira site from your CodeCatalyst space.

**Note**
To disconnect a Jira site, you must first unlink all linked Jira projects from that account. For more information, see **Unlinking a Jira project from a CodeCatalyst project (p. 573)**.

**To disconnect a Jira site**

1. Open the CodeCatalyst console at **https://codecatalyst.aws/**.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed **Jira Software** extension for your space:
   a. Choose **Settings**, and then choose **Installed extensions**.
   b. Choose the **Catalog** icon in the top menu.
4. In **Jira Software**, choose **Configure**.
5. In the **Connected Jira sites** tab, choose the Jira site that you want to disconnect.
6. Choose **Disconnect Jira site**.
7. In the **Disconnect** dialog box, review the effects of disconnecting the site.
8. Enter **disconnect** into the text input field, and then choose **Disconnect**.

### Managing Jira projects in CodeCatalyst

To manage a Jira project in CodeCatalyst, you must first link the Jira project to your CodeCatalyst project. Then, you can add and use Jira issues to a CodeCatalyst pull request in a CodeCatalyst project. Before you can link a Jira project, you must connect the Jira site that the project belongs to with your CodeCatalyst space. For more information, see **Connecting a Jira site to a CodeCatalyst space (p. 571)**.

If you no longer want to use a Jira project in CodeCatalyst, you can unlink it from your CodeCatalyst project. When a Jira project is unlinked, Jira issues will not be available in the CodeCatalyst project, and CodeCatalyst **Issues** will be the issue provider again.

**Important**
To link your Jira project to your CodeCatalyst project, you must be the CodeCatalyst **Space administrator** or CodeCatalyst **Project administrator**.

**Contents**

- **Linking a Jira project to a CodeCatalyst project (p. 573)**
- **Unlinking a Jira project from a CodeCatalyst project (p. 573)**

572
Linking a Jira project to a CodeCatalyst project

You can use linked Jira projects to manage issues and to link CodeCatalyst pull requests to a Jira issue. Summary status of a pull request and the status of associated CodeCatalyst workflow events are reflected in your Jira issue.

**Note**
A CodeCatalyst project can only be linked to one Jira project. A Jira project can be linked to multiple CodeCatalyst projects.

**To link a Jira project to a CodeCatalyst project from the Jira Software extension details page**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed Jira Software extension for your space:
   a. Choose Settings, and then choose Installed extensions.
   b. Choose the Catalog icon in the top menu.
5. In the Linked Jira projects tab, choose Link Jira project.
6. From the Jira site drop-down menu, choose the Jira site that contains the project that you want to link.
7. From the Jira project drop-down menu, choose the project you want to link to a CodeCatalyst project.
8. From the CodeCatalyst project drop-down menu, choose the CodeCatalyst project you want to link to a Jira project.
9. Choose Link.

Once a Jira project is linked to a CodeCatalyst project, access to CodeCatalyst issues is disabled entirely, and Issues in the CodeCatalyst navigation pane will be replaced with a Jira issues item that links to the Jira project.

Unlinking a Jira project from a CodeCatalyst project

Unlinking a project doesn't delete the Jira project, including planning items or development information, or make any changes to it. However, once you unlink a Jira project, the project's Jira issues will no longer be available to link to the CodeCatalyst project.

**To unlink a Jira project**

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space.
3. Do one of the following to view the installed Jira Software extension for your space:
   a. Choose Settings, and then choose Installed extensions.
   b. Choose the Catalog icon in the top menu.
4. In the Linked Jira projects tab, choose the Jira project you want to unlink.
5. Choose Unlink Jira project.
6. In the Unlink dialog box, review the effects of unlinking the repository.
Linking Jira issues to pull requests

You can link pull requests that are created in a CodeCatalyst source repository to Jira issues. After linking a Jira issue, the issue is displayed as a property of the pull request. As a result, pull request events, workflow events, and deployment events are sent to Jira and added to the Jira issue. Pull requests can be linked to one or more Jira issues. You can only link pull requests that are in a CodeCatalyst source repository, not those in a third-party repository like GitHub. Before you can link Jira issues to a pull request, your Jira project must be linked to the CodeCatalyst project. For more information about linking a Jira project to a CodeCatalyst project, see Managing Jira sites in CodeCatalyst (p. 572).

Note
You can't create a pull request without a source repository with two branches in your CodeCatalyst project. For more information on pull requests, see Working with pull requests in CodeCatalyst (p. 100).

To link Jira issues to a CodeCatalyst pull request

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst project.
3. In the navigation pane, choose Code, and then choose Pull requests.
4. Choose Create pull request to enter the pull request details.
5. From the Source repository drop-down menu, choose the source repository in which you want to link a pull request.
6. From the Source branch drop-down menu, choose the branch that contains the changes you want reviewed.
7. From the Destination branch drop-down menu, choose the branch where you want to merge reviewed changes.
8. In the Pull request title text input field, enter the title of your pull requests.
9. Choose Link issues for the Jira issues - optional field, choose the drop-down, and search the Jira issues you want to add from the linked Jira project.
10. Select the Jira issues you want to add to the pull request.
11. Choose Create to create the pull request.

Once you link Jira issues to a CodeCatalyst pull request, a summary of the pull request is available. The summary includes workflow runs, linked issues, required reviewers, optional reviewers, and the author.

Note
Assignee and Created by information associated with the Jira issue is not available in CodeCatalyst.

After linking a pull request, the synced CodeCatalyst project and Jira project allow updates from CodeCatalyst to be reflected in your Jira project. The status of the linked pull request and any workflow events related to the pull request will show up in the Jira issue when viewing it in Jira. For more information on viewing CodeCatalyst events in Jira, see Viewing CodeCatalyst events in Jira (p. 574).

Viewing CodeCatalyst events in Jira

If your CodeCatalyst projects and Jira projects are linked, the summary status of the pull request and the status of associated CodeCatalyst workflow events are reflected in your Jira issue. For example, if you close or merge a pull request in CodeCatalyst, the status update is reflected in the Jira issue. CodeCatalyst workflow CI/CD events related to a CodeCatalyst pull request are synchronized, so a successful workflow run would be sent to the Jira issue as well.
To view CodeCatalyst events in a Jira issue

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst project.
3. In the CodeCatalyst navigation pane, choose Code, choose Pull requests, and then choose the pull request with the Jira issue that you want to view in your Jira project.
4. In the Additional info pane, choose the Jira issue that you want to view in your Jira project.
5. From the Details pane in the Jira project, choose pull request listed for Development to see details of the pull request.
6. (Optional) To see the latest builds, choose the Builds tab.
7. (Optional) To see the development status, choose the Deployments tab.

Searching Jira issues in CodeCatalyst

After linking a Jira project, you can search the linked Jira project for issues using the CodeCatalyst global search bar. You can also search for Jira issues in CodeCatalyst while linking to the issues from a pull request. For more information about linking Jira issues to a CodeCatalyst pull request, see Linking a Jira issue to a CodeCatalyst pull request (p. 574).

To search for Jira issues in linked Jira projects

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst project.
3. In the global search bar, search a linked Jira project for issues or Jira issues you want to link to a pull request.
Search in CodeCatalyst

Use the search bar or a dedicated search results window in CodeCatalyst to search through code, issues, projects, and users CodeCatalyst.

You can find resources across your space and projects by entering queries such as name, description, and status into the search bar. You can also refine your search queries using the search query language.

Topics
- Refining your search query (p. 576)
- Considerations when working with search (p. 578)
- Searchable fields reference (p. 578)

To search
1. In the search bar in the top navigation bar, enter a search query.
2. (Optional) Refine your search query using CodeCatalyst's search query language. For more information, see Refining your search query (p. 576).
3. Do one of the following:
   - To search for resources within the project that you are currently in, choose This project.
   - To search for resources within all of the projects in the space you are currently in, choose This space.
4. View search results in a dedicated search results window by doing one of the following:
   - In the bottom of the quick search results window, choose View all results in project-name | space-name to view all search results.
   - Press Enter to view all search results.

Tip
Mention other project users in a pull request comment or description, or in an issue comment or description, by using the @ sign followed by their display name or user name. You can also link to resources like issues or code files by using the @ sign followed by the name of the issue or code file.

Refining your search query

If you can't find what you're looking for after searching, you can refine your search with CodeCatalyst's specialized query language. Individual fields have no character limit, but the overall query has a limit of 1,024 characters.

Topics
- Refining by type (p. 577)
- Refining by field (p. 577)
Refining by type

To refine the scope of your search to a specific type of information, include `type:result-type` in your search, where `result-type` is code, issue, project, or user.

Examples:
- `type:code AND java` – Show code results in code-related fields that contain “java”.
  
  For more information, see Code fields (p. 578).
- `type:issue AND Bug` – Show issue results in issue-related fields that contain “Bug”.
  
  For more information, see Issue fields (p. 579).
- `type:user AND MaryMajor` – Show user results in user-related fields that contain “MaryMajor”.
  
  For more information, see User fields (p. 581).
- `type:project AND Datafeeder` – Show project results that contain “Datafeeder”.
  
  For more information, see Project fields (p. 581).

Refining by field

To refine the scope of your search to a specific field, include `field-name:query` in your search, where `field-name` is title, username, project, description, and so on, and `query` is the text for which you are searching. For a list of fields, see Searchable fields reference (p. 578). You can search for multiple queries using parentheses.

Examples:
- `title:bug` – Show results where the title contains “bug”.
- `username:John` – Show results where the user name contains “John”.
- `project:DataFeeder` – Show results in the project “DataFeeder”. Query isn't case sensitive.
- `description:overview` – Show results where the description contains “overview”.

Refining with Boolean operators

To specify constraints on search phrases, you can use the Boolean operators AND, OR, and NOT. If you list multiple phrases, CodeCatalyst joins them with OR by default. You can group search phrases using parentheses.

Examples:
- `exception AND type:code` – Show only code results for “exception”.
- `path:README.md AND repo:ServerlessAPI` – Show results for paths with “README.md” where the repository is named “ServerlessAPI”.
- `buildspec.yml AND (repo:ServerlessAPI OR ServerlessWebApp)` – Show results for “buildspec.yml” where the repository is “ServerlessAPI” or “ServerlessWebApp”.
- `path:java NOT (path:py OR path:ts)` – Show results where the path contains “java” but not “py” or “ts”.

577
Refining by project

To refine the scope of your search to a specific project, include `project:name AND query` in your search, where `name` is the project within which you are searching and `query` is the content for which you are searching.

- `project:name AND query` – Show results where the path contains the query and the project name.

Considerations when working with search

**Delayed content updates** – It can take several minutes for content updates, such as name changes or issue reassignments, to be reflected in the search results. Large updates, such as a code base migration, can take longer to appear in search results.

**Escaping special characters** – The following special characters require special consideration in your search queries: `+ - & | ( ) { } [ ] ^ " ~ * : \\`. Special characters will not influence the query, and you must either remove them or escape them. To escape a character, add a backslash (`\`) in front of it. For example, the search query `Feature` should either be `Feature` or `\Feature`.

**Narrowing search** – Search isn't case sensitive. Searching in all lowercase prevents your queries from splitting up words on case change. For example, to query for `MyService` and only `MyService`, consider querying `myservice` to avoid results that contain only `my` or `service`.

Search joins words and parts of words with OR-wise conjunction by default. For example, `new function` could return results containing both `new` and `function` and also results with only `new` or `function`. To avoid the latter, combine multiple words with AND. For example, you can search `new AND function`.

**Default branches** – Search will only return code results from the latest commit on a source repository's default branch. To find code on other branches or commits, consider cloning the repository locally (p. 92), opening the branch in a Dev Environment (p. 113), or viewing the branches and details in the CodeCatalyst UI (p. 97). Changing the default branch results in updates to the files discoverable by search. For more information, see View and change the default branch for a repository (p. 95).

Searchable fields reference

CodeCatalyst searches the following fields when you enter search queries. Aliases are another name that you can use to reference the field in the advanced query language.

### Code fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>branchName</td>
<td>branch</td>
<td>Name of branch the code file is on.</td>
</tr>
<tr>
<td>code</td>
<td>N/A</td>
<td>Information about the code contents in the form of code snippets indicating parts of the source code that matched the search.</td>
</tr>
<tr>
<td>commitId</td>
<td>N/A</td>
<td>Commit ID of the commit in which the returned code file was last updated. May or may not</td>
</tr>
<tr>
<td>Field</td>
<td>Alias</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>be the commit ID at the tip of the branch name specified in branchName.</td>
</tr>
<tr>
<td>commitMessage</td>
<td>N/A</td>
<td>Commit message of the commit in which the code file was last updated. May or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>may not be the commit message at the tip of the branch name specified in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>branchName. If no commit message was provided, this value will be an empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>string.</td>
</tr>
<tr>
<td>filePath</td>
<td>path</td>
<td>File path of this code file.</td>
</tr>
<tr>
<td>lastUpdatedBy</td>
<td>N/A</td>
<td>CodeCatalyst user who last updated the code file. If the user name isn't</td>
</tr>
<tr>
<td></td>
<td></td>
<td>available, this value will be the email address of the user as configured in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the Git configuration file.</td>
</tr>
<tr>
<td>lastUpdatedById</td>
<td>N/A</td>
<td>System-generated unique ID of user that last updated the code file. If the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user ID isn't available, this value might be the email address of the user.</td>
</tr>
<tr>
<td>lastUpdatedTime</td>
<td>N/A</td>
<td>Time when the search data was last updated with the commit that contained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the code file (in coordinated universal time (UTC) timestamp).</td>
</tr>
<tr>
<td>projectId</td>
<td>N/A</td>
<td>System-generated unique ID of the project.</td>
</tr>
<tr>
<td>projectName</td>
<td>projectNames, project</td>
<td>Display name of the project that contains the source repository where the code file has been committed.</td>
</tr>
<tr>
<td>repositoryId</td>
<td>repold</td>
<td>System-generated unique ID of the source repository.</td>
</tr>
<tr>
<td>repositoryName</td>
<td>repository, repo</td>
<td>Display name of the source repository where the code file has been committed.</td>
</tr>
</tbody>
</table>

### Issue fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assigneeIds</td>
<td>assigneeId</td>
<td>System-generated unique IDs of the users assigned to the issue.</td>
</tr>
<tr>
<td>Field</td>
<td>Alias</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>assignees</td>
<td>assignee</td>
<td>User names of the users assigned to the issue.</td>
</tr>
<tr>
<td>createdBy</td>
<td>N/A</td>
<td>Display name of the user who created the issue.</td>
</tr>
<tr>
<td>createdById</td>
<td>N/A</td>
<td>System-generated unique ID of the user who created the issue.</td>
</tr>
<tr>
<td>createdTime</td>
<td>N/A</td>
<td>Time the issue was created (in coordinated universal time (UTC) timestamp).</td>
</tr>
<tr>
<td>description</td>
<td>N/A</td>
<td>Description of the issue.</td>
</tr>
<tr>
<td>isArchived</td>
<td>archived</td>
<td>Boolean value that indicates whether to create the issue in an archived state.</td>
</tr>
<tr>
<td>isBlocked</td>
<td>blocked</td>
<td>Boolean value that indicates whether the issue is marked as blocked.</td>
</tr>
<tr>
<td>labelIds</td>
<td>labelId</td>
<td>System-generated unique IDs of the labels for an issue.</td>
</tr>
<tr>
<td>lastUpdatedBy</td>
<td>N/A</td>
<td>Display name of user who last updated the issue.</td>
</tr>
<tr>
<td>lastUpdatedById</td>
<td>N/A</td>
<td>System-generated unique ID of the user who last updated the issue.</td>
</tr>
<tr>
<td>lastUpdatedTime</td>
<td>N/A</td>
<td>Time the issue was last updated (in coordinated universal time (UTC) timestamp).</td>
</tr>
<tr>
<td>priority</td>
<td>N/A</td>
<td>Priority of the issue, if one has been assigned.</td>
</tr>
<tr>
<td>projectId</td>
<td>N/A</td>
<td>System-generated unique ID of the project.</td>
</tr>
<tr>
<td>projectName</td>
<td>projectNames, project</td>
<td>Project in which this issue can be found.</td>
</tr>
<tr>
<td>shortId</td>
<td>N/A</td>
<td>Shortened, auto-incrementing identifier for the issue.</td>
</tr>
<tr>
<td>status</td>
<td>N/A</td>
<td>Status of the issue that indicates if issue is in backlog or column on board.</td>
</tr>
<tr>
<td>statusId</td>
<td>N/A</td>
<td>System identifier of the status.</td>
</tr>
<tr>
<td>title</td>
<td>N/A</td>
<td>Title of the issue.</td>
</tr>
</tbody>
</table>
## Project fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>N/A</td>
<td>Description of the project.</td>
</tr>
<tr>
<td>lastUpdatedTime</td>
<td>N/A</td>
<td>Time when the project metadata was last updated (in coordinated universal time (UTC) timestamp).</td>
</tr>
<tr>
<td>projectName</td>
<td>project</td>
<td>Name of the project in the space.</td>
</tr>
<tr>
<td>projectPath</td>
<td>N/A</td>
<td>URL-routable name of the project, defined during project creation. Used in URLs that require the project name.</td>
</tr>
</tbody>
</table>

## User fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayName</td>
<td>N/A</td>
<td>Name used for the user in CodeCatalyst. Display names are not unique.</td>
</tr>
<tr>
<td>email</td>
<td>N/A</td>
<td>Email address of the user.</td>
</tr>
<tr>
<td>lastUpdatedTime</td>
<td>N/A</td>
<td>Time when the user metadata was last updated (in coordinated universal time (UTC) timestamp).</td>
</tr>
<tr>
<td>userName</td>
<td>username</td>
<td>User name chosen by the user when they signed up for CodeCatalyst. Unlike display names, user names can't be changed.</td>
</tr>
</tbody>
</table>
Troubleshooting Amazon CodeCatalyst

The following information can help you troubleshoot common issues in CodeCatalyst. You can also use the Amazon CodeCatalyst health report to determine if there are service issues that might be impacting your experience.

Topics
- Troubleshooting general access issues (p. 582)
- Troubleshooting support issues (p. 583)
- Some or all of Amazon CodeCatalyst isn't available (p. 582)
- I can't create a project in CodeCatalyst (p. 583)
- I want to submit feedback in CodeCatalyst (p. 583)
- Troubleshooting problems with source repositories (p. 585)
- Troubleshooting problems with workflows (p. 588)
- Troubleshooting problems with search in CodeCatalyst (p. 598)
- Troubleshooting problems with accounts associated with your space (p. 599)
- Troubleshooting problems with Dev Environments (p. 602)
- Troubleshooting problems with issues (p. 605)
- Troubleshooting problems between Amazon CodeCatalyst and the AWS SDKs or the AWS CLI (p. 605)

Troubleshooting general access issues

I forgot my password

**Problem:** I forgot the password I use for my AWS Builder ID and Amazon CodeCatalyst.

**Possible fixes:** The easiest way to fix this problem is to reset your password.

1. Open Amazon CodeCatalyst and enter your **Email address**. Then, choose **Continue**.
2. Choose **Forgot password?**
3. We'll send you an email with a link for you to change your password. If you don't see the email in your inbox, check your spam folder.

Some or all of Amazon CodeCatalyst isn't available

**Problem:** I navigated to or followed a link to the CodeCatalyst console, but I see an error.

**Possible fixes:** The most common reasons for this problem are that you either followed a link to a project or a space you haven't been invited to, or there is a general availability issue with the service. Check the **Health report** (p. 607) to see if there are any known issues with the service. If not, contact...
I can't create a project in CodeCatalyst

**Problem:** I want to create a project, but the **Create project** button shows as unavailable, or I receive an error message.

**Possible fixes:** The most common reasons for this problem are that you are signed in to the console with an AWS Builder ID that doesn't have the **Space administrator** role. You must have this role to create projects in a space.

If you do have this role and the button does not appear as available, there might be a transitory issue with the service. Refresh your browser and try again.

Troubleshooting support issues

I get an error when I access AWS Support for Amazon CodeCatalyst

**Problem:** When I choose the AWS Support for Amazon CodeCatalyst option, I receive the following error message:

**Unable to assume role**

To access support cases, you must add the role **AWSRoleForCodeCatalystSupport** to the AWS account that is the billing account for the space.

**Possible fixes:** Add the required role to the AWS account that is the billing account for the space. The account designated as the billing account for the space uses the **AWSRoleForCodeCatalystSupport** role and **AmazonCodeCatalystSupportAccess** managed policy. For more information, see [Creating the AWSRoleForCodeCatalystSupport role for your account and space](p. 518).

**Note**

An AWS Builder ID can only get support for the alias they are authenticated with and only for resources based on permissions in CodeCatalyst. Account and Billing support is available for all users in the space. However, builders can only get support for resources and information they have permissions for in CodeCatalyst.

I cannot create technical support cases for my space

**Problem:** I cannot create technical support cases for my space.

**Fixes:** A Business Support or Enterprise Support plan needs to be added to the space billing account in order for users in the space to create technical support cases. Ask your space administrator to add an AWS Support plan to your space billing account or visit https://repost.aws/ to ask the AWS community.

My account for support cases is no longer connected to my space in CodeCatalyst

**Problem:** My account for support cases is no longer connected to my space in CodeCatalyst.
Fixes: If a user with the **Space administrator** role switches the space billing account, this will disconnect the AWS Support plan and all associated cases from the space. The AWS Support cases associated with the old space billing account will no longer be visible in AWS Support for Amazon CodeCatalyst. The root user for that billing account can view and resolve old cases from the AWS Management Console and can set up IAM permissions for AWS Support for other users to view and resolve old cases. You will not be able to continue to get technical support for CodeCatalyst from the old space billing account through the AWS Management Console, but you can receive technical support for other services until your AWS Support plan is canceled.

For more information, see [Updating, resolving, and reopening your case](https://docs.aws.amazon.com/sUPPORT/latest/ug/updated_case.html) in the *AWS Support User Guide*.

**I can't open a support case for another AWS service in AWS Support for Amazon CodeCatalyst**

**Problem:** I can't open a support case for another AWS service in AWS Support for CodeCatalyst.

**Possible fixes:** You can only open CodeCatalyst support cases from AWS Support for CodeCatalyst. If you need support for services or resources deployed from CodeCatalyst to another AWS, Amazon, or other third-party service, you will need to create a case through the AWS Management Console or the third-party service support channel. For more information, see [Creating support cases and case management](https://docs.aws.amazon.com/sUPPORT/latest/ug/support_manualAGIC-2019.html) in the *AWS Support User Guide*.

**Some or all of Amazon CodeCatalyst isn't available**

**Problem:** I navigated to or followed a link to the CodeCatalyst console, but I see an error.

**Possible fixes:** The most common reasons for this problem are that you either followed a link to a project or a space you haven't been invited to, or there is a general availability issue with the service. Check the Health report (p. 607) to see if there are any known issues with the service. If not, contact the person who invited you to the project or space and ask for another invitation. If you haven't been invited to any projects or spaces, you can sign up and [create your own space and projects](https://docs.aws.amazon.com/codecatalyst/latest/userguide/sign-up.html) (p. 8).

**I can't create a project in CodeCatalyst**

**Problem:** I want to create a project, but the **Create project** button shows as unavailable, or I receive an error message.

**Possible fixes:** The most common reasons for this problem are that you are signed in to the console with an AWS Builder ID that doesn't have the **Space administrator** role. You must have this role to create projects in a space.

If you do have this role and the button does not appear as available, there might be a transitory issue with the service. Refresh your browser and try again.

**I want to submit feedback in CodeCatalyst**

**Problem:** I found a bug in CodeCatalyst and I want to submit feedback.

**Possible fixes:** You can submit feedback directly in CodeCatalyst.
Troubleshooting problems with source repositories

The following information can help you troubleshoot common issues with source repositories in CodeCatalyst.

**Topics**
- I have reached the maximum storage for my space and see warnings or errors (p. 585)
- I receive an error when trying to clone or push to an Amazon CodeCatalyst source repository (p. 585)
- I need a source repository for my project (p. 586)
- My source repository is brand-new but contains a commit (p. 586)
- I want a different branch as my default branch (p. 586)
- I am receiving emails about activity in pull requests (p. 587)
- I forgot my personal access token (PAT) (p. 587)
- A pull request doesn’t display the changes I expect (p. 587)
- A pull request shows a status of Not mergeable (p. 587)

I have reached the maximum storage for my space and see warnings or errors

**Problem:** I want to commit code to one or more source repositories in CodeCatalyst, but I see an error. In the console, I see a message on the source repository page that I have reached the storage limit for the space.

**Possible fixes:** Depending on your role in the project or space, you can either reduce the size of one or more of your source repositories, delete unused source repositories, or change your billing tier to one that has more storage.

- To reduce the size of a source repository in a project, you can delete unused branches. For more information, see [To delete a branch (console)](p. 95) and [Contributor role](p. 488).
- To reduce the overall storage for a space, you can delete unused source repositories. For more information, see [Deleting a source repository](p. 93) and [Project administrator role](p. 485).
- To increase the amount of storage available for your space, you can change your billing tier to one with more storage. For more information, see [Changing your CodeCatalyst billing tier](p. 610) and [Space administrator role](p. 477).

I receive an error when trying to clone or push to an Amazon CodeCatalyst source repository

**Problem:** When I try to clone a source repository to a local computer or into an integrated development environment (IDE), or when I try to pull or push to a source repository, I receive a permissions error.
Possible fixes: You might not have a personal access token (PAT) for your AWS Builder ID, you might not have configured your credential management system with your PAT, or your PAT might have expired. Try one or more of the following solutions:

- Create a personal access token (PAT). For more information, see Managing personal access tokens in Amazon CodeCatalyst (p. 497).
- Make sure you have accepted an invitation to the project that contains the source repository and that you are still a member of that project. You cannot clone a source repository if you aren't an active member of that project. Sign in to the console and attempt to navigate to the space and the project where you're trying to clone a source repository. If you cannot see the project in the list of projects for the space, you either aren't a member of that project, or you haven't accepted an invitation to that project. For more information, see Accepting an invitation and creating your AWS Builder ID (p. 11).
- Make sure your clone command is formatted correctly and includes your AWS Builder ID. For example:

  https://LiJuan@git.us-west-2.codecatalyst.aws/v1/ExampleCorp/MyExampleProject/MyExampleRepo

- Use the AWS CLI to make sure that you have a PAT associated with your AWS Builder ID, and that it is not expired. If you don't have one or the PAT is expired, create one. For more information, see Managing personal access tokens in Amazon CodeCatalyst (p. 497).
- Try creating a Dev Environment to work with the code in the source repository instead of cloning it to a local repo or IDE. For more information, see Creating a Dev Environment (p. 113).

I need a source repository for my project

Problem: My project either doesn’t have a source repository, or I need another source repository for my project.

Possible fixes: Some projects are created without any resources. If you are a member of the project, you can create source repositories for that project in CodeCatalyst. If someone with the Space administrator role installs the GitHub Repositories and connects it to a GitHub account, you can link to available GitHub repositories to add them to your project if you have the Project administrator role. For more information, see Creating a source repository (p. 89) and Linking a source repository (p. 90).

My source repository is brand-new but contains a commit

Problem: I just created a source repository. It should be empty, but it has a commit, a branch, and a README.md file in it.

Possible fixes: This is expected behavior. All source repositories in CodeCatalyst include an initial commit that sets the default branch to main and includes either sample code (if the repository was created for a project using a blueprint that included sample code) or a template markdown file for a repository README file. You can create additional branches in the console and in Git clients. You can create and edit files in the console, and delete files in Dev Environments and Git clients.

I want a different branch as my default branch

Problem: My source repository came with a default branch named main, but I want a different branch as my default branch.

Possible fixes: You cannot change or delete the default branch in source repositories in CodeCatalyst. You can create additional branches and use those branches in source actions in workflows. You can also choose to link GitHub repositories and use them as repositories for your project.
I am receiving emails about activity in pull requests

**Problem:** I didn't sign up or configure email notifications about pull request activity, but I'm receiving them anyway.

**Possible fixes:** Email notifications are sent automatically about pull request activity. For more information, see [Working with pull requests in Amazon CodeCatalyst](p. 100).

I forgot my personal access token (PAT)

**Problem:** I've been using a PAT for cloning, pushing, and pulling code for source repositories, but I've lost the value for my token, and I can't find it in the CodeCatalyst console.

**Possible fixes:** The quickest way to solve this problem is to create another PAT and configure your credential manager or IDE to use this new PAT. We only display the value of a PAT when you create it. If you lose this value, it cannot be retrieved. For more information, see [Managing personal access tokens in Amazon CodeCatalyst](p. 497).

A pull request doesn’t display the changes I expect

**Problem:** I created a pull request, but I don’t see the changes I expect to see between the source and destination branches.

**Possible fixes:** This might be caused by a number of issues. Try one or more of the following solutions:

- You might be reviewing the changes between older revisions, or you might not be viewing the latest changes. Refresh your browser and make sure that you've chosen the comparison between revisions you want to view.
- Not all changes in a pull request can be displayed in the console. For example, you cannot view Git submodules in the console, so you cannot view differences in a submodule in a pull request. Some differences might be too large to display. For more information, see [Quotas for source repositories in CodeCatalyst](p. 110) and [Viewing a file](p. 99).
- Pull requests display the differences between the merge base and whatever revision you choose. When you create a pull request, the difference displayed for you is the difference between the tip of the source branch and the tip of the destination branch. Once the pull request has been created, the displayed difference is between the revision and its merge base. The merge base is the commit that was the tip of the destination branch when the revision was created. The merge base can change between revisions. For more information about differences and merge bases in Git, see [git-merge-base](https://git-scm.com/docs/git-merge-base) in the Git documentation.

A pull request shows a status of Not mergeable

**Problem:** I want to merge a pull request, but its status shows as Not mergeable.

**Possible fixes:** This can be caused by one or more problems:

- All required reviewers for your pull request must approve a pull request before it can be merged. Review the list of required reviewers for any reviewers with a clock icon next to the name. A clock icon indicates that the reviewer hasn't approved the pull request.

  **Note**
  If a required reviewer has been removed from your project before approving the pull request, you cannot merge the pull request. Close the pull request and create a new pull request.

- There might be a merge conflict between the source branch and the destination branch. CodeCatalyst does not support all possible Git merge strategies and options. You can evaluate the branches for
Troubleshooting problems with workflows

Consult the following sections to troubleshoot problems related to workflows in Amazon CodeCatalyst. For more information about workflows, see Build, test, and deploy with workflows in CodeCatalyst (p. 128).

Topics

- How do I fix "Workflow is inactive" messages? (p. 588)
- How do I fix "Unable to locate credentials" and "ExpiredToken" errors? (p. 589)
- How do I fix "Unable to connect to the server" errors? (p. 590)
- How do I fix workflow definition errors? (p. 590)
- Why are CodeDeploy fields missing from the visual editor? (p. 591)
- How do I fix IAM capabilities errors? (p. 591)
- How do I fix "npm install" errors? (p. 592)
- Why do multiple workflows have the same name? (p. 595)
- Can I store my workflow definition files in another folder? (p. 595)
- How do I add actions in sequence to my workflow? (p. 595)
- Why does my workflow successfully validate but fail at runtime? (p. 595)
- Auto-discovery doesn't discover any reports for my action (p. 596)
- My action fails on auto-discovered reports after I configure success criteria (p. 596)
- Auto-discovery generates reports that I don't want (p. 596)
- Auto-discovery generates many small reports for a single test framework (p. 597)
- Workflows listed under CI/CD don't match those in the source repository (p. 597)

How do I fix "Workflow is inactive" messages?

**Problem:** In the CodeCatalyst console, under CI/CD, Workflows, your workflow appears with the following message:

Workflow is inactive.

This message indicates that the workflow definition file contains a trigger that doesn't apply to the branch that you're currently on. For example, your workflow definition file might contain a PUSH trigger that references your main branch, but you're on a feature branch. Since the changes you're making in your feature branch don't apply to main, and will not start workflow runs in main, CodeCatalyst decommissions the workflow on the branch and marks it as Inactive.

**Possible fixes:**

If you want to start a workflow on your feature branch, you can do the following:

- In your feature branch, in the workflow definition file, remove the Branches property from the Triggers section so that it looks like this:

  ```
  Triggers:
  - Type: PUSH
  ```
This configuration causes the trigger to activate on a push to any branch, including your feature branch. If the trigger is activated, CodeCatalyst will start a workflow run using the workflow definition file and source files in whatever branch you’re pushing to.

- In your feature branch, in the workflow definition file, remove the Triggers section and run the workflow manually.
- In your feature branch, in the workflow definition file, change the PUSH section so that it references your feature branch rather than another branch (like main, for example).

Important
Be careful not to commit these changes if you don't intend to merge them to back to your main branch.

For more information about editing the workflow definition file, see To edit a workflow (p. 248).

For more information about triggers, see Working with triggers (p. 327).

How do I fix "Unable to locate credentials" and "ExpiredToken" errors?

Problem: While working through Tutorial: Deploy an application to Amazon EKS (p. 207), you see one or both of the following error messages in your development machine's terminal window:

Unable to locate credentials. You can configure credentials by running "aws configure".

ExpiredToken: The security token included in the request is expired

Possible fixes:

These errors indicate that the credentials that you're using to access AWS services have expired. In this case, do not run the aws configure command. Instead, use the following instructions to refresh your AWS access key and session token.

To refresh your AWS access key and session token

1. Make sure you have the AWS access portal URL, username, and password for the user that you're using the complete the Amazon EKS tutorial (codecatalyst-eks-user). You should have configured these items when you completed Step 1: Set up your development machine (p. 208) of the tutorial.

   Note
   If you do not have this information, go to the codecatalyst-eks-user details page in IAM Identity Center, choose Reset password, Generate a one-time password [...] and Reset password again to display the information on the screen.

2. Do one of the following:

   - Paste the AWS access portal URL into your browser's address bar.

   Or

   - Refresh the AWS access portal page if it's already loaded.

3. Sign in with the codecatalyst-eks-user's username and password, if you're not already signed in.

4. Choose AWS account, and then choose the name of the AWS account to which you assigned the codecatalyst-eks-user user and permission set.
5. Next to the permission set name (codecatalyst-eks-permission-set), choose Command line or programmatic access.
6. Copy the commands in the middle of the page. They look similar to the following:

```bash
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtntFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_SESSION_TOKEN="session-token"
```

...where `session-token` is a long random string.
7. Paste the commands into your terminal prompt on your development machine and press Enter.

The new keys and session token are loaded.

You have now refreshed your credentials. The AWS CLI, eksctl, and kubectl commands should now work.

### How do I fix "Unable to connect to the server" errors?

**Problem:** While working through the tutorial described in Tutorial: Deploy an application to Amazon EKS (p. 207), you see an error message similar to the following in your development machine’s terminal window:

Unable to connect to the server: dial tcp: lookup long-string.gr7.us-west-2.eks.amazonaws.com on 1.2.3.4:5: no such host

**Possible fixes:**

This error usually indicates that the credentials that the kubectl utility is using to connect to your Amazon EKS cluster have expired. To solve the issue, refresh the credentials by entering the following command at the terminal prompt:

```bash
aws eks update-kubeconfig --name codecatalyst-eks-cluster --region us-west-2
```

Where:

- `codecatalyst-eks-cluster` is replaced with the name of your Amazon EKS cluster.
- `us-west-2` is replaced with the AWS Region where your cluster is deployed.

### How do I fix workflow definition errors?

**Problem:** While editing a workflow, you choose the Validate button and the following message appears at the top of the CodeCatalyst console:

The workflow definition has errors. Fix the errors and choose Validate to verify your changes.

When you explore the console page further, you see an error similar to the following at the bottom:

The workflow definition could not be parsed because it contains YAML code that is not valid. The issue was detected while scanning line 17 but the error may have occurred elsewhere. Check spacing and indentation, and consider adding quotations around strings that contain colons or other special characters.

**Possible fixes:**
Why are CodeDeploy fields missing from the visual editor?

**Problem:** You are using a **Deploy to Amazon ECS** (p. 230) action, and you are not seeing the CodeDeploy fields such as **CodeDeploy AppSpec** in the workflow’s visual editor. This problem may occur because the Amazon ECS service that you specified in the **Service** field is not configured to perform blue/green deployments.

**Possible fixes:**

- Choose a different Amazon ECS service on the **Deploy to Amazon ECS** action’s **Configuration** tab. For more information, see **Adding the “Deploy to Amazon ECS” action** (p. 230).
- Configure the selected Amazon ECS service to perform blue/green deployments. For more information about configuring blue/green deployments, see **Blue/Green deployment with CodeDeploy** in the **Amazon Elastic Container Service Developer Guide**.

How do I fix IAM capabilities errors?

**Problem:** You are using a **Deploy AWS CloudFormation stack** (p. 228) action, and you see **##[error] requires capabilities: [capability-name]** in your **Deploy AWS CloudFormation stack** action’s logs.

**Possible fixes:** Complete the following procedure to add the capability to the workflow definition file. For more information about IAM capabilities, see **Acknowledging IAM resources in AWS CloudFormation templates** in the **IAM User Guide**.

**Visual**

To add an IAM capability using the visual editor

1. Open the CodeCatalyst console at **https://codecatalyst.aws/**.
2. Choose your project.
3. In the navigation pane, choose **CI/CD**, and then choose **Workflows**.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose **Edit**.
6. Choose **Visual**.
7. In the workflow diagram, choose your Deploy AWS CloudFormation stack action.
8. Choose the Configuration tab.
9. At the bottom, choose Advanced - optional.
10. In the Capabilities drop-down list, select the check box next to the capability mentioned in the error message. If the capability is not available in the list, use the YAML editor to add it.
11. (Optional) Choose Validate to validate the workflow's YAML code before committing.
12. Choose Commit, enter a commit message, and choose Commit again.
13. If a new workflow run doesn't start automatically, run the workflow manually to see if the changes fix the error. For more information about running a workflow manually, see Starting a workflow run (p. 314).

YAML

To add an IAM capability using the YAML editor
1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Workflows.
4. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.
5. Choose Edit.
6. Choose YAML.
7. In the Deploy AWS CloudFormation stack action, add a capabilities property, like this:

```yaml
DeployCloudFormationStack:
  Configuration:
    capabilities: capability-name
```

Replace capability-name with the name of the IAM capability shown in the error message. Use commas and no spaces to list multiple capabilities. For more information, see the description of the capabilities property in the Deploy AWS CloudFormation stack action reference (p. 411).

8. (Optional) Choose Validate to validate the workflow's YAML code before committing.
9. Choose Commit, enter a commit message, and choose Commit again.
10. If a new workflow run doesn't start automatically, run the workflow manually to see if the changes fix the error. For more information about running a workflow manually, see Starting a workflow run (p. 314).

How do I fix "npm install" errors?

**Problem:** Your AWS CDK deploy action (p. 234) or AWS CDK bootstrap action (p. 254) fails with an npm install error. This error may occur because you are storing your AWS CDK app dependencies in a private node package manager (npm) registry that cannot be accessed by the action.

**Possible fixes:** Use the following instructions to update your AWS CDK app's cdk.json file with additional registry and authentication information.

**Before you begin**

1. Create secrets for your authentication information. You'll reference these secrets in the cdk.json file instead of providing the cleartext equivalents. To create the secrets:
How do I fix "npm install" errors?

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Choose your project.
3. In the navigation pane, choose CI/CD, and then choose Secrets.
4. Create two secrets with the following properties:

<table>
<thead>
<tr>
<th>First secret</th>
<th>Second secret</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name:</strong> npmUsername</td>
<td><strong>Name:</strong> npmAuthToken</td>
</tr>
<tr>
<td><strong>Value:</strong> npm-username, where npm-username is the username used to</td>
<td><strong>Value:</strong> npm-auth-token, where npm-auth-token is the access token used to</td>
</tr>
<tr>
<td>authenticate to your private npm registry.</td>
<td>authenticate to your private npm registry.</td>
</tr>
<tr>
<td><em>(Optional) Description:</em> The username used to authenticate to the private</td>
<td><em>(Optional) Description:</em> The access token used to authenticate to the</td>
</tr>
<tr>
<td>npm registry.</td>
<td>private npm registry.</td>
</tr>
</tbody>
</table>

For more information about secrets, see Working with secrets (p. 322).

2. Add the secrets as environment variables to your AWS CDK action. The action will replace the variables with real values when it runs. To add the secrets:

a. In the navigation pane, choose CI/CD, and then choose Workflows.

b. Choose the name of your workflow. You can filter by the source repository or branch name where the workflow is defined, or filter by workflow name.

c. Choose Edit.

d. Choose Visual.

e. In the workflow diagram, choose your AWS CDK action.

f. Choose the Inputs tab.

g. Add two variables with the following properties:

<table>
<thead>
<tr>
<th>First variable</th>
<th>Second variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name:</strong> NPMUSER</td>
<td><strong>Name:</strong> NPMTOKEN</td>
</tr>
<tr>
<td><strong>Value:</strong> ${Secrets.npmUsername}</td>
<td><strong>Value:</strong> ${Secrets.npmAuthToken}</td>
</tr>
</tbody>
</table>

You now have two variables containing references to secrets.

Your workflow definition file YAML code should look similar to the following:

**Note**
The following code sample is from an AWS CDK bootstrap action; a AWS CDK deploy action will look similar.

```yaml
Name: CDK_Bootstrap_Action
SchemaVersion: 1.0
Actions:
  CDKBootstrapAction:
```
You are now ready to use the NPMUSER and NPMTOKEN variables in your cdk.json file. Go to the next procedure.

**To update your cdk.json file**

1. Change to the root directory of your AWS CDK project, and open the cdk.json file.
2. Find the "app": property, and change it to include the code shown in *red italics*:

   ```json
   "watch": {   "include": [   "***"   ],
   "exclude": [   "README.md",
   "cdk*.json",
   "**/*.d.ts",
   "**/*.js",
   "tsconfig.json",
   "package*.json",
   ...
   }   }
   ```

3. In the code highlighted in *red italics*, replace:

   - **your-registry/folder/CDK-package/** with the path to your AWS CDK project dependencies in your private registry.
   - **hello-cdk.ts|js** with the name of your entrypoint file. This may be a .ts (TypeScript) or .js (JavaScript) file depending on the language you're using.

   **Note**
   The following sample code is from a TypeScript project. If you're using a JavaScript project, the code will look similar though not identical.

   **Note**
   The action will replace the NPMUSER and NPMTOKEN variables with the npm username and access token that you specified in Secrets.

4. Save your cdk.json file.
5. Re-run the action manually to see if the changes fix the error. For more information about running actions manually, see Starting a workflow run (p. 314).

Why do multiple workflows have the same name?

Workflows are stored per branch per repository. Two different workflows can have the same name if they exist in different branches. In the Workflows page, you can differentiate workflows of the same name by looking at the branch name. For more information, see Working with branches in Amazon CodeCatalyst (p. 94).

Can I store my workflow definition files in another folder?

No, you must store all workflow definition files in the .codecatalyst/workflows folder. If you are using a mono repo with multiple logical projects, place all your workflow definition files in the .codecatalyst/workflows folder, and then use the FilesChanged property inside a Trigger to trigger the workflow at a specified project path. For more information, see Working with triggers (p. 327).

How do I add actions in sequence to my workflow?

By default, when you add an action to your workflow, it will have no dependencies and will run in parallel with other actions.

If you want to arrange actions in sequence, you can set a dependency on another action by setting the DependsOn field. You can also configure an action to consume artifacts or variables which are outputs of other actions. For more information, see Configuring actions to depend on other actions (p. 280).

Why does my workflow successfully validate but fail at runtime?

If you validated your workflow using the Validate button, but your workflow failed anyway, it might because a limitation in the validator.

Any errors in reference to a CodeCatalyst resource like secrets, environments, or fleets in the workflow configuration will not register during a commit. If any references that aren't valid are used, the error will only be identified when a workflow is run. Similarly, if there are any errors in your action configuration
Auto-discovery doesn't discover any reports for my action

Problem: I configured auto-discovery for an action that runs tests, but no reports are discovered by CodeCatalyst.

Possible fixes: This might be caused by a number of issues. Try one or more of the following solutions:

- Make sure that the tool used to run tests produces outputs in one of the formats that CodeCatalyst understands. For example, if you would like pytest to allow CodeCatalyst to discover test and code coverage reports, include the following arguments:

```
--junitxml=test_results.xml --cov-report xml:test_coverage.xml
```

For more information, see Test report types (p. 147).

- Make sure that the file extension for the outputs are consistent with the chosen format. For example, when configuring pytest to produce results in JUnitXML format, check that the file extension is `.xml`. For more information, see Test report types (p. 147).

- Make sure that IncludePaths is configured to include the entire file system (`**/*`) unless you are excluding certain folders on purpose. Similarly, make sure that ExcludePaths don't exclude directories where you expect your reports to be located.

- If you manually configured a report to use a specific output file, it will be excluded from auto-discovery. For more information, see Example: Configuring reports (p. 152).

- Auto-discovery may not find reports because the action failed before any outputs were generated. For example, the build may have failed before any unit tests have been run.

My action fails on auto-discovered reports after I configure success criteria

Problem: When I enable auto-discovery and configure success criteria, some of the reports don't meet the success criteria and the action fails as a result.

Possible fixes: To resolve this, try one or more of the following solutions:

- Modify IncludePaths or ExcludePaths to exclude reports that you are not interested in.
- Update success criteria to allow all reports to pass. For example, if two reports were discovered with one having line coverage of 50% and another one of 70%, adjust the minimum line coverage to 50%. For more information, see Success criteria (p. 160)
- Turn the failing report into a manually configured report. This allows you to configure different success criteria for that specific report. For more information, see Configuring success criteria for reports (p. 149).

Auto-discovery generates reports that I don't want

Problem: When I enable auto-discovery, it generates reports that I don't want. For example, CodeCatalyst generates code coverage reports for files included in my application's dependencies stored in `node_modules`.
Possible fixes: You can adjust the ExcludePaths configuration to exclude unwanted files. For example, to exclude node_modules, add node_modules/**/*.

Auto-discovery generates many small reports for a single test framework

Problem: When I use certain test and code coverage reporting frameworks, I noticed that auto-discovery generates a large number of reports. For example, when using the Maven Surefire Plugin, auto-discovery produces a different report for each test class.

Possible fixes: Your framework may be able to aggregate outputs into a single file. For example, if you are using Maven Surefire Plugin, you can use npx junit-merge to aggregate the files manually. The full expression may look like this:

```
mvn test; cd test-package-path/surefire-reports && npx junit-merge -d ./ && rm *Test.xml
```

Workflows listed under CI/CD don't match those in the source repository

Problem: The workflows displayed on the CI/CD, Workflows page do not match those in the ~/.codecatalyst/workflows/ folder in your source repository (p. 79). You may see the following mismatches:

- A workflow appears on the Workflows page, but a corresponding workflow definition file does not exist in your source repository.
- A workflow definition file exists in your source repository, but a corresponding workflow does not appear on the Workflows page.
- A workflow exists in both the source repository and Workflows page, but the two are different.

This problem may occur if the Workflows page hasn't had time to refresh, or if a workflow quota was exceeded.

Possible fixes:

- Wait. You usually have to wait two or three seconds after a commit to source before you see the change on the Workflows page.
- If you've exceeded a workflow quota, do one of the following:

  Note
  To determine whether a workflow quota was exceeded, review Quotas for workflows in CodeCatalyst (p. 457), and cross-check the documented quotas against the workflows in your source repository or on the Workflows page. There is no error message to indicate that a quota was exceeded, so you'll have to investigate on your own.

- If you've exceeded the Maximum number of workflows per space quota, delete some workflows (p. 248) and then perform a test commit against the workflow definition file. An example of a test commit might be to add a space to the file.
- If you've exceeded the Maximum workflow definition file size quota, change the workflow definition file to reduce its length.
- If you've exceeded the Maximum number of workflow files processed in a single source event quota, perform several test commits. Modify fewer than the maximum number of workflows in each commit.
• Increase the workflow quotas by turning on Standard tier billing. For more information, see Billing (p. 608).

Troubleshooting problems with search in CodeCatalyst

Consult the following sections to troubleshoot problems related to searching in CodeCatalyst. For more information about workflows, see Search in CodeCatalyst (p. 576).

Topics
• I can't find a user in my project (p. 598)
• I don't see what I'm looking for in my project or space (p. 598)
• Number of search results keep changing when I navigate through the pages (p. 598)
• My search query isn't being completed (p. 598)

I can't find a user in my project

Problem: When I try to view a user's details, I don't see their information in the project.

Possible fixes: Search doesn't currently support searching for users within a project. To search for users with access to your space, switch to This space in QuickSearch, or remove any project filters you might have specified using the advanced query language.

I don't see what I'm looking for in my project or space

Problem: Results don't appear when I try to search for particular information.

Possible fixes: Content updates are likely to take a few seconds to update in search results. Large updates can take several minutes.

For resources that haven't been recently updated, you might need to refine your search. You can refine by adding more keywords or using the advanced query language. For more information about refining your queries, see Refining your search query (p. 576).

Number of search results keep changing when I navigate through the pages

Problem: The number of search results appear to change when I go to the next page, so it’s not clear how many total results there are.

Possible fixes: When navigating through pages of search results, you might see a change in the number of search results that match your query. The number of results might update to reflect a more accurate number of matches discovered as you navigate through the pages.

As you navigate through the results, you might see the following message: No results for "test". You will receive the message if you don't have access to the remaining results.

My search query isn't being completed

Problem: The results of my search query aren't showing up, and it appears to be taking too long.
Possible fixes: Your search may not be completed when there are many searches being made at the same time in the space, either programmatically or because of high team activity. If you're running programmatic searches, pause or decrease them. Otherwise, try again in a few seconds.

Troubleshooting problems with accounts associated with your space

In CodeCatalyst, you can add an AWS account to your space to grant permissions to resources and for billing purposes. The following information can help you troubleshoot common issues with associated accounts in CodeCatalyst.

Topics
- My AWS account connection request receives an invalid token error (p. 599)
- My Amazon CodeCatalyst project workflow fails with an error for the configured account, environment, or IAM role (p. 599)
- I need an associated account, role, and environment to create a project (p. 600)
- I cannot access the Amazon CodeCatalyst Spaces page in the AWS Management Console (p. 601)
- I cannot change the billing tier for my space (p. 601)
- I want a different account as my billing account (p. 586)

My AWS account connection request receives an invalid token error

Problem: When creating a connection request with a connection token, the page does not accept the token and shows an error stating that the token is not valid.

Possible fixes: Make sure you provide the account ID that you want to add to your space. You must have administrative permissions for your AWS account or be able to work with your administrator to add the account.

When you choose to verify the account, a new browser window will open in the AWS Management Console. The same account is required to be logged in on the console side. Try again after verifying the following:

- You are logged in to the AWS Management Console with the same AWS account that you want to add to your space.
- You are logged in to the AWS Management Console with the US West (Oregon) AWS Region (us-west-2) selected.
- If you have arrived from the billing page and you want to add the AWS account as a specified billing account for your space, make sure the account is not already a billing account for another space.

My Amazon CodeCatalyst project workflow fails with an error for the configured account, environment, or IAM role

Problem: When the workflow runs and does not find a configured account or IAM roles associated with your space, you must fill in the role, connection, and environment fields manually in the workflow YAML. View the failed workflow action, and note whether the error messages are as follows:
I need an associated account, role, and environment to create a project

**Problem:** In the project creation options, my project either doesn’t have an added account available in my space, or I need another account added to my space for my project to use.

**Possible fixes:** For your space, you can add authorized AWS accounts to add them to your project if you have the **Space administrator** role. You must also have an AWS account where you have administrative permissions or can work with your AWS administrator.

To make sure an account and role will be available in the project creation screen, you must first add the account and roles. For more information, see **Managing AWS accounts**.

You have the option to choose to create a service role with a role policy called the **CodeCatalystWorkflowDevelopmentRole-spaceName** role policy. The role will have a...
I cannot access the Amazon CodeCatalyst Spaces page in the AWS Management Console

Problem: When I try to access the Amazon CodeCatalyst page in the AWS Management Console to add an account to my CodeCatalyst space or add roles to an account in AWS, I receive a permissions error.

Possible fixes:

For your space, you can add authorized AWS accounts to add them to your project if you have the Space administrator role. You must also have an AWS account where you have administrative permissions or can work with your AWS administrator. You must first make sure you are signed in to the AWS Management Console with the same account that you want to manage. After you are signed in to the AWS Management Console, you can open the console and try again.


I cannot change the billing tier for my space

Problem: I am able to view billing for my space but cannot change the subscription tier from Free to Standard.

Possible fixes: For your space, you can change subscription tiers or authorize billing accounts if you have the Space administrator role.

Before you can change your CodeCatalyst plan from the Free tier, you must first have an account added to your space, as detailed in Managing personal access tokens in Amazon CodeCatalyst (p. 36). Next, you must set up the account for billing by turning on the Standard tier. See the steps in Turning on the Standard tier for a billing account (p. 609).

This does not change the billing tier for the space. It authorizes the Standard tier for the account, so the Space administrator will be able to upgrade to the Standard tier.

You must have the Space administrator role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

For more information, see Billing (p. 608).

I want a different account as my billing account

Problem: When I set up my CodeCatalyst login, I completed several steps to set up my space and associate an authorized AWS account. Now, I want to authorize a different account for billing.

Possible fixes: For your space, you can authorize billing accounts if you have the Space administrator role. You must also have an AWS account where you have administrative permissions or can work with your AWS administrator.

For more information, see Billing (p. 608).
Troubleshooting problems with Dev Environments

Consult the following sections to troubleshoot problems related to Dev Environments. For more information about Dev Environments, see Dev Environments in CodeCatalyst (p. 113).

Topics

- My Dev Environment creation didn't succeed (p. 602)
- My Dev Environment didn't resume (p. 602)
- My Dev Environment disconnected (p. 602)
- I can't find which directory my project is in (p. 602)
- Troubleshooting problems with IDEs (p. 602)
- Troubleshooting problems with devfiles (p. 603)

My Dev Environment creation didn't succeed

To fix the problem, view the Billing tab of the Amazon CodeCatalyst space to see if the Usage quotas have reached the maximum limits. If the quotas have reached the maximum limit, contact your Space administrator to raise the billing tier.

If the issue continues and you get an error in your IDE, check that you have a CodeCatalyst role that allows you to create a Dev Environment. The Space administrator role, Project administrator role, and Contributor role all have permission to create Dev Environments. For more information, see Working with roles in Amazon CodeCatalyst (p. 476).

My Dev Environment didn't resume

To fix the problem, view the Billing tab of the Amazon CodeCatalyst space to see if the Usage quotas have reached the maximum limits. If the quotas have reached the maximum limit, contact your Space administrator to raise the billing tier.

My Dev Environment disconnected

To fix the problem, check your internet connection. If you are not connected to the internet, connect and resume working in your Dev Environment.

I can't find which directory my project is in

To locate your project, change directory to /projects. This is the directory where you can find your projects.

Troubleshooting problems with IDEs

Consult the following sections to troubleshoot problems related to IDEs in CodeCatalyst. For more information on IDEs, see Using Dev Environments with an IDE (p. 126).

Topics

- I have mismatched runtime image versions in AWS Cloud9 (p. 603)
- I can't access my files in /projects/projects in AWS Cloud9 (p. 603)
- I can't launch my Dev Environment in AWS Cloud9 using a custom devfile (p. 603)
- I'm having issues in AWS Cloud9 (p. 603)
- In JetBrains, I can't connect to my Dev Environments through CodeCatalyst (p. 603)
I have mismatched runtime image versions in AWS Cloud9

AWS Cloud9 is using different versions of the frontend asset and the backend runtime image. Using different versions might cause the Git extension and AWS Toolkit to work incorrectly. To fix the problem, navigate to the Dev Environment dashboard, stop your Dev Environment, and then start it again. To fix the problem using APIs, use the UpdateDevEnvironment API to update the runtime. For more information, see UpdateDevEnvironment in the Amazon CodeCatalyst API reference.

I can't access my files in /projects/projects in AWS Cloud9

The AWS Cloud9 editor is unable to access files in the directory /projects/projects. To fix the problem, use the AWS Cloud9 terminal to access your files or move them to a different directory.

I can't launch my Dev Environment in AWS Cloud9 using a custom devfile

Your devfile image might not be compatible with AWS Cloud9. To fix the problem, review the devfile from your repository and corresponding Dev Environment and create a new one to continue.

I'm having issues in AWS Cloud9

For other issues, check the troubleshooting section in the AWS Cloud9 User Guide.

In JetBrains, I can't connect to my Dev Environments through CodeCatalyst

To fix the problem, check that you have only latest version of JetBrains installed. If you have multiple versions, uninstall the older versions and register your protocol handler again by closing the IDE and the browser. Then open JetBrains and register the protocol handler again.

I can't install AWS Toolkit for my IDE

To fix this problem for VS Code, manually install AWS Toolkit for Visual Studio Code from GitHub.

To fix this problem for JetBrains, manually install AWS Toolkit for JetBrains from GitHub.

In my IDE, I can't launch my Dev Environments

To fix this problem for VS Code, check that you have latest version of VS Code and AWS Toolkit for Visual Studio Code installed. If you don't have the latest version, update and launch your Dev Environment. For more information, see Amazon CodeCatalyst for VS Code.

To fix this problem for JetBrains, check that you have latest version of JetBrains and AWS Toolkit for JetBrains installed. If you don't have the latest version, update and launch your Dev Environment. For more information, see Amazon CodeCatalyst for JetBrains.

Troubleshooting problems with devfiles

Consult the following sections to troubleshoot problems related to devfiles in CodeCatalyst. For more information on devfiles, see Configuring your Dev Environment (p. 118).
Troubleshooting devfiles

Topics

- **My Dev Environment is using the default universal devfile even though I have implemented a custom image in a custom devfile (p. 604)**
- **My project is not building in my Dev Environment with the default universal devfile (p. 604)**
- **I want to move a repository devfile for a Dev Environment (p. 604)**
- **I'm having a problem starting my devfile (p. 604)**
- **I'm not sure how to check my devfile status (p. 604)**
- **My devfile is not compatible with the tooling provided in the latest image (p. 605)**

My Dev Environment is using the default universal devfile even though I have implemented a custom image in a custom devfile

If CodeCatalyst encounter errors while starting a Dev Environment that is using a custom devfile, the Dev Environment defaults to the default universal devfile. To fix the problem, view the Dev Environment logs in the /aws/mde/logs file to locate and fix any errors.

My project is not building in my Dev Environment with the default universal devfile

To fix the problem check that you are not using a custom devfile. If you are not using a custom devfile, view the devfile.yaml file in the source repository of the project to locate and fix any errors.

I want to move a repository devfile for a Dev Environment

You can move the default devfile in /projects/devfile.yaml to your source code repository. To update the location of the devfile, use following command: /aws/mde/mde start --location repository-name/devfile.yaml.

I'm having a problem starting my devfile

If there's a problem starting your devfile, it will enter recovery mode so that you can still connect to your environment and fix your devfile. While in recovery mode, running /aws/mde/mde status won't contain the location of your devfile.

```json
{
  "status": "STABLE"
}
```

You can check the error in the logs under /aws/mde/logs, fix the devfile, and try running /aws/mde/mde start again.

I'm not sure how to check my devfile status

You can check your devfile status by running /aws/mde/mde status. After running this command, you may see one of the following:

- `{"status": "STABLE", "location": "devfile.yaml" }`
  
  This indicates that your devfile is correct.

- `{"status": "STABLE" }`
  
  This indicates that your devfile is could not start and has entered recovery mode.
You can check the exact error in the logs under /aws/mde/logs/devfile.log.

You can also check if postStart execution was successful in your logs: /aws/mde/logs/devfileCommand.log.

For more information, see Universal devfile images (p. 123).

My devfile is not compatible with the tooling provided in the latest image

In your Dev Environment, devfile or devfile postStart may fail if the latest tooling does not have the tooling required for a specific project. To fix the problem, do the following:

1. Navigate to your devfile.
2. In your devfile, update to a granular image version instead of latest. It may look similar to the following:

   ```
   components:
   - container:
     image: public.ecr.aws/amazonlinux/universal-image:1.0
   ```

3. Create a new Dev Environment using the updated devfile.

Troubleshooting problems with issues

The following information can help you troubleshoot common problems with issues in CodeCatalyst.

Topics
- I can't choose an assignee for my issue (p. 605)

I can't choose an assignee for my issue

Problem: When creating an issue, the list of assignees is empty.

Possible fixes: The list of assignees is directly linked to the CodeCatalyst users listed as members for the project. To verify that user profile access is functioning properly, choose the profile icon and then choose User profile. If the user profile information does not populate, check the health report for any incidents. If it does populate, file a service ticket.

Troubleshooting problems between Amazon CodeCatalyst and the AWS SDKs or the AWS CLI

The following information can help you troubleshoot common issues when working with CodeCatalyst and the AWS CLI or the AWS SDKs.

Topics
- I receive an error when I enter aws codecatalyst at a command line or terminal saying it's an invalid choice (p. 606)
- I receive a credentials error when I run aws codecatalyst commands (p. 606)
I receive an error when I enter **aws codecatalyst** at a command line or terminal saying it's an invalid choice

**Problem:** When I try to use the AWS CLI with CodeCatalyst, one or more of the **aws codecatalyst** commands are not recognized as valid.

**Solution:** The most common cause for this problem is that you are using a version of the AWS CLI that does not contain the most recent updates for the latest services and commands. Update your installation of the AWS CLI and then try again. For more information see [Setting up to use the AWS CLI with CodeCatalyst (p. 15)](#).

I receive a credentials error when I run **aws codecatalyst** commands

**Problem:** When I try to use the AWS CLI with CodeCatalyst, I receive a message stating You can configure credentials by running "aws configure". or Unable to locate authorization token.

**Solution:** You must configure an AWS CLI profile to work with CodeCatalyst commands. For more information see [Setting up to use the AWS CLI with CodeCatalyst (p. 15)](#).
CodeCatalyst health report

The Amazon CodeCatalyst health report is a public dashboard that provides users with an aggregated list of up-to-the-minute notifications regarding resource performance and availability of services in CodeCatalyst that have widespread impact. You can see which resources are having problems and may affect applications in CodeCatalyst. This allows you to track outages and other resource downtime system wide. When an incident occurs, a blue indicator appears on the health report icon. Additionally, CodeCatalyst automatically sends an alert and email notification to all users with the Space administrator role in the project, providing details and a history of the incident in near real time.

The dashboard provides a list of all active events and a record of up to 100 previous incidents occurring in the last 30 days. You can organize the list of incidents based on the date the incident was updated. You can also refresh the list of incidents for up to the minute updates.

Here is a possible workflow for using the CodeCatalyst health report:

Mateo Jackson is a developer in the Budding Space with Space administrator permissions. While attempting to create a pull request, he keeps getting an error message. He checks his email and discovers he received an auto-generated system incident email from CodeCatalyst providing a detailed history about the system issue affecting his space. He chooses View update and is taken to the CodeCatalyst health report where he can view all system-reported incidents. He chooses the incident from the list to find out more information. A split screen opens that provides a timestamp of the last update, history, impacted capabilities, start time, and current status of the incident. He can also see that the issue is ongoing, but the service team has begun working on it. Each time there is an update to the history or status of the incident, he receives an email. In the event that he does not have access to his email, he can choose the bell icon in the top panel to get to the CodeCatalyst health report.

CodeCatalyst health report concepts

Learning the following concepts will help you understand the CodeCatalyst health report and how they enable you to track the health of your applications, services, and resources.

**Incident**

The incident is the system event that is affecting applications and resources within CodeCatalyst. You can choose the incident to view a detailed history of the event, including the time it began and if the service team is working on resolving it.

**Status**

The status is the real-time status of the incident. It will show as Ongoing or Resolved.

**Impacted capabilities**

The impacted capabilities are the resources or applications affected by the incident. A single incident can affect multiple areas in the system, including pull requests, issues, workflows, test, deploy, and source.

**Updated on**

Updated on provides a timestamp of the last update for the incident.
Billing

Your CodeCatalyst space uses resources, such as Dev Environments and compute, that are charged based on separate subscription tiers. To check your usage and to view pricing for each subscription tier, see the Billing tab for your space.

You can view or change your CodeCatalyst subscription for your space. CodeCatalyst has a Free tier and a Standard tier. If usage is needed beyond the Free tier, you can change subscriptions to the Standard tier. To set up for billing, you must add an authorized AWS account to your space.

Additional considerations:

• All spaces are required to have a billing account, even if the usage does not exceed the Free tier. We recommend that you set up billing using the Standard tier in order to prevent interruption in case your usage goes beyond the Free tier.

• AWS is the billing provider for CodeCatalyst. The account you use for billing must be an AWS account.

• Billing is configured at the space level. Multiple AWS accounts can be associated with a space, but only one can be active for CodeCatalyst billing at a time. Only one AWS account can be used as a billing account for a space in CodeCatalyst. In other words, if you have more than one space, you must use a different account as the billing account for the additional space.

• You can view your CodeCatalyst charges on your AWS invoice, including current charges. Billing data is updated regularly on the AWS Management Console page for AWS Billing.

Here is one possible flow to set up billing in CodeCatalyst.

Mary Major is a Space administrator for a CodeCatalyst space. Mary works with the AWS administrator for the AWS account that was added to her space in CodeCatalyst when it was created and so defaults as the billing account. Mary can also specify a different account as a billing account for her space. Mary initially set up her subscription to default to the Free tier. Mary knows that hitting the maximum can halt services for the rest of her space on the Free tier, so she periodically reviews usage for her space on the CodeCatalyst console Billing tab. Mary views the pricing subscription for the Standard tier, and after the space grows in size, Mary decides to use the CodeCatalyst billing page to change the subscription tier from the Free tier to the Standard tier. Mary works with her AWS administrator to access her AWS invoice and views the charges for usage beyond the Free tier in CodeCatalyst.

Viewing your CodeCatalyst plan and billing

Your CodeCatalyst space defaults to the Free tier. Your current subscription tier and the configured billing account are available in the CodeCatalyst console.

To access billing information, you must have the Space administrator role.

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space. Choose Settings, and then choose Billing.

   The Billing details page displays.
3. On the Billing details page, view your current subscription and usage. In Name and Status, view the configured AWS account for billing.
To view the CodeCatalyst billing status

1. In the AWS Management Console, make sure you are signed in with the same account that you want to manage. You will be automatically directed to a page in the AWS Management Console from the CodeCatalyst console.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. Navigate to your CodeCatalyst space. Choose Settings, and then choose Billing.

The Billing details page displays.
4. Choose CodeCatalyst in AWS console. You are directed to the Amazon CodeCatalyst Spaces page.

   You can view details about the connection between your AWS account and CodeCatalyst space.
   Under Billing details, view the CodeCatalyst subscriptions allowed for the account.

Viewing CodeCatalyst accounts for billing in AWS

The Amazon CodeCatalyst page in the AWS Management Consoleshowsthe account name that you have specified for billing. This page also shows the account connections and whether the account has billing turned on.

You must have the Space administrator role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

To view the CodeCatalyst billing status page for an added account

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space. Choose Settings, and then choose Billing.

   The Billing details page displays.
3. In AWS account name, view the account name, such as my-space-billing-account. In Status, view the billing status for the account.

To view CodeCatalyst connections and billing in AWS

1. In the AWS Management Console, make sure you are signed in with the same account that you want to manage. You will be automatically directed to a page in the AWS Management Console from the CodeCatalyst console.
2. Open the CodeCatalyst console at https://codecatalyst.aws/.
3. Navigate to your CodeCatalyst space. Choose Settings, and then choose Billing.

   The Billing details page displays.

   Note
   You might need to sign in to AWS before you can open the page in the AWS Management Console.

Turning on the Standard tier for a billing account

To set up a billing account, your AWS administrator must authorize the CodeCatalyst tiers that you want to allow your space to use.
Changing your CodeCatalyst billing tier

Your CodeCatalyst space defaults to the Free tier. Your current plan and configuration for billing, if set up, is available in the CodeCatalyst console.

You must have the **Space administrator** role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

1. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **Billing**.

   The **Billing details** page displays.

2. Choose **Manage billing in AWS**. This opens the Amazon CodeCatalyst Spaces in the AWS Management Console. If you are prompted to sign in, sign in to AWS, and then choose the button again to load the page.

3. Choose **Spaces**. The list of each space with a connection to the account you are signed in with, if any, displays.

4. Choose the link to the connection ID for the account that you added to your CodeCatalyst space. The **Edit connection** page displays.

5. Under **Billing details**, choose **Edit billing**. The **Update allowed billing tiers** page displays.

6. Choose **Spaces**. The list of each space with a connection to the account you are signed in with, if any, displays.

7. Choose **Change billing tier**. The **Compare tiers** page displays with available plans and the list of charges that apply to each plan.

   On the **Compare tiers** page, choose the tier with the charges that you want to authorize your CodeCatalyst space to use. The following tiers are available:

   - **Free**
   - **Standard**

8. Choose **Change tier**.
Setting up a billing account

To set up billing, you must add an authorized AWS account to your CodeCatalyst space and configure it for billing. Before you can set up a billing account, you must work with your AWS administrator to complete the prerequisites in the following procedure.

Before you can change your CodeCatalyst plan from the Free tier, you must first have an account set up for billing. When your space was created, an account was added to your space. This account will be used by default, but you can change it to another account.

Where necessary, these steps link you to the procedures in this guide for adding the AWS account you want to use for billing by creating a connection and then turn on the tiers for the account.

Only one AWS account can be used as a billing account for a space in CodeCatalyst. If an account is already used for a space, you must use a different billing account for the additional space.

You must have the **Space administrator** role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

**Step 1: Add the billing account to your space**

To choose a new account for billing, you must have completed the steps to add it to your space. Complete the steps as detailed in *Adding an AWS account to a space* (p. 37).

You must have the **Space administrator** role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

**Step 2: Turn on the Standard tier for your billing account**

Before you can change your CodeCatalyst plan from the Free tier, you must first have an account added as detailed in the previous step. Next, you set up the account for billing by turning on the Standard tier. See the steps in *Turning on the Standard tier for a billing account* (p. 609).

This does not change the billing tier for the space. It allows the Standard tier for the account, so the **Space administrator** will be able to upgrade to the Standard tier.

You must have the **Space administrator** role in CodeCatalyst and have administrator permissions for your account in AWS to manage billing.

**Step 3: Choose the billing account in CodeCatalyst**

Next, specify the connected account as a billing account.

The account ID is now authorized for billing. This is the account that you will add to your space, and it is available in the list of options. Only AWS accounts that have been added to your space will show in the list.

You must have the **Space administrator** role to manage billing and accounts for your space.

**To designate an added AWS account to a CodeCatalyst space for billing**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space. Choose **Settings**, and then choose **Billing**.
Changing a billing account

You can change the AWS account that you want to specify as the billing account for your CodeCatalyst space.

Important
You cannot delete the connection for an account that is designated as the billing account for your space.

You must have the Space administrator role to manage billing and accounts for your space.

Only one AWS account can be used as a billing account for a space in CodeCatalyst. If an account is already used for a space, you must use a different billing account for the additional space.

To change the AWS account for a CodeCatalyst space for billing

1. Open the CodeCatalyst console at https://codecatalyst.aws/.
2. Navigate to your CodeCatalyst space. Choose Settings, and then choose Billing.

   The Billing details page displays.
3. Choose Change billing tier. The Compare tiers page displays with available plans and the list of charges that apply to each plan. Choose Compare tiers, and then choose Change tier.
5. Choose the Edit button next to the AWS account name field. The Change AWS billing account page displays.
6. In Choose from AWS accounts already associated with your space, choose the name of another added account that you want to designate as the billing account for your CodeCatalyst space. The account name must be for an account that has been added to your space with a connection and is listed in the AWS accounts list for your space.
7. Choose Change billing account.

As a next step, we recommend that you authorize the billing account for the Standard tier. This authorizes the tier for the account so that the Space administrator can change the billing tier in CodeCatalyst. See the steps in Turning on the Standard tier for a billing account (p. 609).

Getting AWS billing information

If you have the necessary permissions, you can get information about your AWS charges from the AWS Management Console.
For more information about billing, including information about the tiers available, rates, and information included in the monthly AWS billing cycle, see the Amazon CodeCatalyst pricing page.

**To get your billing information**

1. Sign in to the AWS Management Console and navigate to **Billing**. In the navigation bar, choose your account name.
2. Choose **Billing dashboard**.
3. Use the AWS Billing and Cost Management dashboard to find a summary and a breakdown of your monthly spending. To learn more, see the AWS Billing User Guide.
AWS Support for Amazon CodeCatalyst

When you create a space, you must connect an AWS account and designate it as the billing account for your space. The AWS account you designate as your billing account is also where you access your AWS Support plan for Amazon CodeCatalyst. If you need support, you can create support cases from this designated AWS account.

CodeCatalyst users in a space use the AWS Support for Amazon CodeCatalyst page in CodeCatalyst to manage support cases. You can upgrade to an AWS Support plan such as Business Support or Enterprise Support to create and manage CodeCatalyst technical support cases in CodeCatalyst.

Only cases specific to the CodeCatalyst service and resources can be supported through AWS Support for Amazon CodeCatalyst. CodeCatalyst resources include resources deployed within CodeCatalyst and by users in CodeCatalyst, but these do not include resources deployed for other AWS or third-party services. If you need support for any other AWS service, you must open it through the AWS Management Console.

To change your support plan, see Changing support plans.

Note
Developer Support plans are not designed for production environments. If a space billing account has a Developer Support plan, this plan does not cascade to all space administrators and space members within AWS Support in CodeCatalyst.

Billing for AWS Support for Amazon CodeCatalyst

When you create a space in CodeCatalyst, users in the space can create and manage support cases from AWS Support for Amazon CodeCatalyst. You can create two types of customer cases:

- **Account and billing** support cases are available to all CodeCatalyst users in the space. You can get help with billing and account questions based on your permissions in CodeCatalyst.
- **Technical** support cases connect you to a technical support engineer for help with service-related technical issues and extensions to third-party applications. If you have Basic Support, you can’t create a technical support case.

The AWS account designated as the billing account for the space must have a Business Support or Enterprise Support plan for the space to use AWS Support for CodeCatalyst for technical cases.

Note
If your space uses AWS Support for Amazon CodeCatalyst from an account that doesn't have a Business Support or Enterprise Support plan, you can still use AWS Support for Amazon CodeCatalyst for account and billing cases.

For technical support, you must open all cases through the CodeCatalyst console. You cannot create technical support cases for CodeCatalyst from AWS Support in the AWS Management Console.

Note
Service limit increase requests are not available from AWS Support for Amazon CodeCatalyst. These requests can only be submitted by the root user for the space billing account in the AWS Support Center Console.
AWS Support for Amazon CodeCatalyst has the same support agreements as AWS Support, with the following considerations:

- Severity lists, response times, and SLAs in AWS Support apply for support cases in AWS Support for CodeCatalyst, as detailed in Choosing a severity.
- Space administrators and space members cannot use the AWS Support APIs or AWS SDK or AWS Support app in Slack to create cases for CodeCatalyst. CodeCatalyst support cases can only be submitted from CodeCatalyst.

**Note**
CodeCatalyst is not fully integrated with AWS Trusted Advisor or AWS Incident Detection and Response. Validate how CodeCatalyst is integrated to ensure your business practices are aligned with the current integration.

You must be a user in the space where you want to request support.

**Note**
If you have more than one builder in your space, we recommend that you purchase a Business Support or Enterprise Support plan. These plans provide technical support for the space for up to 5,000 builders.

The AWS account designated as the billing account for the space uses the AWSRoleForCodeCatalystSupport role and AmazonCodeCatalystSupportAccess managed policy. This allows CodeCatalyst users in a space to access the AWS Support for Amazon CodeCatalyst page. For more information about this role and policy, see AmazonCodeCatalystSupportAccess. For other considerations about billing and account connections, see Billing and Managing AWS accounts.

Here is a possible flow for a builder creating a support case in CodeCatalyst:

Mateo Jackson is a developer on a project in CodeCatalyst. After signing up the AWS account that manages billing with AWS Support for Amazon CodeCatalyst and upgrading to a Business Support plan, all builders in the space can create technical support cases. Mateo submits a technical support case for a failed workflow in their project. Mateo uses the AWS Support for Amazon CodeCatalyst page to fill out the form and create a case, providing the workflow ID and other details in the request. The case is created with a case ID and includes the account ID of the AWS account designated as the billing account and associated with support plan for the space.

While all builders can create support cases in AWS Support for CodeCatalyst, you are not charged for each case created. You can open virtually unlimited cases and contacts based on the AWS Support Premium plan you purchase on your space billing account.

**Note**
The space billing account is the AWS account that you are charged for CodeCatalyst users and resources. If you have deployed to additional AWS accounts, contact AWS Support through the AWS Management Console for assistance with resources deployed to other services. You can identify the AWS account you deployed to from the workflow.
The AWSRoleForCodeCatalystSupport role is a service role that is used for support cases in your space. The role must be added to the designated billing account for the space. For more information or to create the role, see Creating the AWSRoleForCodeCatalystSupport role for your account and space (p. 518).

Note
For a space that was created before April 20, 2023, you must create the role in order for support for CodeCatalyst to work for your space. If creating a space after April 20, 2023, you can create the role during space creation, on the Billing details page in CodeCatalyst, or by clicking the support banner link in CodeCatalyst.

To set up support for your space

1. When you create a CodeCatalyst space, you are instructed to connect a billing account. The designated billing account for the space will be billed by AWS. For more information about creating a space, see Sign up to create your first space and your development role (p. 8).

2. When you create a CodeCatalyst space, the option is available to create the AWSRoleForCodeCatalystSupport service role that allows CodeCatalyst users to access support. The role uses the managed policy AmazonCodeCatalystSupportAccess. The role must be added to the AWS account designated as the billing account for the space. For more information about creating this role, see Creating the AWSRoleForCodeCatalystSupport role for your account and space (p. 518).

3. For the designated billing account for the space, the space administrator is recommended to purchase a Business Support or Enterprise Support plan for the AWS account. All members in the space will be able to manage support cases from AWS Support for Amazon CodeCatalyst, and channels of support will be aligned to the AWS Support plan you have purchased where integrations are completed.

4. To create and manage support cases in CodeCatalyst, see Creating a CodeCatalyst support case in CodeCatalyst (p. 617).

Accessing support for CodeCatalyst in the AWS Management Console

If the support enabled billing account for a space is disconnected, AWS Support cases associated with the previous space billing account and associated support plan will no longer be visible in AWS Support for Amazon CodeCatalyst. The root user for that billing account can view and resolve old cases from the AWS Management Console and can set up IAM permissions for AWS Support for other users to view and resolve old cases. You will still be able to partake in the benefits of your support plan from the AWS Management Console for all other AWS services and complete any CodeCatalyst support cases that were not previously resolved.

For more information, see Updating, resolving, and reopening your case in the AWS Support User Guide.

Support cases for general how-to information about CodeCatalyst can also be opened in the AWS Management Console, but no technical support can be received through this channel for CodeCatalyst. For more information, see Creating support cases and case management in the AWS Support User Guide.

Here is a possible flow for a user resolving a support case for CodeCatalyst in the AWS Management Console:

While all builders can create support cases with AWS Support for Amazon CodeCatalyst, the support requests are billed from the account that is designated as the billing account for the space. Mateo Jackson is a developer on a project in CodeCatalyst who opened a technical support case for a failed workflow in their project. However, the billing account for the space that was signed up with AWS Support for Amazon CodeCatalyst and had purchased a Business Support plan has been disconnected.
from the space. The only way for Mateo to view the latest communication and resolve cases opened for CodeCatalyst is to manage the case ID from the AWS Support Center in the AWS Management Console. To do this, Mateo is given IAM permissions from the root user of the previous space billing account attached to their support case and resolves the case through AWS Support in the console.

**Important**

If you change the designated billing account for your space, your AWS Support plan will still be accessible until the end of the month through the AWS Management Console only. You will need to repurchase AWS Support on the updated billing account to continue accessing your previously created support cases in CodeCatalyst. We recommend waiting until you have resolved all of your support cases to change space billing accounts to avoid any impact to accessing your support cases through AWS Support for Amazon CodeCatalyst.

---

### Creating a CodeCatalyst support case in CodeCatalyst

You can create a support case in the AWS Support for Amazon CodeCatalyst page.

An AWS Builder ID can only get support for the alias they are authenticated with and only for resources based on their permissions. Account and billing options are available for all space administrators and space members. However, users can only get support for resources they have access to in CodeCatalyst and not in relation to managing billing for the account.

You can create an Account and Billing case or technical support case for your CodeCatalyst resources using the AWS Support for CodeCatalyst page for your space.

**Note**

Only cases specific to the CodeCatalyst service and resources can be supported through AWS Support for Amazon CodeCatalyst. CodeCatalyst resources include resources deployed within CodeCatalyst and by users in CodeCatalyst, but these do not include resources deployed for other AWS or third-party services. If you need support for any other AWS service, you must open it through the AWS Management Console.

**To create a support case in CodeCatalyst**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   
   **Tip**
   
   If you belong to more than one space, choose a space in the top navigation bar.
3. At the top of the page, choose the ? icon, and then choose **Support**.
4. Choose **Create case**.
5. Choose one of the following options:
   - **Account and billing**
   - **Technical**

   **Note**

In AWS Support for Amazon CodeCatalyst, if a Business Support or Enterprise Support plan is added to the space billing account, CodeCatalyst technical case support will be available to all space administrators and space members. For troubleshooting information, see [I cannot create technical support cases for my space (p. 583)](https://aws.amazon.com/support/help/community/cannot-create-technical-support-cases-for-my-space). AWS Support plans do not span across spaces. If you are a member of multiple spaces, your space administrator will need to purchase an AWS Support Premium plan for every space in order to receive technical support across all spaces.
6. Choose the **Service**, **Category**, and **Severity**. For information about choosing a severity, see [Choosing a severity](#).

   - **General guidance**
   - **System impaired**
   - **Production system impaired**
   - **Production system down**
   - **Business-critical system down**

7. Choose **Next step: Additional information**.

8. On the **Additional information** page, for **Subject**, enter a title about your issue.

9. For **Description**, follow the prompts to describe your case, such as the following:

   - Troubleshooting information that is specific to CodeCatalyst, such as workflow ID, logs, or screenshots
   - Error messages that you received
   - Troubleshooting steps that you followed

**Note**
Don’t share any sensitive information in case correspondences, such as credentials, credit cards, signed URLs, or personally identifiable information.

10. (Optional) Choose **Attach files** to add any relevant files to your case, such as error logs or screenshots. You can attach up to three files. Each file can be up to 5 MB.

11. In **Space name**, the name of your space displays.

12. In **Builder name**, the full name associated with your AWS Builder ID autopopulates.

13. (Optional) Choose the project in **Project name (if applicable)**.

**Note**
You will only be shown projects you have permissions to. If you need access to another project, ask your project administrator to provide you with access before creating a support case.

14. Choose **Next step: Contact us**.

15. In **Preferred contact language**, choose the default. Only **English** is available at this time.

16. Choose the **Web** option for contact method. This is the only supported contact method in AWS Support for Amazon CodeCatalyst.

17. Review your case details, and then choose **Submit**. Your case ID number and summary appear.

The support case is created at the space level and is viewable by all members with access to the space and project (if selected) that are defined in your support case. There is no way to omit a support case from individual users at this time.

---

**Resolving a support case in CodeCatalyst**

You can resolve open support cases from the AWS Support for Amazon CodeCatalyst page.

You must have a **Space administrator** or **Space member** role in the space where you want to resolve a support case. If you do not have the **Space administrator** role, or if a project was selected when the case was created, you will also need to have membership to the project to view and resolve the case.

**To resolve an open support case in CodeCatalyst**

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   
   **Tip**
   If you belong to more than one space, choose a space in the top navigation bar.
3. At the top of the page, choose the ? icon, and then choose **AWS Support for Amazon CodeCatalyst**.
4. Choose the link for the support case that you want to manage. Choose **Resolve case**.

## Reopening a support case in CodeCatalyst

You can use the reopen a resolved support case from the AWS Support for Amazon CodeCatalyst page.

**Note**
You can reopen your support case up to 14 days from when your issue was resolved. However, you can't reopen a case that has been inactive for more than 14 days. If you're unable to reopen your case, open a new case and include the previous case ID as a reference.
If you reopen an existing case that has different information than your current issue, the support agent might ask you to create a new case.

### To reopen a support case in CodeCatalyst

1. Open the CodeCatalyst console at [https://codecatalyst.aws/](https://codecatalyst.aws/).
2. Navigate to your CodeCatalyst space.
   
   **Tip**
   If you belong to more than one space, choose a space in the top navigation bar.
3. At the top of the page, choose the ? icon, and then choose **AWS Support for CodeCatalyst**.
4. Choose the link for the support case that you want to manage. Choose **Reopen**. Choose **OK** on the confirmation screen, and then choose **Submit**.
5. Fill in the **Description** with the latest information about the same issue. Don't share any sensitive information in case correspondences, such as credentials, credit cards, signed URLs, or personally identifiable information.
Quotas for CodeCatalyst

The following table describes quotas and limits for Amazon CodeCatalyst. You can find additional information for specific aspects of CodeCatalyst in the following topics:

- Quotas for source repositories in CodeCatalyst (p. 110)
- Quotas for identity, permission, and access in CodeCatalyst (p. 553)
- Quotas for workflows in CodeCatalyst (p. 457)
- Quotas for Dev Environments in CodeCatalyst (p. 127)
- Quotas for projects in CodeCatalyst (p. 65)
- Quotas for spaces in CodeCatalyst (p. 55)
- Quotas for issues in CodeCatalyst (p. 475)

<table>
<thead>
<tr>
<th>Maximum number of spaces in an account</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of spaces a user can create in a calendar month</td>
<td>5</td>
</tr>
<tr>
<td>Minimum number of AWS accounts for a space</td>
<td>1</td>
</tr>
<tr>
<td>Maximum number of account connections for a space</td>
<td>5,000</td>
</tr>
<tr>
<td>Maximum number of projects in a space</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of projects to which a user can belong</td>
<td>1,000</td>
</tr>
<tr>
<td>Space descriptions</td>
<td>Space descriptions are optional. If specified, they must be between 0 and 200 characters in length. They can contain any combination of letters, numbers, spaces, periods, underscores, commas, dashes, and the following special characters: <code>? &amp; $ % + = / \ ; : \n \t \r</code></td>
</tr>
<tr>
<td>Space names</td>
<td>Space names must be unique across CodeCatalyst. You cannot reuse names of deleted spaces. Space names must be between 3 and 63 characters in length. They must also begin with an alphanumeric character. Space names can contain any combination of letters, numbers, periods, underscores, and dashes. They cannot contain any of the following characters: `! ? @ # $ % ^ &amp; * ( ) + = { } [ ]</td>
</tr>
</tbody>
</table>
# Document history for Amazon CodeCatalyst

The following table describes the documentation history and updates for the overall documentation for CodeCatalyst.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated content: Working with pull requests (p. 100)</td>
<td>Updated the documentation to reflect changes in how information about a pull request is displayed.</td>
<td>November 10, 2023</td>
</tr>
<tr>
<td>Updated content: Working with file caching (p. 311) (p. 311)</td>
<td>Updated the documentation to include file caching limitations.</td>
<td>November 10, 2023</td>
</tr>
<tr>
<td>Updated content: Tutorial: Deploy an application to Amazon EKS (p. 207) (p. 207)</td>
<td>Updated the documentation to mention the EKS App Deployment blueprint.</td>
<td>November 9, 2023</td>
</tr>
<tr>
<td>New content: Reviewer role (p. 492)</td>
<td>Added documentation for the Reviewer role.</td>
<td>November 1, 2023</td>
</tr>
<tr>
<td>New content: Read only role (p. 495)</td>
<td>Added documentation for the Read only role.</td>
<td>November 1, 2023</td>
</tr>
<tr>
<td>New content: Power user role (p. 483)</td>
<td>Added documentation for the Power user role.</td>
<td>November 1, 2023</td>
</tr>
<tr>
<td>New content: Limited access role (p. 484)</td>
<td>Added documentation for the Limited access role.</td>
<td>November 1, 2023</td>
</tr>
<tr>
<td>New and updated content: Working with roles (p. 476)</td>
<td>Updated the documentation for four new roles in CodeCatalyst: Power user, Limited access, Reviewer, and Read only.</td>
<td>November 1, 2023</td>
</tr>
<tr>
<td>Updated content: Working with GitHub Actions output parameters (p. 276) (p. 276)</td>
<td>Updated the examples to use the GITHUB_OUTPUT environment file instead of the set-output command. Using environment files is GitHub’s recommended method for setting output parameters.</td>
<td>October 24, 2023</td>
</tr>
<tr>
<td>New content: Working with triggers (p. 327) (p. 327)</td>
<td>Added documentation for schedule triggers.</td>
<td>October 16, 2023</td>
</tr>
<tr>
<td>Updated content: &quot;Deploy to Kubernetes cluster&quot; action reference (p. 432) (p. 432)</td>
<td>Added information about using the CodeCatalystWorkflowDevelopmentRole-spaceName role to the &quot;Deploy to Kubernetes cluster&quot; action reference (p. 432) and Tutorial:</td>
<td>September 22, 2023</td>
</tr>
<tr>
<td>Updated content: New role name and policy for the CodeCatalystWorkflowDevelopmentRole-spaceName role (p. 516)</td>
<td>Deploy an application to Amazon EKS (p. 207) topics.</td>
<td>September 20, 2023</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Updated content: Working with variables (p. 337) (p. 337)</td>
<td>Introduced two new concepts: <em>user-defined variables</em> and <em>predefined variables</em>. These concepts should make the Working with variables (p. 337) section easier to read and understand.</td>
<td>September 19, 2023</td>
</tr>
<tr>
<td>Updated content: Working with commits (p. 108)</td>
<td>Updated the documentation to reflect the change in displayed information and provide details about viewing commits with multiple parents.</td>
<td>September 7, 2023</td>
</tr>
<tr>
<td>New content: Working with triggers (p. 327) (p. 327)</td>
<td>Added the following example to the Working with triggers (p. 327) topic: Example: A simple 'push to main' trigger (p. 334)</td>
<td>September 6, 2023</td>
</tr>
<tr>
<td>Updated content: New AWS Region availability for space creation in Amazon CodeCatalyst (p. 32)</td>
<td>Updated the steps for signing up and creating a space to include choosing a Region due to new Europe (Ireland) Region availability. See Creating additional spaces in CodeCatalyst (p. 53) and Sign up to create your first space and your development role (p. 8).</td>
<td>August 31, 2023</td>
</tr>
<tr>
<td>Updated content: Working with pull requests (p. 100)</td>
<td>Updated the documentation to reflect the change in display order for source branch and destination branch when creating a pull request.</td>
<td>August 30, 2023</td>
</tr>
<tr>
<td>New content: View and change the default branch (p. 94)</td>
<td>Added documentation for branches to include information about viewing and changing the default branch for a source repository.</td>
<td>August 30, 2023</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Updated content: “Deploy to Kubernetes cluster” action reference (p. 432) (p. 432)</td>
<td>Added a note about Helm and Kustomize to the Manifests (p. 438) property description in the “Deploy to Kubernetes cluster” action reference (p. 432).</td>
<td>August 15, 2023</td>
</tr>
<tr>
<td>New content: Managing issue attachments (p. 462)</td>
<td>Added documentation for working with and managing attachments on issues.</td>
<td>August 15, 2023</td>
</tr>
<tr>
<td>Updated content: Working with triggers (p. 327) (p. 327)</td>
<td>Improved and expanded the documentation related to workflow triggers.</td>
<td>August 11, 2023</td>
</tr>
<tr>
<td>New content: Troubleshooting role permissions (p. 74)</td>
<td>Added information about updating the role permissions to run a workflow that requires access to Amazon CodeGuru. See Modern three-tier web application blueprint workflow OnPullRequest fails with permissions error for Amazon CodeGuru (p. 75).</td>
<td>August 11, 2023</td>
</tr>
<tr>
<td>New content: How do I fix “Workflow is inactive” messages? (p. 588) (p. 588)</td>
<td>Added the following troubleshooting topic: How do I fix “Workflow is inactive” messages? (p. 588)</td>
<td>August 11, 2023</td>
</tr>
<tr>
<td>New content: Adding the “Deploy to Kubernetes cluster” action (p. 231) (p. 231)</td>
<td>Added documentation for the Deploy to Kubernetes cluster action. For more information, see Adding the “Deploy to Kubernetes cluster” action (p. 231) and Tutorial: Deploy an application to Amazon EKS (p. 207).</td>
<td>July 27, 2023</td>
</tr>
<tr>
<td>Updates for how management events are logged for a CodeCatalyst space (p. 545)</td>
<td>Added information about how management events are logged for specific actions in a CodeCatalyst space with AWS CloudTrail. Added information about how all events in a space can be viewed with the list-event-logs command. See Monitoring in Amazon CodeCatalyst (p. 545).</td>
<td>July 20, 2023</td>
</tr>
<tr>
<td>Updated content: Working with triggers (p. 327) (p. 327)</td>
<td>Updated the documentation to indicate that pull request triggers are now supported with GitHub source repositories. Previously, pull request triggers were only supported with CodeCatalyst source repositories.</td>
<td>July 14, 2023</td>
</tr>
<tr>
<td>Updated content: Workflow definition reference (p. 354) (p. 354)</td>
<td>Fixed a formatting error in the Compute code block.</td>
<td>June 27, 2023</td>
</tr>
<tr>
<td>Updated content: Quotas for workflows in CodeCatalyst (p. 457) (p. 457)</td>
<td>Updated the Quotas for workflows in CodeCatalyst (p. 457) topic with the Maximum amount of time an action can run quota.</td>
<td>June 27, 2023</td>
</tr>
<tr>
<td>Updated content: Data protection (p. 504)</td>
<td>Updated the documentation to include additional information about data replication.</td>
<td>June 26, 2023</td>
</tr>
<tr>
<td>Updated content: Working with environments (p. 307) (p. 307)</td>
<td>Clarified the Which actions support environments? (p. 307) section.</td>
<td>June 14, 2023</td>
</tr>
<tr>
<td>Updated content: Reorganized issues documentation (p. 459)</td>
<td>Reorganized most of the issues documentation to better align with the overall documentation set and user flows.</td>
<td>May 31, 2023</td>
</tr>
<tr>
<td>Updated content: Issues view switcher (p. 459)</td>
<td>Updated various user flows to align with the updated issue view switcher.</td>
<td>May 31, 2023</td>
</tr>
<tr>
<td>Updated content: Managing notifications (p. 68)</td>
<td>Updated the documentation for notifications to include information about configuring personal Slack notifications.</td>
<td>May 30, 2023</td>
</tr>
<tr>
<td>Updated content: Managing notifications (p. 71)</td>
<td>Updated the documentation for notifications to include information about configuring personal Slack notifications.</td>
<td>May 30, 2023</td>
</tr>
<tr>
<td>New content: CodeCatalyst trust model (p. 543)</td>
<td>Added a new topic with information about the trust model, which allows CodeCatalyst to assume the service role in the connected AWS account. Added a new section about the defined service principals for CodeCatalyst. See CodeCatalyst trust model (p. 543).</td>
<td>May 20, 2023</td>
</tr>
<tr>
<td>Updated content: Working with sources (p. 325) (p. 325)</td>
<td>Simplified the instructions in Referencing files in a source repository (p. 327).</td>
<td>May 10, 2023</td>
</tr>
<tr>
<td>Updated content: Quotas for workflows in CodeCatalyst (p. 457) (p. 457)</td>
<td>Updated the Quotas for workflows in CodeCatalyst (p. 457) topic with the Maximum length of an output variable value quota.</td>
<td>May 10, 2023</td>
</tr>
<tr>
<td>New content: Working with artifacts (p. 286) (p. 286)</td>
<td>Added two examples: Example: Referencing a file in a single artifact (p. 292) and Example: Referencing a file in an artifact when a WorkflowSource is present (p. 293).</td>
<td>May 10, 2023</td>
</tr>
<tr>
<td>Updated content: Working with pull requests (p. 100)</td>
<td>Updated the documentation for pull requests to include information about configuring email preferences for pull request events.</td>
<td>April 21, 2023</td>
</tr>
<tr>
<td>Updated content: Managing notifications (p. 71)</td>
<td>Updated the documentation for notifications to include information about configuring email preferences for pull request events.</td>
<td>April 21, 2023</td>
</tr>
<tr>
<td>Updates to managed policies (p. 512)</td>
<td>Added the AWS managed policy: AmazonCodeCatalystFullAccess (p. 513), AWS managed policy: AmazonCodeCatalystReadOnlyAccess (p. 514), and AWS managed policy: AmazonCodeCatalystSupportAccess (p. 512) managed policies. See CodeCatalyst updates to AWS managed policies (p. 515).</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>New content: Removing a deployment target (p. 245) (p. 245)</td>
<td>Added a Removing a deployment target (p. 245) topic.</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>New content: Action types (p. 249) (p. 249)</td>
<td>Added a CodeCatalyst Labs actions (p. 249) topic.</td>
<td>April 20, 2023</td>
</tr>
<tr>
<td>Updates for managing a user with the Space administrator role in a space (p. 32)</td>
<td>Added information about removing or changing the role for a user with the Space administrator role in a space. See Removing or changing the role for a user with the Space administrator role (p. 50).</td>
<td>April 19, 2023</td>
</tr>
<tr>
<td>Updates for administering Dev Environments (p. 32)</td>
<td>Added information about administering Dev Environments as a Space administrator. See Administering Dev Environments for a space (p. 51).</td>
<td>April 19, 2023</td>
</tr>
<tr>
<td>Updated content: Finding and viewing issues (p. 459)</td>
<td>Reorganized the Finding and viewing issues (p. 468) topic and subtopics.</td>
<td>April 19, 2023</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Updated content: Working with compute (p. 293) (p. 293)</td>
<td>Added support for Arm64 architecture on Amazon Linux 2.</td>
<td>April 19, 2023</td>
</tr>
<tr>
<td>New content: Moving issues within groups (p. 459)</td>
<td>Added documentation for moving issues within groups (p. 467) on the Board and All issues views.</td>
<td>April 19, 2023</td>
</tr>
<tr>
<td>Updated content: Quotas for workflows in CodeCatalyst (p. 457) (p. 457)</td>
<td>Updated the Quotas for workflows in CodeCatalyst (p. 457) topic with missing quotas, and updated the Maximum total size of a single action's output variables quota to 120 KB (from 2 KB).</td>
<td>April 18, 2023</td>
</tr>
<tr>
<td>New content: Viewing an action's source code (p. 283) (p. 283)</td>
<td>Added a Viewing an action's source code (p. 283) topic.</td>
<td>April 18, 2023</td>
</tr>
<tr>
<td>New content: Retrying test cases (p. 163) (p. 162)</td>
<td>Added a Retrying test cases (p. 163) topic.</td>
<td>April 11, 2023</td>
</tr>
<tr>
<td>New content: Stopping a workflow run (p. 314) (p. 314)</td>
<td>Added a Stopping a workflow run (p. 314) topic.</td>
<td>April 10, 2023</td>
</tr>
<tr>
<td>New content: Added sections for tagging resources for account connections between AWS and Amazon CodeCatalyst (p. 505)</td>
<td>Added information for tagging account connection resources and managing IAM policies for connection resources. See Tagging account connections (p. 43), Using tags to control access to account connection resources (p. 509), and CodeCatalyst permissions reference (p. 510).</td>
<td>April 6, 2023</td>
</tr>
<tr>
<td>New content: Action types (p. 249) (p. 249)</td>
<td>Added an Action types (p. 249) topic.</td>
<td>April 6, 2023</td>
</tr>
<tr>
<td>Updated content: &quot;Deploy AWS CloudFormation stack&quot; action reference (p. 411) (p. 411)</td>
<td>Updated the parameter-overrides property description. It now supports JSON files.</td>
<td>April 5, 2023</td>
</tr>
<tr>
<td>New content: Creating a project in CodeCatalyst with a linked GitHub repository (p. 58)</td>
<td>Added a new section to Creating a project in Amazon CodeCatalyst (p. 58) titled Creating a project with a linked GitHub repository (p. 59) with instructions to create a project that links to your GitHub repository.</td>
<td>April 5, 2023</td>
</tr>
<tr>
<td>Updated content: Working with notifications (p. 68)</td>
<td>Updated the documentation for notifications to include information about configuring emails about project events.</td>
<td>March 31, 2023</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Updated content: Working with roles (p. 476)</td>
<td>Updated the documentation for the Space administrator, Project administrator, and Contributor roles to include permissions for linking issues to pull requests.</td>
<td>March 13, 2023</td>
</tr>
<tr>
<td>Updated content: Working with pull requests (p. 100)</td>
<td>Updated the documentation for pull requests to include information about linking issues to pull requests.</td>
<td>March 13, 2023</td>
</tr>
<tr>
<td>Updated content: Working with issues (p. 459)</td>
<td>Updated the documentation for issues to include information about linking issues to pull requests.</td>
<td>March 13, 2023</td>
</tr>
<tr>
<td>New content: Viewing the status and details of all runs in your project (p. 316) (p. 315)</td>
<td>Added a section that describes the new aggregated workflow run page.</td>
<td>March 8, 2023</td>
</tr>
<tr>
<td>New content: How do I fix workflow definition errors? (p. 590) (p. 590)</td>
<td>Added a section on how to troubleshoot &quot;The workflow definition has errors&quot; errors.</td>
<td>March 7, 2023</td>
</tr>
<tr>
<td>Updated content: Creating, editing, and deleting a workflow (p. 246) (p. 246)</td>
<td>Updated the instructions to reflect new UI.</td>
<td>March 3, 2023</td>
</tr>
<tr>
<td>Updated content: Build, test, and deploy with workflows in CodeCatalyst (p. 128) (p. 128)</td>
<td>Updated various sections to reflect new source repository, branch, and workflow name filters on the Workflows summary page.</td>
<td>March 2, 2023</td>
</tr>
<tr>
<td>New content: Viewing code quality and deployment status by commit in CodeCatalyst (p. 137) (p. 137)</td>
<td>Added a section on viewing code quality and deployment status by commit.</td>
<td>February 27, 2023</td>
</tr>
<tr>
<td>New content: &quot;BranchName&quot; and &quot;CommitId&quot; variables (p. 347) (p. 346)</td>
<td>Added a new BranchName predefined variable.</td>
<td>February 16, 2023</td>
</tr>
<tr>
<td>Updated content: Managing space members in Amazon CodeCatalyst (p. 32)</td>
<td>Updated information about changing member roles, inviting members, and removing members in two new tables based on the user's assigned role in CodeCatalyst.</td>
<td>February 15, 2023</td>
</tr>
<tr>
<td>Updated content: Added steps for PAT management in the Amazon CodeCatalyst console (p. 497)</td>
<td>Added steps for viewing, creating, and deleting PATs in the console.</td>
<td>February 15, 2023</td>
</tr>
<tr>
<td>Updated content: Working with runtime environment Docker images (p. 302) (p. 302)</td>
<td>Added more tools to the Default image tool versions table.</td>
<td>January 10, 2023</td>
</tr>
<tr>
<td>Updated content: Working with artifacts (p. 286) (p. 286)</td>
<td>Fixed an artifact path.</td>
<td>January 3, 2023</td>
</tr>
<tr>
<td>Updated content: Working with sources (p. 325) (p. 325)</td>
<td>Fixed a source path.</td>
<td>January 3, 2023</td>
</tr>
<tr>
<td>Updated content: &quot;GitHub Actions&quot; action reference (p. 439) (p. 439)</td>
<td>Fixed the code snippet in the Steps section.</td>
<td>January 3, 2023</td>
</tr>
<tr>
<td>Updated content: Updating a pull request (p. 105)</td>
<td>Updated the documentation to include information about updating required or optional reviewers for a pull request.</td>
<td>December 23, 2022</td>
</tr>
<tr>
<td>New content: Working with file caching (p. 311) (p. 311)</td>
<td>Added a page for file caching in a workflow.</td>
<td>December 20, 2022</td>
</tr>
<tr>
<td>Updated content: Working with pull requests (p. 100)</td>
<td>Updated the documentation for pull requests to include information about notifications.</td>
<td>December 16, 2022</td>
</tr>
<tr>
<td>New content: &quot;AWS CDK deploy&quot; action reference (p. 393) (p. 393)</td>
<td>Added a new CdkRootPath property.</td>
<td>December 16, 2022</td>
</tr>
<tr>
<td>New content: Sharing compute across actions (p. 298) (p. 298)</td>
<td>Added a Sharing compute across actions (p. 298) topic.</td>
<td>December 14, 2022</td>
</tr>
<tr>
<td>Updated content: Working with artifacts (p. 286) (p. 286)</td>
<td>Fixed examples showing how to specify input artifacts.</td>
<td>December 13, 2022</td>
</tr>
<tr>
<td>Updated content: Quotas for projects in CodeCatalyst (p. 65)</td>
<td>Updated the documentation with a maximum of 100 projects in a space.</td>
<td>December 2, 2022</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>New content (p. 621)</td>
<td>Initial publication of the Amazon CodeCatalyst User Guide.</td>
<td>December 1, 2022</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS glossary in the AWS Glossary Reference.