aws

User Guide

AWS CodeStar

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodeStar: User Guide

Copyright © 2024 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

..... viii
What Is AWS CodeStar? 1
What Can I Do with AWS CodeStar? 1
How Do I Get Started with AWS CodeStar? 2
Setting Up 3
Step 1: Create an account 3
Sign up for an AWS account 3
Create a user with administrative access 4
Step 2: Create the AWS CodeStar Service Role 5
Step 3: Configure the User's IAM Permissions 5
Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects 6
Step 5: Open the AWS CodeStar Console 6
Next Steps 6
Getting Started with AWS CodeStar 7
Step 1: Create an AWS CodeStar Project 8
Step 2: Add display information for your AWS CodeStar User Profile 13
Step 3: View Your Project 13
Step 4: Commit a Change 14
Step 5: Add More Team Members 19
Step 6: Clean Up 22
Step 7: Get Your Project Ready for a Production Environment 22
Next Steps 23
Serverless Project Tutorial 23
Overview 24
Step 1: Create the Project 25
Step 2: Explore Project Resources 26
Step 3: Test the Web Service 29
Step 4: Set Up Your Local Workstation to Edit Project Code 30
Step 5: Add Logic to the Web Service 30
Step 6: Test the Enhanced Web Service 32
Step 7: Add a Unit Test to the Web Service 33
Step 8: View Unit Test Results 35
Step 9: Clean Up 36
Next Steps 37

AWS CLI Project Tutorial ... 37
Step 1: Download and Review the Sample Source Code ... 38
Step 2: Download the Sample Toolchain Template ... 39
Step 3: Test Your Toolchain Template in AWS CloudFormation ... 40
Step 4: Upload Your Source Code and Toolchain Template ... 41
Step 5: Create a Project in AWS CodeStar ... 42
Alexa Skill Project Tutorial ... 44
Prerequisites ... 45
Step 1: Create the project and connect your Amazon developer account ... 45
Step 2: Test your skill in the Alexa Simulator ... 47
Step 3: Explore your project resources ... 47
Step 4: Make a change in your skill's response ... 47
Step 5: Set up your local workstation to connect to your project repository ... 48
Next Steps ... 49
Tutorial: Create a Project with a GitHub Source Repository ... 49
Step 1: Create the project and create your GitHub repository ... 49
Step 2: View your source code ... 53
Step 3: Create a GitHub Pull Request ... 53
Project Templates ... 55
AWS CodeStar Project Files and Resources ... 55
Get Started: Choose a Project Template ... 57
Choose a Template Compute Platform ... 57
Choose a Template Application Type ... 57
Choose a Template Programming Language ... 59
How to Make Changes to Your AWS CodeStar Project ... 59
Change Application Source Code and Push Changes ... 59
Change Application Resources with the Template.yml File ... 60
... 60
AWS CodeStar Best Practices ... 61
Security Best Practices for AWS CodeStar Resources ... 61
Best Practices for Setting Versions for Dependencies ... 61
Monitoring and Logging Best Practices for AWS CodeStar Resources ... 62
Working with Projects ... 63
Create a Project ... 64
Create a Project in AWS CodeStar (Console) ... 65
Create a Project in AWS CodeStar (AWS CLI) ... 70

Use an IDE with AWS CodeStar ... 77
Use AWS Cloud9 with AWS CodeStar ... 78
Use Eclipse with AWS CodeStar ... 85
Use Visual Studio with AWS CodeStar ... 90
Change Project Resources ... 92
Supported Resource Changes ... 92
Add a Stage to AWS CodePipeline ... 94
Change AWS Elastic Beanstalk Environment Settings ... 94
Change an AWS Lambda Function in Source Code ... 95
Enable Tracing for a Project ... 95
Add a Resource to a Project ... 98
Add an IAM Role to a Project ... 104
Add a Prod Stage and Endpoint to a Project ... 104
Securely Use SSM Parameters in an AWS CodeStar Project ... 113
Shift Traffic for an AWS Lambda Project ... 115
Transition your AWS CodeStar Project to Production ... 122
Create a GitHub Repository ... 123
Working with Project Tags ... 124
Add a Tag to a Project ... 124
Remove a Tag from a Project ... 124
Get a List of Tags for a Project ... 125
Delete a Project ... 125
Delete a Project in AWS CodeStar (Console) ... 126
Delete a Project in AWS CodeStar (AWS CLI) ... 127
Working with Teams ... 129
Add Team Members to a Project ... 131
Add a Team Member (Console) ... 133
Add and View Team Members (AWS CLI) ... 134
Manage Team Permissions ... 136
Manage Team Permissions (Console) ... 137
Manage Team Permissions (AWS CLI) ... 137
Remove Team Members from a Project ... 138
Remove Team Members (Console) ... 139
Remove Team Members (AWS CLI) ... 139
Working with Your AWS CodeStar User Profile ... 141
Manage Display Information ... 141

Manage Your User Profile (Console) ... 142
Manage User Profiles (AWS CLI) ... 143
Add a Public Key to Your User Profile ... 145
Manage Your Public Key (Console) ... 146
Manage Your Public Key (AWS CLI) ... 147
Connect to Amazon EC2 Instance with Your Private Key ... 148
Security ... 149
Data Protection ... 150
Data Encryption in AWS CodeStar ... 151
Identity and Access Management ... 151
Audience ... 152
Authenticating With Identities ... 152
Managing Access Using Policies ... 155
How AWS CodeStar Works with IAM ... 157
AWS CodeStar Project-Level Policies and Permissions ... 168
Identity-Based Policy Examples ... 174
Troubleshooting ... 204
Logging AWS CodeStar API Calls with AWS CloudTrail ... 206
AWS CodeStar Information in CloudTrail ... 207
Understanding AWS CodeStar Log File Entries ... 207
Compliance Validation ... 209
Resilience ... 209
Infrastructure Security ... 210
Limits ... 211
Troubleshooting AWS CodeStar ... 213
Project creation failure: A project was not created ... 213
Project creation: I see an error when I try to edit Amazon EC2 configuration when creating a project ... 214
Project deletion: An AWS CodeStar project was deleted, but resources still exist ... 215
Team management failure: An IAM user could not be added to a team in an AWS CodeStar project ... 216
Access failure: A federated user cannot access an AWS CodeStar project ... 217
Access failure: A federated user cannot access or create an AWS Cloud9 environment ... 217
Access failure: A federated user can create an AWS CodeStar project, but cannot view project resources ... 217
Service role issue: The service role could not be created ... 218

Service role issue: The service role is not valid or missing 218
Project role issue: AWS Elastic Beanstalk health status checks fail for instances in an AWS CodeStar project 218
Project role issue: A project role is not valid or missing 219
Project extensions: Can't connect to JIRA 220
GitHub: Can't access a repository's commit history, issues, or code 220
AWS CloudFormation: Stack Creation Rolled Back for Missing Permissions 220
AWS CloudFormation is not authorized to perform iam:PassRole on Lambda execution role . 221
Unable to create the connection for a GitHub repository 222
Release Notes 223
AWS Glossary 228

On July 31, 2024, Amazon Web Services (AWS) will discontinue support for creating and viewing AWS CodeStar projects. After July 31, 2024, you will no longer be able to access the AWS CodeStar console or create new projects. However, the AWS resources created by AWS CodeStar, including your source repositories, pipelines, and builds, will be unaffected by this change and will continue to function. AWS CodeStar Connections and AWS CodeStar Notifications will not be impacted by this discontinuation.

If you wish to track work, develop code, and build, test, and deploy your applications, Amazon CodeCatalyst provides a streamlined getting started process and additional functionality to manage your software projects. Learn more about functionality and pricing of Amazon CodeCatalyst.

What Is AWS CodeStar?

AWS CodeStar is a cloud-based service for creating, managing, and working with software development projects on AWS. You can quickly develop, build, and deploy applications on AWS with an AWS CodeStar project. An AWS CodeStar project creates and integrates AWS services for your project development toolchain. Depending on your choice of AWS CodeStar project template, that toolchain might include source control, build, deployment, virtual servers or serverless resources, and more. AWS CodeStar also manages the permissions required for project users (called team members). By adding users as team members to an AWS CodeStar project, project owners can quickly and simply grant each team member role-appropriate access to a project and its resources.

Topics

- What Can I Do with AWS CodeStar?
- How Do I Get Started with AWS CodeStar?

What Can I Do with AWS CodeStar?

You can use AWS CodeStar to help you set up your application development in the cloud and manage your development from a single, centralized dashboard. Specifically, you can:

- **Start new software projects on AWS in minutes using templates for web applications, web services, and more:** AWS CodeStar includes project templates for various project types and programming languages. Because AWS CodeStar takes care of the setup, all of your project resources are configured to work together.
- **Manage project access for your team**: AWS CodeStar provides a central console where you can assign project team members the roles they need to access tools and resources. These permissions are applied automatically across all AWS services used in your project, so you don't need to create or manage complex IAM policies.
- **Visualize, operate, and collaborate on your projects in one place**: AWS CodeStar includes a project dashboard that provides an overall view of the project, its toolchain, and important events. You can monitor the latest project activity, like recent code commits, and track the status of your code changes, build results, and deployments, all from the same webpage. You can monitor what's going on in the project from a single dashboard and drill into problems to investigate.

- **Iterate quickly with all the tools you need**: AWS CodeStar includes an integrated development toolchain for your project. Team members push code, and changes are automatically deployed. Integration with issue tracking allows team members to keep track of what needs to be done next. You and your team can work together more quickly and efficiently across all phases of code delivery.

How Do I Get Started with AWS CodeStar?

To get started with AWS CodeStar:

1. **Prepare** to use AWS CodeStar by following the steps in Setting Up AWS CodeStar.
2. **Experiment** with AWS CodeStar by following the steps in the Getting Started with AWS CodeStar tutorial.
3. **Share** your project with other developers by following the steps in Add Team Members to an AWS CodeStar Project .
4. **Integrate** your favorite IDE by following the steps in Use an IDE with AWS CodeStar.

Setting Up AWS CodeStar

Before you can start using AWS CodeStar, you must complete the following steps.

Topics

- Step 1: Create an account
- Step 2: Create the AWS CodeStar Service Role
- Step 3: Configure the User's IAM Permissions
- Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects
- Step 5: Open the AWS CodeStar Console
- Next Steps

Step 1: Create an account

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create a user with administrative access

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

Secure your AWS account root user

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

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

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

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

Create a user with administrative access

1. Enable IAM Identity Center.

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

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

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

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

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

Step 2: Create the AWS CodeStar Service Role

Create a service role that is used to give AWS CodeStar permission to administer AWS resources and IAM permissions on your behalf. You only need to create the service role once.

> **Important**
>
> You must be signed in as an administrative user (or root account) to create a service role. For more information, see Creating Your First IAM user and Group.

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Start project**.

   If you do not see **Start project** and are directed to the projects list page instead, the service role has been created.
3. In **Create service role**, choose **Yes, create role**.
4. Exit the wizard. You come back to this later.

Step 3: Configure the User's IAM Permissions

In addition to the administrative user, you can use AWS CodeStar as an IAM user, a federated user, the root user, or an assumed role. For information about what AWS CodeStar can do for IAM users versus federated users, see AWS CodeStar IAM Roles.

If you have not set up any IAM users, see IAM user.

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

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

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

Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects

Many AWS CodeStar projects use AWS CodeDeploy or AWS Elastic Beanstalk to deploy code to Amazon EC2 instances. To access Amazon EC2 instances associated with your project, create an Amazon EC2 key pair for your IAM user. Your IAM user must have permissions to create and manage Amazon EC2 keys (for example, permission to take the `ec2:CreateKeyPair` and `ec2:ImportKeyPair` actions). For more information, see Amazon EC2 Key Pairs.

Step 5: Open the AWS CodeStar Console

Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.

Next Steps

Congratulations, you have completed the setup! To start working with AWS CodeStar, see Getting Started with AWS CodeStar.

Getting Started with AWS CodeStar

In this tutorial, you use AWS CodeStar to create a web application. This project includes sample code in a source repository, a continuous deployment toolchain, and a project dashboard where you can view and monitor your project.

By following the steps, you:

- Create a project in AWS CodeStar.
- Explore the project.
- Commit a code change.
- See your code change deployed automatically.
- Add other people to work on your project.
- Clean up project resources when they're no longer needed.

> **Note**
>
> If you haven't already, first complete the steps in Setting Up AWS CodeStar, including Step 2: Create the AWS CodeStar Service Role. You must be signed in with an account that is an administrative user in IAM. To create a project, you must sign in to the AWS Management Console using an IAM user that has the **`AWSCodeStarFullAccess`** policy.

Topics

- Step 1: Create an AWS CodeStar Project
- Step 2: Add display information for your AWS CodeStar User Profile
- Step 3: View Your Project
- Step 4: Commit a Change
- Step 5: Add More Team Members
- Step 6: Clean Up
- Step 7: Get Your Project Ready for a Production Environment
- Next Steps
- Tutorial: Creating and Managing a Serverless Project in AWS CodeStar

- Tutorial: Create a Project in AWS CodeStar with the AWS CLI
- Tutorial: Create an Alexa Skill Project in AWS CodeStar
- Tutorial: Create a Project with a GitHub Source Repository

Step 1: Create an AWS CodeStar Project

In this step, you create a JavaScript (Node.js) software development project for a web application. You use an AWS CodeStar project template to create the project.

> **Note**
>
> The AWS CodeStar project template used in this tutorial uses the following options:
>
> - **Application category**: Web application
> - **Programming language**: Node.js
> - **AWS Service**: Amazon EC2
>
> If you choose other options, your experience might not match what's documented in this tutorial.

To create a project in AWS CodeStar

1. Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.

   Make sure that you are signed in to the AWS Region where you want to create the project and its resources. For example, to create a project in US East (Ohio), make sure you have selected that AWS Region. For information about AWS Regions where AWS CodeStar is available, see Regions and Endpoints in the *AWS General Reference* .
2. On the **AWS CodeStar** page, choose **Create project**.
3. On the **Choose a project template** page, choose the project type from the list of AWS CodeStar project templates. You can use the filter bar to narrow your choices. For example, for a web application project written in Node.js to be deployed to Amazon EC2 instances, select the **Web application**, **Node.js**, and **Amazon EC2** check boxes. Then choose from the templates available for that set of options.

For more information, see AWS CodeStar Project Templates.

4. Choose **Next**.
5. In the **Project name** text input field, enter a name for the project, such as *My First Project*. In **Project ID**, the ID for the project is derived from this project name, but is limited to 15 characters.

   For example, the default ID for a project named *My First Project* is *my-first-projec*. This project ID is the basis for the names of all resources associated with the project. AWS CodeStar uses this project ID as part of the URL for your code repository and for the names of related security access roles and policies in IAM. After the project is created, the project ID cannot be changed. To edit the project ID before you create the project, in **Project ID**, enter the ID you want to use.

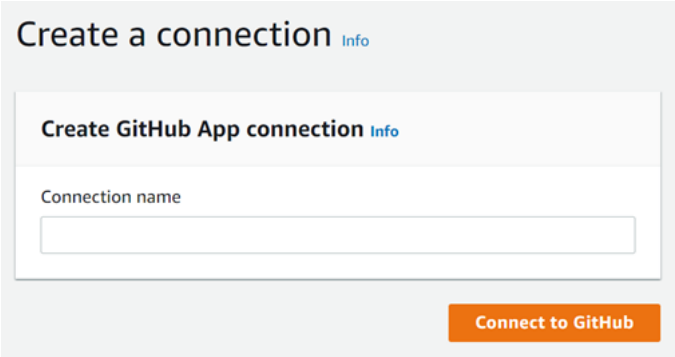
   For information about the limits on project names and project IDs, see Limits in AWS CodeStar.

   > **Note**
   >
   > Project IDs must be unique for your AWS account in an AWS Region.

6. Choose the repository provider, **AWS CodeCommit** or **GitHub**.
7. If you chose **AWS CodeCommit**, for **Repository name**, accept the default AWS CodeCommit repository name, or enter a different one. Then skip ahead to step 9.
8. If you chose **GitHub**, you need to choose or create a connection resource. If you have an existing connection, choose it in the search field. Otherwise, create a new connection now. Choose **Connect to GitHub**.

   The **Create a connection** page displays.

   > **Note**
   >
   > To create a connection, you must have a GitHub account. If you are creating a connection for an organization, you must be the organization owner.



a. Under **Create GitHub App connection**, in the **Connection name** input text field, enter a name for your connection. Choose **Connect to GitHub**.

   The **Connect to GitHub** page displays and shows the **GitHub Apps** field.

b. Under **GitHub Apps**, choose an app installation or choose **Install a new app** to create one.

> **Note**
>
> You install one app for all of your connections to a particular provider. If you have already installed the AWS Connector for GitHub app, choose it and skip this step.

c. On the **Install AWS Connector for GitHub** page, choose the account where you want to install the app.

> **Note**
>
> If you previously installed the app, you can choose **Configure** to proceed to a modification page for your app installation, or you can use the back button to return to the console.

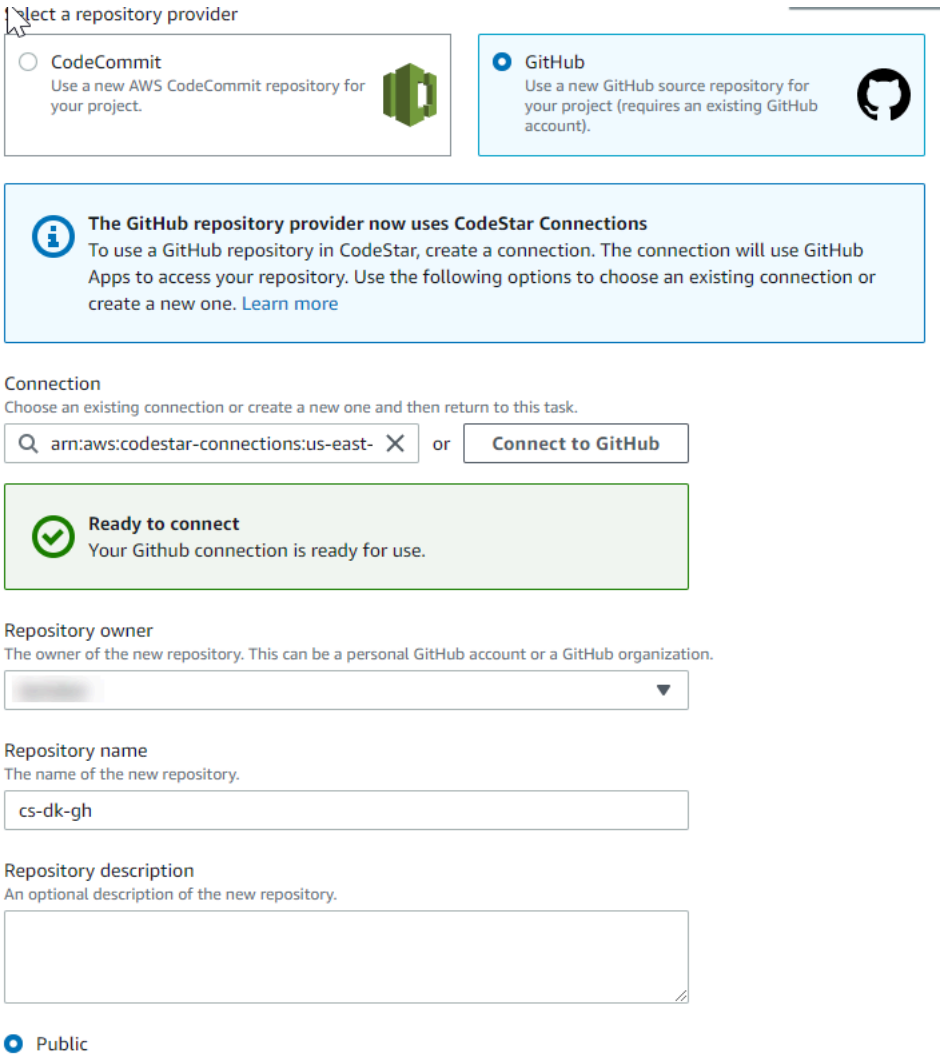
d. If the **Confirm password to continue** page is displayed, enter your GitHub password, and then choose **Sign in**.

e. On the **Install AWS Connector for GitHub** page, keep the defaults, and choose **Install**.

f. On the **Connect to GitHub** page, the installation ID for your new installation appears in the **GitHub Apps** text input field.

   After the connection is created, in the CodeStar create project page, the message **Ready to connect** displays.

> **Note**
>
> You can view your connection under **Settings** in the **Developer Tools** console. For more information, see Getting started with connections.



g. For **Repository owner**, choose the GitHub organization or your personal GitHub account.

h. For **Repository name**, accept the default GitHub repository name, or enter a different one.

i. Choose **Public** or **Private**.

> **Note**
>
> To use AWS Cloud9 as your development environment, you must choose **Public**.

j. (Optional) For **Repository description**, enter a description for the GitHub repository.

> **Note**
>
> If you choose an Alexa Skill project template, you need to connect an Amazon developer account. For more information about working with Alexa Skill projects, see Tutorial: Create an Alexa Skill Project in AWS CodeStar.

9. If your project is deployed to Amazon EC2 instances and you want to make changes, configure your Amazon EC2 instances in **Amazon EC2 Configuration**. For example, you can choose from available instance types for your project.

> **Note**
>
> Different Amazon EC2 instance types provide different levels of computing power and might have different associated costs. For more information, see Amazon EC2 Instance Types and Amazon EC2 Pricing.
> If you have more than one virtual private cloud (VPC) or multiple subnets created in Amazon Virtual Private Cloud, you can also choose the VPC and subnet to use. However, if you choose an Amazon EC2 instance type that is not supported on dedicated instances, you cannot choose a VPC whose instance tenancy is set to **Dedicated**.
> For more information, see What Is Amazon VPC? and Dedicated Instance Basics.

In **Key pair**, choose the Amazon EC2 key pair you created in Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects. Select **I acknowledge that I have access to the private key file**.

10. Choose **Next**.
11. Review the resources and configuration details.
12. Choose **Next** or **Create project**. (The displayed choice depends on your project template.)

It might take a few minutes to create the project, including the repository.

13. After your project has a repository, you can use the **Repository** page to configure access to it. Use the links in **Next steps** to configure an IDE, set up issue tracking, or add team members to your project.

Step 2: Add display information for your AWS CodeStar User Profile

When you create a project, you're added to the project team as an owner. If this is the first time you've used AWS CodeStar, you are asked to provide:

- Your display name to show to other users.
- The email address to show to other users.

This information is used in your AWS CodeStar user profile. User profiles are not project-specific, but are limited to an AWS Region. You must create a user profile in each AWS Region in which you belong to projects. Each profile can contain different information, if you prefer.

Enter a user name and email address, and then choose **Next**.

> **Note**
>
> This user name and email address is used in your AWS CodeStar user profile. If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), those resource providers might have their own user profiles, with different user names and email addresses. For more information, see the resource provider's documentation.

Step 3: View Your Project

Your AWS CodeStar project page is where you and your team view the status of your project resources, including the latest commits to your project, the state of your continuous delivery pipeline, and the performance of your instances. To see more information about any of these resources, choose the corresponding page from the navigation bar.

In your new project, the navigation bar contains the following pages:

- The **Overview** page contains information about your project's activity, project resources, and your project's README contents.
- The **IDE** page is where you connect your project to an integrated development environment (IDE) to modify, test, and push source code changes. It contains instructions for configuring IDEs for both GitHub and AWS CodeCommit repositories and information about your AWS Cloud9 environments.
- The **Repository** page displays your repository details, including the name, provider, when it was last modified, and clone URLs. You can also see information about the most recent commit and view and create pull requests.
- The **Pipeline** page displays CI/CD information about your pipeline. You can view pipeline details such as name, most recent action, and status. You can see the history of the pipeline and release a change. You can also view the status of the individual steps of your pipeline.
- The **Monitoring** page displays either Amazon EC2 or AWS Lambda metrics depending on your project's configuration. For example, it displays the CPU utilization of any Amazon EC2 instances deployed to by AWS Elastic Beanstalk or CodeDeploy resources in your pipeline. In projects that use AWS Lambda, it displays invocation and error metrics for the Lambda function. This information is displayed by the hour. If you used the suggested AWS CodeStar project template for this tutorial, you should see a noticeable spike in activity as your application is first deployed to those instances. You can refresh monitoring to see changes in your instance health, which can help you identify problems or the need for more resources.
- The **Issues** page is for integrating your AWS CodeStar project with an Atlassian JIRA project. Configuring this tile makes it possible for you and your project team to track JIRA issues from the project dashboard.

The navigation pane on the left side of the console is where you can navigate between your **Project**, **Team**, and **Settings** pages.

Step 4: Commit a Change

First, take a look at the sample application that was included in your project. See what the application looks like by choosing **View application** from anywhere in your project navigation. Your sample web application will be displayed in a new window or browser tab. This is the project sample that AWS CodeStar built and deployed.

If you'd like to look at the code, in the navigation bar choose **Repository**. Choose the link under **Repository name** and your project's repository opens in a new tab or window. Read the contents of the repository's readme file (`README.md`), and browse the content of those files.

In this step, you make a change to the code and then push the change to your repository. You can do this in one of several ways:

- If the project's code is stored in a CodeCommit or GitHub repository, you can use AWS Cloud9 to work with the code directly from your web browser, without installing any tools. For more information, see Create an AWS Cloud9 Environment for a Project.
- If the project's code is stored in a CodeCommit repository, and you have Visual Studio or Eclipse installed, you can use the AWS Toolkit for Visual Studio or AWS Toolkit for Eclipse to more easily connect to the code. For more information, see Use an IDE with AWS CodeStar. If you don't have Visual Studio or Eclipse, install a Git client, and follow the instructions later in this step.
- If the project's code is stored in a GitHub repository, you can use your IDE's tools for connecting to GitHub.
  - For Visual Studio, you can use tools such as the GitHub Extension for Visual Studio. For more information, see the Overview page on the GitHub Extension for Visual Studio website and Getting Started with GitHub for Visual Studio on the GitHub website.
  - For Eclipse, you can use a tool such as EGit for Eclipse. For more information, see the EGit Documentation on the EGit website.
  - For other IDEs, consult your IDE's documentation.
- For other types of code repositories, see the repository provider's documentation.

The following instructions show you how to make a minor change to the sample.

To set up your computer to commit changes (IAM user)

> **Note**
>
> In this procedure, we assume that your project's code is stored in a CodeCommit repository. For other types of code repositories, see the repository provider's documentation, and then skip ahead to the next procedure, To clone the project repository and make a change.
> If the code is stored in CodeCommit and you are already using CodeCommit or you used the AWS CodeStar console to create an AWS Cloud9 development environment for the project,

you don't need more configuration. Skip ahead to the next procedure, To clone the project repository and make a change.

1. Install Git on your local computer.
2. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Sign in as the IAM user who will use Git credentials for connections to your AWS CodeStar project repository in CodeCommit.
3. In the IAM console, in the navigation pane, choose **Users**, and from the list of users, choose your IAM user.
4. On the user details page, choose the **Security Credentials** tab, and in **HTTPS Git credentials for CodeCommit**, choose **Generate**.

   > **Note**
   >
   > You cannot choose your own sign-in credentials for Git credentials. For more information, see Use Git Credentials and HTTPS with CodeCommit.

5. Copy the sign-in credentials that IAM generated for you. You can choose **Show** and then copy and paste this information into a secure file on your local computer, or you can choose **Download credentials** to download this information as a .CSV file. You need this information to connect to CodeCommit.

   After you have saved your credentials, choose **Close**.

   > **Important**
   >
   > This is your only chance to save the sign-in credentials. If you do not save them, you can copy the user name from the IAM console, but you cannot look up the password. You must reset the password and then save it.

To set up your computer to commit changes (federated user)

You can use the console to upload files to your repository, or you can use Git to connect from your local computer. If you are using federated access, follow these steps to use Git to connect to and clone your repository from your local computer.

> **Note**
>
> In this procedure, we assume that your project's code is stored in a CodeCommit repository. For other types of code repositories, see the repository provider's documentation, and then skip ahead to the next procedure, To clone the project repository and make a change.

1. Install Git on your local computer.
2. Install the AWS CLI.
3. Configure your temporary security credentials for a federated user. For information, see Temporary Access to CodeCommit Repositories. Temporary credentials consist of:

   - AWS access key
   - AWS secret key
   - Session token

   For more information about temporary credentials, see Permissions for GetFederationToken.
4. Connect to your repository using the AWS CLI credential helper. For information, see Setup Steps for HTTPS Connections to CodeCommit Repositories on Linux, macOS, or Unix with the AWS CLI Credential Helper or Setup Steps for HTTPS Connections to CodeCommit Repositories on Windows with the AWS CLI Credential Helper
5. The following example shows how to connect to a CodeCommit repository and push a commit to it.

Example: To clone the project repository and make a change

> **Note**
>
> This procedure shows how to clone the project's code repository to your computer, make a change to the project's `index.html` file, and then push your change to the remote

repository. In this procedure, we assume that your project's code is stored in a CodeCommit repository and that you're using a Git client from the command line. For other types of code repositories or tools, see the provider's documentation for how to clone the repository, change the file, and then push the code.

1. If you used the AWS CodeStar console to create an AWS Cloud9 development environment for the project, open the development environment, and then skip to step 3 in this procedure. To open the development environment, see Open an AWS Cloud9 Environment for a Project.

   With your project open in the AWS CodeStar console, on the navigation bar, choose **Repository**. In **Clone URL**, choose the protocol for the connection type you have set up for CodeCommit, and then copy the link. For example, if you followed the steps in the previous procedure to set up Git credentials for CodeCommit, choose **HTTPS**.

2. On your local computer, open a terminal or command line window and change directories to a temporary directory. Run the **git clone** command to clone the repository to your computer. Paste the link you copied. For example, for CodeCommit using HTTPS:

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-first-projec
   ```

   The first time you connect, you are prompted for the sign-in credentials for the repository. For CodeCommit, enter the Git credentials sign-in credentials you downloaded in the previous procedure.

3. Navigate to the cloned directory on your computer and browse the contents.

4. Open the `index.html` file (in the public folder) and make a change to the file. For example, add a paragraph after the <H2> tag such as:

   ```
   <P>Hello, world!</P>
   ```

   Save the file.

5. At the terminal or command prompt, add your changed file, and then commit and push your change:

   ```
   git add index.html
   git commit -m "Making my first change to the web app"
   git push
   ```

6. On the **Repository** page, view the changes in progress. You should see that the commit history for the repository is updated with your commit, including the commit message. In the **Pipeline** page, you can see the pipeline pick up your change to the repository and start building and deploying it. After your web application is deployed, you can choose **View application** to view your change.

   Note

   If **Failed** is displayed for any of the pipeline stages, see the following for troubleshooting help:

   - For the **Source** stage, see Troubleshooting AWS CodeCommit in the *AWS CodeCommit User Guide*.
   - For the **Build** stage, see Troubleshooting AWS CodeBuild in the *AWS CodeBuild User Guide*.
   - For the **Deploy** stage, see Troubleshooting AWS CloudFormation in the *AWS CloudFormation User Guide*.
   - For other issues, see Troubleshooting AWS CodeStar.

Step 5: Add More Team Members

Every AWS CodeStar project is already configured with three AWS CodeStar roles. Each role provides its own level of access to the project and its resources:

- **Owner**: Can add and remove team members, change the project dashboard, and delete the project.
- **Contributor**: Can change the project dashboard and contribute code if the code is stored in CodeCommit, but cannot add or remove team members or delete the project. This is the role you should choose for most team members in an AWS CodeStar project.
- **Viewer**: Can view the project dashboard, project code if the code is stored in CodeCommit, and the state of the project, but cannot move, add, or remove tiles from the project dashboard.

Important

If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), access to those resources is controlled by the resource provider, not AWS CodeStar. For more information, see the resource provider's documentation.
Anyone who has access to an AWS CodeStar project might be able to use the AWS CodeStar console to access resources that are outside of AWS but are related to the project.
AWS CodeStar does not allow project team members to participate in any related AWS Cloud9 development environments for a project. To allow a team member to participate in a shared environment, see Share an AWS Cloud9 Environment with a Project Team Member.

For more information about teams and project roles, see Working with AWS CodeStar Teams.

To add a team member to an AWS CodeStar project (console)

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose your project.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose **Add team member**.
5. In **Choose user**, do one of the following:

   - If an IAM user already exists for the person you want to add, choose the IAM user from the list.

     Note

     Users who have already been added to another AWS CodeStar project appear in the **Existing AWS CodeStar users** list.

     In **Project role**, choose the AWS CodeStar role (Owner, Contributor, or Viewer) for this user. This is an AWS CodeStar project-level role that can only be changed by an owner of the project. When applied to an IAM user, the role provides all permissions required to access AWS CodeStar project resources. It applies policies required for creating and managing Git credentials for code stored in CodeCommit in IAM or uploading Amazon EC2 SSH keys for the user in IAM.

> **Important**
>
> You cannot provide or change the display name or email information for an IAM user unless you are signed in to the console as that user. For more information, see Manage Display Information for Your AWS CodeStar User Profile .

Choose **Add team member**.

- If an IAM user does not exist for the person you want to add to the project, choose **Create new IAM user**. You will be redirected to the IAM console where you can create a new IAM user, see Creating IAM users in the *IAM user guide* for more information. After you create your IAM user, return to the AWS CodeStar console, refresh the list of users, and choose the IAM user you created from the dropdown list. Enter the AWS CodeStardisplay name, email address, and project role you want to apply to this new user, and then choose **Add team member**.

> **Note**
>
> For ease of management, at least one user should be assigned the Owner role for the project.

6. Send the new team member the following information:

   - Connection information for your AWS CodeStar project.
   - If the source code is stored in CodeCommit, instructions for setting up access with Git credentials to the CodeCommit repository from their local computers.
   - Information about how the user can manage their display name, email address, and public Amazon EC2 SSH key, as described in Working with Your AWS CodeStar User Profile .
   - One-time password and connection information, if the user is new to AWS and you created an IAM user for that person. The password expires the first time the user signs in. The user must choose a new password.

Step 6: Clean Up

Congratulations! You've finished the tutorial. If you don't want to continue to use this project and its resources, you should delete it to avoid possible continued charges to your AWS account.

To delete a project in AWS CodeStar

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** in the navigation pane.
3. Select the project you want to delete and choose **Delete**.

   Or, open the project and choose **Settings** from the navigation pane on the left side of the console. On the project details page, choose **Delete project**.
4. In the **Delete confirmation page**, enter *delete*. Keep **Delete resources** selected if you wish to delete project resources. Choose **Delete**.

   Deleting a project can take several minutes. After it's deleted, the project no longer appears in the list of projects in the AWS CodeStar console.

   > **Important**
   >
   > If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), those resources are not deleted, even if you select the check box.
   > Your project cannot be deleted if any AWS CodeStar managed policies have been manually attached to roles that are not IAM users. If you have attached your project's managed policies to a federated user's role, you must detach the policy before you can delete the project. For more information, see ???.

Step 7: Get Your Project Ready for a Production Environment

After you have created your project, you are ready to create, test, and deploy code. Review the following considerations for maintaining your project in a production environment:

- Regularly apply patches and review security best practices for the dependencies used by your application. For more information, see Security Best Practices for AWS CodeStar Resources.

- Regularly monitor the environment settings suggested by the programming language for your project.

Next Steps

Here are some other resources to help you learn about AWS CodeStar:

- The Tutorial: Creating and Managing a Serverless Project in AWS CodeStar uses a project that creates and deploys a web service using logic in AWS Lambda and can be called by an API in Amazon API Gateway.
- AWS CodeStar Project Templates describes other types of projects you can create.
- Working with AWS CodeStar Teams provides information about enabling others to help you work on your projects.

Tutorial: Creating and Managing a Serverless Project in AWS CodeStar

In this tutorial, you use AWS CodeStar to create a project that uses the AWS Serverless Application Model (AWS SAM) to create and manage AWS resources for a web service hosted in AWS Lambda.

AWS CodeStar uses AWS SAM, which relies on AWS CloudFormation, to provide a simplified way of creating and managing supported AWS resources, including Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables. (This project does not use any Amazon DynamoDB tables.)

For more information, see AWS Serverless Application Model (AWS SAM) on GitHub.

Prerequisite: Complete the steps in Setting Up AWS CodeStar.

> **Note**
>
> Your AWS account might be charged for costs related to this tutorial, including costs for AWS services used by AWS CodeStar. For more information, see AWS CodeStar Pricing.

Topics

- Overview

- Step 1: Create the Project
- Step 2: Explore Project Resources
- Step 3: Test the Web Service
- Step 4: Set Up Your Local Workstation to Edit Project Code
- Step 5: Add Logic to the Web Service
- Step 6: Test the Enhanced Web Service
- Step 7: Add a Unit Test to the Web Service
- Step 8: View Unit Test Results
- Step 9: Clean Up
- Next Steps

Overview

In this tutorial, you:

1. Use AWS CodeStar to create a project that uses AWS SAM to build and deploy a Python-based web service. This web service is hosted in AWS Lambda and can be accessed through Amazon API Gateway.
2. Explore the project's main resources, which include:
   - The AWS CodeCommit repository where the project's source code is stored. This source code includes the web service's logic and defines related AWS resources.
   - The AWS CodePipeline pipeline that automates the building of the source code. This pipeline uses AWS SAM to create and deploy a function to AWS Lambda, create a related API in Amazon API Gateway, and connect the API to the function.
   - The function that is deployed to AWS Lambda.
   - The API that is created in Amazon API Gateway.
3. Test the web service to confirm that AWS CodeStar built and deployed the web service as expected.
4. Set up your local workstation to work with the project's source code.
5. Change the project's source code using your local workstation. When you add a function to the project and then push your changes to the source code, AWS CodeStar rebuilds and redeploys the web service.
6. Test the web service again to confirm that AWS CodeStar rebuilt and redeployed as expected.

7. Write a unit test using your local workstation to replace some of your manual testing with an automated test. When you push the unit test, AWS CodeStar rebuilds and redeploys the web service and runs the unit test.
8. View the results of the unit tests.
9. Clean up the project. This step helps you avoid charges to your AWS account for costs related to this tutorial.

Step 1: Create the Project

In this step, you use the AWS CodeStar console to create a project.

1. Sign in to the AWS Management Console and open the AWS CodeStar console, at https://console.aws.amazon.com/codestar/.

   > **Note**
   >
   > You must sign in to the AWS Management Console using credentials associated with the IAM user you created or identified in Setting Up AWS CodeStar. This user must have the **`AWSCodeStarFullAccess`** managed policy attached.

2. Choose the AWS Region where you want to create the project and its resources.

   For information about AWS Regions where AWS CodeStar is available, see Regions and Endpoints in the *AWS General Reference*.
3. Choose **Create project**.
4. On the **Choose a project template** page:

   - For **Application type**, select **Web service**.
   - For **Programming language**, select **Python**.
   - For **AWS service**, select **AWS Lambda**.
5. Choose the box that contains your selections. Choose **Next**.
6. For **Project name**, enter a name for the project (for example, **`My SAM Project`**). If you use a name different from the example, be sure to use it throughout this tutorial.

   For **Project ID**, AWS CodeStar chooses a related identifier for this project (for example, **my-sam-project**). If you see a different project ID, be sure to use it throughout this tutorial.

Leave **AWS CodeCommit** selected, and do not change the **Repository name** value.

7. Choose **Next**.
8. Review your settings and then choose **Create Project**.

   If this is your first time using AWS CodeStar in this AWS Region, for **Display Name** and **Email**, enter the display name and email address you want AWS CodeStar to use for your IAM user. Choose **Next**.
9. Wait while AWS CodeStar creates the project. This might take several minutes. Do not continue until you see the **Project provisioned** banner when you refresh.

Step 2: Explore Project Resources

In this step, you explore four of the project's AWS resources to understand how the project works:

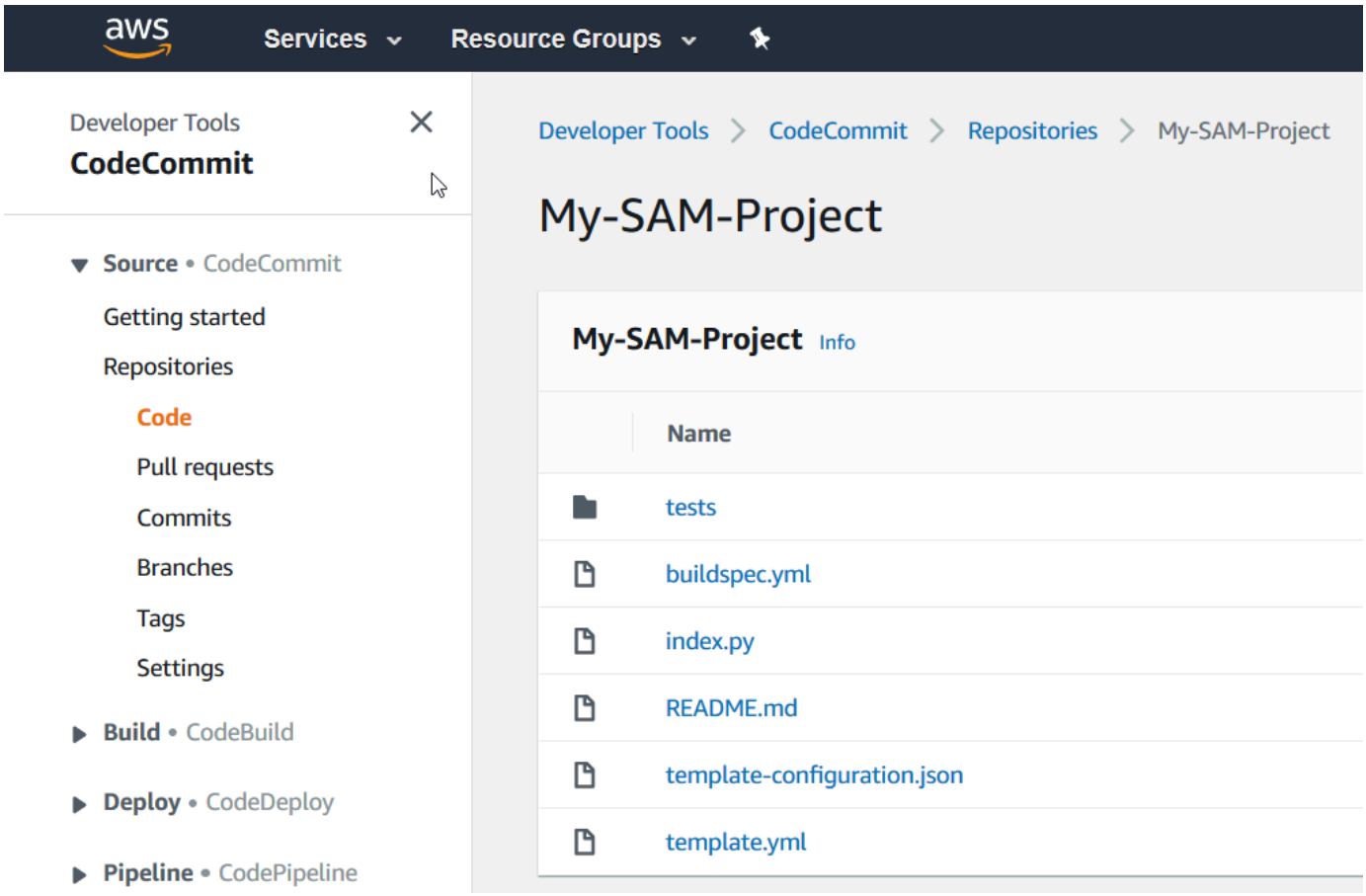
- The AWS CodeCommit repository where the project's source code is stored. AWS CodeStar gives the repository the name **my-sam-project**, where **my-sam-project** is the name of the project.
- The AWS CodePipeline pipeline that uses CodeBuild and AWS SAM to automate building and deploying the web service's Lambda function and API in API Gateway. AWS CodeStar gives the pipeline the name **my-sam-project--Pipeline**, where **my-sam-project** is the ID of the project.
- The Lambda function that contains the logic of the web service. AWS CodeStar gives the function the name **awscodestar-my-sam-project-lambda-HelloWorld-*RANDOM_ID***, where:
  - **my-sam-project** is the ID of the project.
  - **HelloWorld** is the function ID as specified in the `template.yaml` file in the AWS CodeCommit repository. You explore this file later.
  - *RANDOM_ID* is a random ID that AWS SAM assigns to the function to help ensure uniqueness.
- The API in API Gateway that makes it easier to call the Lambda function. AWS CodeStar gives the API the name **awscodestar-my-sam-project--lambda**, where **my-sam-project** is the ID of the project.

To explore the source code repository in CodeCommit

1. With your project open in the AWS CodeStar console, on the navigation bar, choose **Repository**.
2. Choose the link to your CodeCommit repository (**`My-SAM-Project`**) in **Repository details**.

3. In the CodeCommit console, on the **Code** page, the source code files for the project are displayed:

   - `buildspec.yml`, which CodePipeline instructs CodeBuild to use during the build phase, to package the web service using AWS SAM.
   - `index.py`, which contains the logic for the Lambda function. This function simply outputs the string `Hello World` and a timestamp, in ISO format.
   - `README.md`, which contains general information about the repository.
   - `template-configuration.json`, which contains the project ARN with placeholders used for tagging resources with the project ID
   - `template.yml`, which AWS SAM uses to package the web service and create the API in API Gateway.



To view the contents of a file, choose it from the list.

For more information about using the CodeCommit console, see the AWS CodeCommit User Guide.

To explore the pipeline in CodePipeline

1. To view information about the pipeline, with your project open in the AWS CodeStar console, on the navigation bar, choose **Pipeline** and you see the pipeline contains:
   - A **Source** stage for getting the source code from CodeCommit.
   - A **Build** stage for building the source code with CodeBuild.
   - A **Deploy** stage for deploying the built source code and AWS resources with AWS SAM.
2. To view more information about the pipeline, in **Pipeline details**, choose your pipeline to open the pipeline in the CodePipeline console.

For information about using the CodePipeline console, see the AWS CodePipeline User Guide.

To explore project activity and AWS service resources on the Overview page

1. Open your project in the AWS CodeStar console and from the navigation bar, choose **Overview**.
2. Review the **Project activity** and **Project resources** lists.

To explore the function in Lambda

1. With your project open in the AWS CodeStar console, on the side navigation bar, choose **Overview**.
2. In **Project resources**, in the **ARN** column, choose the link for the Lambda function.

   The function's code is displayed in the Lambda console.

For information about using the Lambda console, see the AWS Lambda Developer Guide.

To explore the API in API Gateway

1. With your project open in the AWS CodeStar console, on the side navigation bar, choose **Overview**.

2. In **Project resources**, in the **ARN** column, choose the link for the Amazon API Gateway API.

   Resources for the API are displayed in the API Gateway console.

For information about using the API Gateway console, see the API Gateway Developer Guide.

Step 3: Test the Web Service

In this step, you test the web service that AWS CodeStar just built and deployed.

1. With your project still open from the previous step, on the navigation bar, choose **Pipeline**.
2. Make sure **Succeeded** is displayed for the **Source**, **Build**, and **Deploy** stages before you continue. This might take several minutes.

   > **Note**
   >
   > If **Failed** is displayed for any of the stages, see the following for troubleshooting help:
   >
   > - For the **Source** stage, see Troubleshooting AWS CodeCommit in the *AWS CodeCommit User Guide*.
   > - For the **Build** stage, see Troubleshooting AWS CodeBuild in the *AWS CodeBuild User Guide*.
   > - For the **Deploy** stage, see Troubleshooting AWS CloudFormation in the *AWS CloudFormation User Guide*.
   > - For other issues, see Troubleshooting AWS CodeStar.

3. Choose **View Application**.

On the new tab that opens in your web browser, the web service displays the following response output:

```
{"output": "Hello World", "timestamp": "2017-08-30T15:53:42.682839"}
```

Step 4: Set Up Your Local Workstation to Edit Project Code

In this step, you set up your local workstation to edit the source code in the AWS CodeStar project. Your local workstation can be a physical or virtual computer running macOS, Windows, or Linux.

1. With your project still open from the previous step:

   - In the navigation bar, choose **IDE**, and then expand **Access your project code**.
   - Choose **View instructions** underneath **Command line interface**.

     If you have Visual Studio or Eclipse installed, choose **View instructions** underneath **Visual Studio** or **Eclipse** instead, follow the instructions, and then skip to Step 5: Add Logic to the Web Service.
2. Follow the instructions to complete the following tasks:

   a. Set up Git on your local workstation.

   b. Use the IAM console to generate Git credentials for your IAM user.

   c. Clone the project's CodeCommit repository onto your local workstation.
3. In the left navigation, choose **Project** to return to your project overview.

Step 5: Add Logic to the Web Service

In this step, you use your local workstation to add logic to the web service. Specifically, you add a Lambda function and then connect it to the API in API Gateway.

1. On your local workstation, go to the directory that contains the cloned source code repository.
2. In that directory, create a file named `hello.py`. Add the following code, and then save the file:

```
import json

def handler(event, context):
  data = {
    'output': 'Hello ' + event["pathParameters"]["name"]
  }
  return {
    'statusCode': 200,
    'body': json.dumps(data),
```

```
    'headers': {'Content-Type': 'application/json'}
}
```

The preceding code outputs the string `Hello` and the string the caller sends to the function.

3. In the same directory, open the `template.yml` file. Add the following code to the end of the file, and then save the file:

```
Hello:
  Type: AWS::Serverless::Function
  Properties:
    FunctionName: !Sub 'awscodestar-${ProjectId}-lambda-Hello'
    Handler: hello.handler
    Runtime: python3.7
    Role:
      Fn::GetAtt:
      - LambdaExecutionRole
      - Arn
    Events:
      GetEvent:
        Type: Api
        Properties:
          Path: /hello/{name}
          Method: get
```

AWS SAM uses this code to create a function in Lambda, add a new method and path to the API in API Gateway, and then connect this method and path to the new function.

> **Note**
>
> The indentation of the preceding code is important. If you don't add the code exactly as it's shown, the project might not build correctly.

4. Run **git add .** to add your file changes to the staging area of the cloned repository Do not forget the period (**.**), which adds all changed files.

> **Note**
>
> If you are using Visual Studio or Eclipse instead of the command line, the instructions for using Git might be different. See the Visual Studio or Eclipse documentation.

5. Run **git commit -m "Added hello.py and updated template.yaml."** to commit your staged files in the cloned repository
6. Run **git push** to push your commit to the remote repository.

> **Note**
>
> You might be prompted for the sign-in credentials generated for you earlier. To keep from being prompted each time you interact with the remote repository, consider installing and configuring a Git credential manager. For example, on macOS or Linux, you can run **git config credential.helper 'cache --timeout 900'** in the terminal to be prompted no sooner than every 15 minutes. Or you can run **git config credential.helper 'store --file ~/.git-credentials'** to never be prompted again. Git stores your credentials in clear text in a plain file in your home directory. For more information, see Git Tools - Credential Storage on the Git website.

After AWS CodeStar detects the push, it instructs CodePipeline to use CodeBuild and AWS SAM to rebuild and redeploy the web service. You can watch the deployment progress on the **Pipeline** page.

AWS SAM gives the new function the name **awscodestar-my-sam-project-lambda-Hello-*RANDOM_ID***, where:

- **my-sam-project** is the ID of the project.
- **Hello** is the function ID, as specified in the `template.yaml` file.
- *RANDOM_ID* is a random ID that AWS SAM assigns to the function for uniqueness.

Step 6: Test the Enhanced Web Service

In this step, you test the enhanced web service that AWS CodeStar built and deployed, based on the logic you added in the previous step.

1. With your project still open in the AWS CodeStar console, on the navigation bar, choose **Pipeline**.
2. Make sure the pipeline has run again and that **Succeeded** is displayed for the **Source**, **Build**, and **Deploy** stages before you continue. This might take several minutes.

> **Note**
>
> If **Failed** is displayed for any of the stages, see the following for troubleshooting help:
>
> - For the **Source** stage, see Troubleshooting AWS CodeCommit in the *AWS CodeCommit User Guide*.
> - For the **Build** stage, see Troubleshooting AWS CodeBuild in the *AWS CodeBuild User Guide*.
> - For the **Deploy** stage, see Troubleshooting AWS CloudFormation in the *AWS CloudFormation User Guide*.
> - For other issues, see Troubleshooting AWS CodeStar.

3. Choose **View Application**.

   On the new tab that opens in your web browser, the web service displays the following response output:

```
{"output": "Hello World", "timestamp": "2017-08-30T15:53:42.682839"}
```

4. In the tab's address box, add the path **/hello/** and your first name to the end of the URL (for example, **https://*API_ID*.execute-api.*REGION_ID*.amazonaws.com/Prod/hello/*YOUR_FIRST_NAME***), and then press **Enter**.

If your first name is Mary, the web service displays the following response output:

```
{"output": "Hello Mary"}
```

Step 7: Add a Unit Test to the Web Service

In this step, you use your local workstation to add a test that AWS CodeStar runs on the web service. This test replaces the manual testing you did earlier.

1. On your local workstation, go to the directory that contains the cloned source code repository.

2. In that directory, create a file named `hello_test.py`. Add the following code, and then save the file.

```
from hello import handler

def test_hello_handler():

  event = {
    'pathParameters': {
      'name': 'testname'
    }
  }

  context = {}

  expected = {
    'body': '{"output": "Hello testname"}',
    'headers': {
      'Content-Type': 'application/json'
    },
    'statusCode': 200
  }

  assert handler(event, context) == expected
```

This test checks whether the output of the Lambda function is in the expected format. If so, the test succeeds. Otherwise, the test fails.

3. In the same directory, open the `buildspec.yml` file. Replace the file's contents with the following code, and then save the file.

```
version: 0.2

phases:
   install:
      runtime-versions:
         python: 3.7

      commands:
         - pip install pytest
         # Upgrade AWS CLI to the latest version
         - pip install --upgrade awscli
```

```
   pre_build:
      commands:
         - pytest

   build:
      commands:
         # Use AWS SAM to package the application by using AWS CloudFormation
         - aws cloudformation package --template template.yml --s3-bucket
 $S3_BUCKET --output-template template-export.yml

         # Do not remove this statement. This command is required for AWS CodeStar
 projects.
         # Update the AWS Partition, AWS Region, account ID and project ID in the
 project ARN on template-configuration.json file so AWS CloudFormation can tag
 project resources.
         - sed -i.bak 's/\$PARTITION\$/'${PARTITION}'/g;s/\$AWS_REGION
\$/'${AWS_REGION}'/g;s/\$ACCOUNT_ID\$/'${ACCOUNT_ID}'/g;s/\$PROJECT_ID\
$/'${PROJECT_ID}'/g' template-configuration.json

artifacts:
   type: zip
   files:
      - template-export.yml
      - template-configuration.json
```

This build specification instructs CodeBuild to install pytest, the Python test framework, into its build environment. CodeBuild uses pytest to run the unit test. The rest of the build specification is the same as before.

4. Use Git to push these changes to the remote repository.

```
git add .

git commit -m "Added hello_test.py and updated buildspec.yml."

git push
```

Step 8: View Unit Test Results

In this step, you see whether the unit test succeeded or failed.

1. With your project still open in the AWS CodeStar console, on the navigation bar, choose **Pipeline**.
2. Make sure the pipeline has run again before you continue. This might take several minutes.

   If the unit test was successful, **Succeeded** is displayed for the **Build** stage.
3. To view the unit test result details, in the **Build** stage, choose the **CodeBuild** link.
4. In the CodeBuild console, on the **Build Project: my-sam-project** page, in **Build history**, choose the link in the **Build run** column of the table.
5. On the **my-sam-project:*BUILD_ID*** page, in **Build logs**, choose the **View entire log** link.
6. In the Amazon CloudWatch Logs console, look in the log output for a test result similar to the following. In the following test result, the test passed:

```
...
============================= test session starts ==============================
platform linux2 -- Python 2.7.12, pytest-3.2.1, py-1.4.34, pluggy-0.4.0
rootdir: /codebuild/output/src123456789/src, inifile:
collected 1 item

hello_test.py .

=========================== 1 passed in 0.01 seconds ===========================
...
```

   If the test failed, there should be details in the log output to help you troubleshoot the failure.

Step 9: Clean Up

In this step, you clean up the project to avoid ongoing charges for this project.

If you want to keep using this project, you can skip this step, but your AWS account might continue to be charged.

1. With your project still open in the AWS CodeStar console, on the navigation bar, choose **Settings**.
2. In **Project details**, Choose **Delete project**.
3. Enter **`delete`**, keep the **Delete resources** box selected, and then choose **Delete**.

> **Important**
>
> If you clear this box, the project record is deleted from AWS CodeStar, but many of the project's AWS resources are retained. Your AWS account might continue to be charged.

If there is still an Amazon S3 bucket that AWS CodeStar created for this project, follow these steps to delete it. :

1. Open the Amazon S3 console, at https://console.aws.amazon.com/s3/.
2. In the list of buckets, choose the icon next to **aws-codestar-*REGION_ID*-*ACCOUNT_ID*-my-sam-project--pipe**, where:
   - *REGION_ID* is the ID of the AWS Region for the project you just deleted.
   - *ACCOUNT_ID* is your AWS account ID.
   - **my-sam-project** is the ID of the project you just deleted.
3. Choose **Empty Bucket**. Enter the name of the bucket, and then choose **Confirm**.
4. Choose **Delete Bucket**. Enter the name of the bucket, and then choose **Confirm**.

Next Steps

Now that you have completed this tutorial, we suggest you review the following resources:

- The Getting Started with AWS CodeStar tutorial uses a project that creates and deploys a Node.js-based web application running on an Amazon EC2 instance.
- AWS CodeStar Project Templates describes other types of projects you can create.
- Working with AWS CodeStar Teams shows you how others can help you work on your projects.

Tutorial: Create a Project in AWS CodeStar with the AWS CLI

This tutorial shows you how to use the AWS CLI to create an AWS CodeStar project with sample source code and a sample toolchain template. AWS CodeStar provisions the AWS infrastructure and IAM resources specified in an AWS CloudFormation toolchain template. The project manages your toolchain resources to build and deploy your source code.

AWS CodeStar uses AWS CloudFormation to build and deploy your sample code. This sample code creates a web service that is hosted in AWS Lambda and can be accessed through Amazon API Gateway.

Prerequisites:

- Complete the steps in Setting Up AWS CodeStar.
- You must have created an Amazon S3 storage bucket. In this tutorial, you upload the sample source code and toolchain template to this location.

> **Note**
>
> Your AWS account might be charged for costs related to this tutorial, including AWS services used by AWS CodeStar. For more information, see AWS CodeStar Pricing.

Topics

- Step 1: Download and Review the Sample Source Code
- Step 2: Download the Sample Toolchain Template
- Step 3: Test Your Toolchain Template in AWS CloudFormation
- Step 4: Upload Your Source Code and Toolchain Template
- Step 5: Create a Project in AWS CodeStar

Step 1: Download and Review the Sample Source Code

For this tutorial, there is a zip file available for download. It contains sample source code for a Node.js sample application on the Lambda compute platform. When the source code is placed in your repository, its folder and files appear as shown:

```
tests/
app.js
buildspec.yml
index.js
package.json
README.md
template.yml
```

The following project elements are represented in your sample source code:

- `tests/`: Unit tests set up for this project's CodeBuild project. This folder is included in the sample code, but it is not required to create a project.
- `app.js`: Application source code for your project.
- `buildspec.yml`: The build instructions for your CodeBuild resource's build stage. This file is required for a toolchain template with an CodeBuild resource.
- `package.json`: The dependencies information for your application source code.
- `README.md`: The project readme file included in all AWS CodeStar projects. This file is included in the sample code, but it is not required to create a project.
- `template.yml`: The infrastructure template file or SAM template file included in all AWS CodeStar projects. This is different from the toolchain template.yml you upload later in this tutorial. This file is included in the sample code, but it is not required to create a project.

Step 2: Download the Sample Toolchain Template

The sample toolchain template provided for this tutorial creates a repository (CodeCommit), pipeline (CodePipeline), and build container (CodeBuild) and uses AWS CloudFormation to deploy your source code to a Lambda platform. In addition to these resources, there are also IAM roles that you can use to scope the permissions of your runtime environment, an Amazon S3 bucket that CodePipeline uses to store your deployment artifacts, and an CloudWatch Events rule that is used to trigger pipeline deployments when you push code to your repository. To align with AWS IAM best practices, scope down the policies of your toolchain roles defined in this example.

Download and unzip the sample AWS CloudFormation template in YAML format.

When you run the **create-project** command later in the tutorial. this template creates the following customized toolchain resources in AWS CloudFormation. For more information about the resources created in this tutorial, see the following topics in the *AWS CloudFormation User Guide*:

- The AWS::CodeCommit::Repository AWS CloudFormation resource creates a CodeCommit repository.
- The AWS::CodeBuild::Project AWS CloudFormation resource creates an CodeBuild build project.
- The AWS::CodeDeploy::Application AWS CloudFormation resource creates a CodeDeploy application.
- The AWS::CodePipeline::Pipeline AWS CloudFormation resource creates a CodePipeline pipeline.

- The AWS::S3::Bucket AWS CloudFormation resource creates your pipeline's artifact bucket.
- The AWS::S3::BucketPolicy AWS CloudFormation resource creates the artifact bucket policy for your pipeline's artifact bucket.
- The AWS::IAM::Role AWS CloudFormation resource creates the CodeBuild IAM worker role that gives AWS CodeStar permissions to manage your CodeBuild build project.
- The AWS::IAM::Role AWS CloudFormation resource creates the CodePipeline IAM worker role that gives AWS CodeStar permissions to create your pipeline.
- The AWS::IAM::Role AWS CloudFormation resource creates the AWS CloudFormation IAM worker role that gives AWS CodeStar permissions to create your resource stack.
- The AWS::IAM::Role AWS CloudFormation resource creates the AWS CloudFormation IAM worker role that gives AWS CodeStar permissions to create your resource stack.
- The AWS::IAM::Role AWS CloudFormation resource creates the AWS CloudFormation IAM worker role that gives AWS CodeStar permissions to create your resource stack.
- The AWS::Events::Rule AWS CloudFormation resource creates the CloudWatch Events rule that monitors your repository for push events.
- The AWS::IAM::Role AWS CloudFormation resource creates the CloudWatch Events IAM role.

Step 3: Test Your Toolchain Template in AWS CloudFormation

Before you upload your toolchain template, you can test your toolchain template in AWS CloudFormation and troubleshoot any errors.

1. Save your updated template to your local computer, and open the AWS CloudFormation console. Choose **Create Stack**. You should see your new resources in the list.
2. View your stack for any stack creation errors.
3. After your testing is complete, delete the stack.

> **Note**
>
> Make sure you delete your stack and all resources created in AWS CloudFormation. Otherwise, when you create your project, you might encounter errors for resource names already in use.

Step 4: Upload Your Source Code and Toolchain Template

To create an AWS CodeStar project, you must first package your source code into a .zip file and place it in Amazon S3. AWS CodeStar initializes your repository with these contents. You specify this location in your input file when you run the command to create your project in the AWS CLI.

You must also upload your `toolchain.yml` file and place it in Amazon S3. You specify this location in your input file when you run the command to create your project in the AWS CLI

To upload your source code and toolchain template

1. The following sample file structure shows the source files and the toolchain template ready to be zipped and uploaded. The sample code includes the `template.yml` file. Remember, this file is different from the `toolchain.yml` file.

   ```
   ls
   src toolchain.yml

   ls src/
   README.md    app.js        buildspec.yml    index.js    package.json
    template.yml    tests
   ```

2. Create the .zip for the source code files.

   ```
   cd src; zip -r "../src.zip" *; cd ../
   ```

3. Use the **cp** command and include the files as parameters.

   The following commands upload the .zip file and `toolchain.yml` to Amazon S3.

   ```
   aws s3 cp src.zip s3://MyBucket/src.zip
   aws s3 cp toolchain.yml s3://MyBucket/toolchain.yml
   ```

To configure your Amazon S3 bucket to share your source code

- Because you're storing your source code and toolchain in Amazon S3, you can use the Amazon S3 bucket policies and object ACLs to ensure that other IAM users or AWS accounts can create projects from your samples. AWS CodeStar ensures that any user who creates a custom project has access to the toolchain and source they want to use.

To let anyone use your sample, run the following commands:

```
aws s3api put-object-acl --bucket MyBucket --key toolchain.yml --acl public-read
aws s3api put-object-acl --bucket MyBucket --key src.zip --acl public-read
```

Step 5: Create a Project in AWS CodeStar

Use these steps to create your project.

> **Important**
>
> Make sure you configure the preferred AWS Region in AWS CLI. Your project is created in the AWS Region configured in the AWS CLI.

1. Run the **create-project** command and include the `--generate-cli-skeleton` parameter:

   ```
   aws codestar create-project --generate-cli-skeleton
   ```

   JSON-formatted data appears in the output. Copy the data to a file (for example, *`input.json`*) in a location on your local computer or instance where the AWS CLI is installed. Modify the copied data as follows, and save your results. This input file is configured for a project named `MyProject` with a bucket name of `myBucket`.

   - Make sure you provide the `roleArn` parameter. For custom templates, like the sample template in this tutorial, you must provide a role. This role must have permissions to create all of the resources specified in Step 2: Download the Sample Toolchain Template.
   - Make sure you provide the `ProjectId` parameter under `stackParameters`. The sample template provided for this tutorial requires this parameter.

   ```
   {
       "name": "MyProject",
       "id": "myproject",
       "description": "Sample project created with the CLI",
       "sourceCode": [
           {
   ```

```
            "source": {
                "s3": {
                    "bucketName": "MyBucket",
                    "bucketKey": "src.zip"
                }
            },
            "destination": {
                "codeCommit": {
                    "name": "myproject"

                }
            }
        }
    ],
    "toolchain": {
        "source": {
            "s3": {
                "bucketName": "MyBucket",
                "bucketKey": "toolchain.yml"
            }
        },
        "roleArn": "role_ARN",
        "stackParameters": {
            "ProjectId": "myproject"
        }
    }
}
```

2. Switch to the directory that contains the file you just saved, and run the **create-project** command again. Include the `--cli-input-json` parameter.

```
aws codestar create-project --cli-input-json file://input.json
```

3. If successful, data similar to the following appears in the output:

```
{
    "id": "project-ID",
    "arn": "arn"
}
```

- The output contains information about the new project:
  - The `id` value represents the project ID.

- The `arn` value represents the ARN of the project.

4. Use the **describe-project** command to check the status of your project creation. Include the `--id` parameter.

```
aws codestar describe-project --id <project_ID>
```

Data similar to the following appears in the output:

```
{
    "name": "MyProject",
    "id": "myproject",
    "arn": "arn:aws:codestar:us-east-1:account_ID:project/myproject",
    "description": "",
    "createdTimeStamp": 1539700079.472,
    "stackId": "arn:aws:cloudformation:us-east-1:account_ID:stack/awscodestar-
myproject/stack-ID",
    "status": {
        "state": "CreateInProgress"
    }
}
```

- The output contains information about the new project:
  - The `id` value represents the unique project ID.
  - The `state` value represents the status of the project creation, such as `CreateInProgress` or `CreateComplete`.

While your project is being created, you can add team members or configure access to your project repository from the command line or your favorite IDE.

Tutorial: Create an Alexa Skill Project in AWS CodeStar

AWS CodeStar is a cloud-based development service on AWS that provides the tools you need to quickly develop, build, and deploy applications on AWS. With AWS CodeStar, you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster. The Alexa skill project templates on AWS CodeStar enable you to create a simple Hello World Alexa skill from your AWS account with just a few clicks. The templates also create a basic deployment pipeline that gets you started with a continuous integration (CI) workflow for skill development.

The main benefits of creating Alexa skills from AWS CodeStar are that you can get started with skill development in AWS and connect your Amazon developer account to the project to deploy skills to the development stage directly from AWS. You also get a ready to use deployment (CI) pipeline with a repository with all the source code for the project. You can configure this repository with your preferred IDE to create skills with tools you are familiar with.

Prerequisites

- Create an Amazon developer account by going to https://developer.amazon.com. Signup is free. This account owns your Alexa skills.
- If you do not have an AWS account, use the following procedure to create one.

  To sign up for AWS

  1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.

     > **Note**
     >
     > If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using AWS account root user credentials**. Then choose **Create a new AWS account**.

  2. Follow the online instructions.

> **Important**
>
> After you create the Alexa skill project, make all edits in the project repository only. We recommend that you do not edit this skill directly using any other Alexa Skills Kit tools, such as the ASK CLI or ASK developer console. These tools are not integrated with the project repository. Using them causes the skill and repository code to become out of sync.

Step 1: Create the project and connect your Amazon developer account

In this tutorial, you create a skill using Node.js running on AWS Lambda. Most of the steps are the same for other languages, although the skill name differs. Refer to the README.md file in the project repository for details of the specific project template you choose.

1. Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose the AWS Region where you want to create the project and its resources. The Alexa skill runtime is available in the following AWS Regions:
   - Asia Pacific (Tokyo)
   - EU (Ireland)
   - US East (N. Virginia)
   - US West (Oregon)
3. Choose **Create project**.
4. On the **Choose a project template** page:
   a. For **Application type**, choose **Alexa Skill**.
   b. For **Programming language**, choose **Node.js**.
5. Choose the box that contains your selections.
6. For **Project name**, enter a name for the project (for example, **`My Alexa Skill`**). If you use a different name, be sure to use it throughout this tutorial. AWS CodeStar chooses a related identifier for this project for the **Project ID** (for example, **my-alexa-skill**). If you see a different project ID, be sure to use it throughout this tutorial.
7. Choose **AWS CodeCommit** for the repository in this tutorial and do not change the **Repository name** value.
8. Choose **Connect Amazon developer account** to link to your Amazon developer account for hosting the skill. If you don't have an Amazon developer account, create an account and complete the registration first from Amazon Developers.
9. Sign in with your Amazon developer credentials. Choose **Allow**, and then choose **Confirm** to complete the connection.
10. If you have multiple vendor IDs associated with your Amazon developer account, choose the one that you want to use for this project. Make sure you use an account with the Administrator or Developer role assigned.
11. Choose **Next**.
12. (Optional) If this is your first time using AWS CodeStar in this AWS Region, enter the display name and email address you want AWS CodeStar to use for your IAM user. Choose **Next**.
13. Wait while AWS CodeStar creates the project. This might take several minutes. Do not continue until you see the **Project provisioned** banner.

Step 2: Test your skill in the Alexa Simulator

In the first step, AWS CodeStar created a skill for you and deployed it to the Alexa skill development stage. Next, you test the skill in the Alexa Simulator.

1. In your project in the AWS CodeStar console, choose **View application**. A new tab opens in the Alexa Simulator.
2. Sign in with your Amazon developer credentials for the account you connected to your project in Step 1.
3. Under **Test,** choose **Development** to enable testing.
4. Enter `ask hello node hello`. The default invocation name for your skill is `hello node`.
5. Your skill should respond `Hello World!`.

When the skill is enabled in the Alexa Simulator, you can also invoke it on an Alexa-enabled device that is registered to your Amazon developer account. To test your skill on a device, say *Alexa, ask hello node to say hello*.

For more information about the Alexa Simulator, see Test Your Skill in the Developer Console.

Step 3: Explore your project resources

As part of creating the project, AWS CodeStar also created AWS resources on your behalf. These resources include a project repository using CodeCommit, a deployment pipeline using CodePipeline and an AWS Lambda function. You can access these resources from the navigation bar. For example, choosing **Repository** shows details about the CodeCommit repository. You can view the pipeline deployment status in the **Pipeline** page. You can view a complete list of AWS resources created as part of your project by choosing **Overview** in the navigation bar. This list includes links to each resource.

Step 4: Make a change in your skill's response

In this step, you make a minor change to your skill's response to understand the iteration cycle.

1. In the navigation bar, choose **Repository**. Choose the link under **Repository name** and your project's repository opens in a new tab or window. This repository contains the build specification (buildspec.yml), AWS CloudFormation application stack (template.yml), readme file, and your skill's source code in the skill package format (project structure).

2. Navigate to the file lambda > custom > index.js (in case of Node.js.). This file contains your request handling code, which uses the ASK SDK.
3. Choose **Edit**.
4. Replace the string `Hello World!` in line 24 with the string `Hello. How are you?`.
5. Scroll down to the end of the file. Enter author name and email address and an optional commit message.
6. Choose **Commit changes** to commit the changes to the repository.
7. Return to the project in AWS CodeStar and check the **Pipeline** page. You should now see the pipeline deploying.
8. When the pipeline finishes deployment, test your skill again in the Alexa Simulator. Your skill should now respond with `Hello. How are you?`

Step 5: Set up your local workstation to connect to your project repository

Earlier you made a small change to the source code directly from the CodeCommit console. In this step, you configure the project repository with your local workstation so that you can edit and manage code from the command line or your favorite IDE. The following steps explain how to set up command line tools.

1. Navigate to the project dashboard in AWS CodeStar, if necessary.
2. In the navigation bar, choose **IDE**.
3. In **Access your project code**, **View instructions** underneath **Command line interface**.
4. Follow the instructions to complete the following tasks:
   a. Install Git on your local workstation from a website such as Git Downloads.
   b. Install the AWS CLI. For information, see Installing the AWS Command Line Interface.
   c. Configure the AWS CLI with your IAM user access key and secret key. For information, see Configuring the AWS CLI.
   d. Clone the project's CodeCommit repository onto your local workstation. For more information, see Connect to a CodeCommit Repository.

Next Steps

This tutorial showed you how to get started with a basic skill. To continue your skill development journey, see the following resources.

- Understand the fundamentals of a skill by watching How Alexa Skills Work and other videos on the Alexa Developers YouTube channel.
- Understand the various components of your skill by reviewing the documentation for the skill package format, the skill manifest schemas, and the interaction model schemas.
- Turn your idea into a skill by reviewing the documentation for Alexa Skills Kit and the ASK SDKs.

Tutorial: Create a Project with a GitHub Source Repository

With AWS CodeStar, you can set up your repository to create, review, and merge pull requests with your project team.

In this tutorial, you create a project with sample web application source code in a GitHub repository, a pipeline that deploys your changes, and EC2 instances where your application is hosted in the cloud. After your project is created, this tutorial shows you how to create and merge a GitHub pull request that makes a change to your web application's home page.

Topics

- Step 1: Create the project and create your GitHub repository
- Step 2: View your source code
- Step 3: Create a GitHub Pull Request

Step 1: Create the project and create your GitHub repository

In this step, use the console to create your project and create a connection to your new GitHub repository. To access your GitHub repository, you create a connection resource that AWS CodeStar uses to manage authorization with GitHub. When the project is created, its additional resources are provisioned for you.

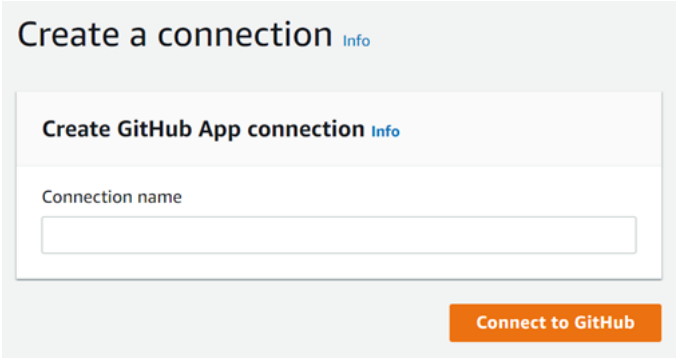
1. Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose the AWS Region where you want to create the project and its resources.

3. On the **AWS CodeStar** page, choose **Create project**.
4. On the **Choose a project template** page, select the **Web application**, **Node.js**, and **Amazon EC2** check boxes. Then choose from the templates available for that set of options.

   For more information, see AWS CodeStar Project Templates.
5. Choose **Next**.
6. For **Project name**, enter a name for the project (for example, **`MyTeamProject`**). If you use a different name, be sure to use it throughout this tutorial.
7. Under **Project repository**, choose **GitHub**.
8. If you chose **GitHub**, you will need to choose or create a connection resource. If you have an existing connection, choose it in the search field. Otherwise, you will create a new connection here. Choose **Connect to GitHub**.

   The **Create a connection** page displays.

   > **Note**
   >
   > To create a connection, you must have a GitHub account. If you are creating a connection for an organization, you must be the organization owner.



   a. Under **Create GitHub App connection**, in **Connection name**, enter a name for your connection. Choose **Connect to GitHub**.

      The **Connect to GitHub** page displays and shows the **GitHub Apps** field.

   b. Under **GitHub Apps**, choose an app installation or choose **Install a new app** to create one.

> **Note**
>
> You install one app for all of your connections to a particular provider. If you have already installed the AWS Connector for GitHub app, choose it and skip this step.

c. On the **Install AWS Connector for GitHub** page, choose the account where you want to install the app.

> **Note**
>
> If you previously installed the app, you can choose **Configure** to proceed to a modification page for your app installation, or you can use the back button to return to the console.

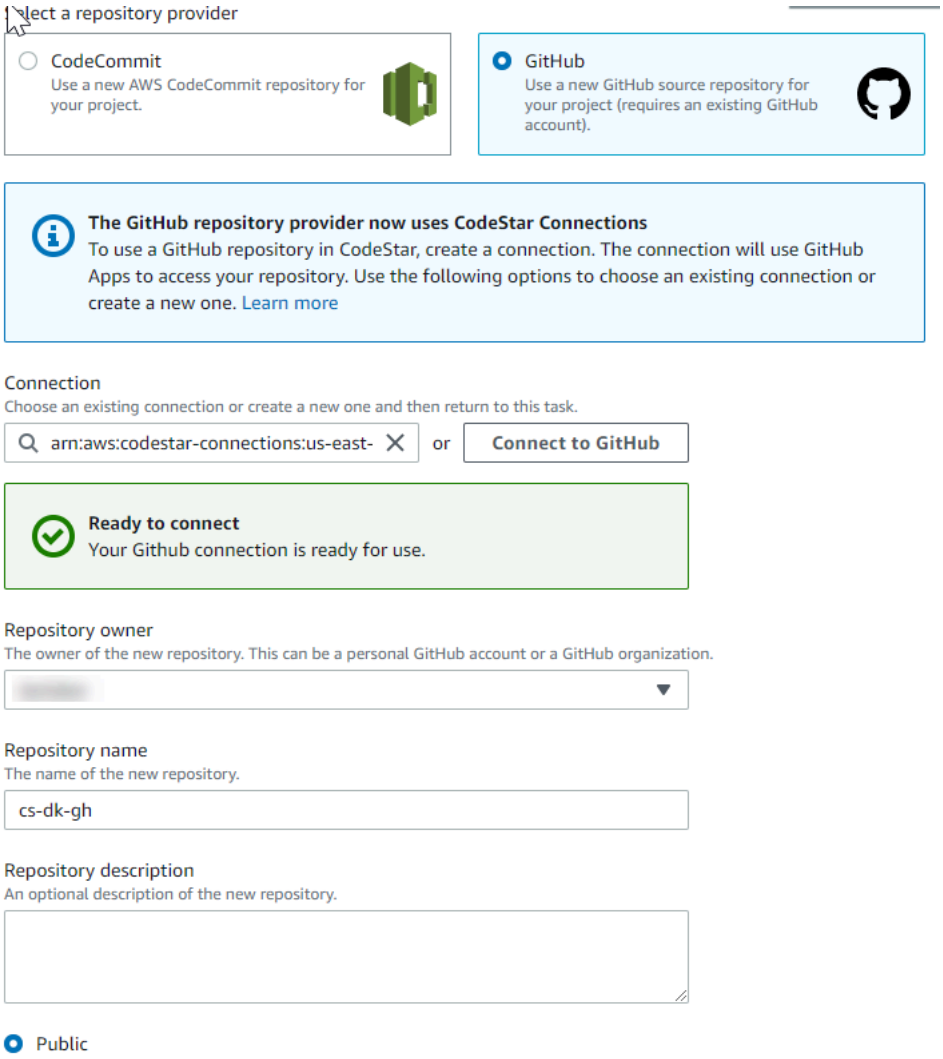
d. If the **Confirm password to continue** page is displayed, enter your GitHub password, and then choose **Sign in**.

e. On the **Install AWS Connector for GitHub** page, leave the defaults, and choose **Install**.

f. On the **Connect to GitHub** page, the installation ID for your new installation appears in **GitHub Apps**.

After the connection is successfully created, in the CodeStar create project page, the message **Ready to connect** displays.

> **Note**
>
> You can view your connection under Settings in the Developer Tools console. For more information, see Getting started with connections.



g. For **Repository owner**, choose the GitHub organization or your personal GitHub account.

h. For **Repository name**, accept the default GitHub repository name, or enter a different one.

i. Choose **Public** or **Private**.

> **Note**
>
> If you want to use AWS Cloud9 as your development environment, you must choose a public repository.

j. (Optional) For **Repository description**, enter a description for the GitHub repository.

9. Configure your Amazon EC2 instances in **Amazon EC2 Configuration** if your project is deployed to Amazon EC2 instances and you want to make changes. For example, you can choose from available instance types for your project.

   In **Key pair**, choose the Amazon EC2 key pair you created in Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects. Select **I acknowledge that I have access to the private key file**.
10. Choose **Next**.
11. Review the resources and configuration details.
12. Choose **Next** or **Create project**. (The displayed choice depends on your project template.)

    Allow a few minutes while your project is created.
13. After your project is created, choose **View application** to view your web application.

Step 2: View your source code

In this step, you view your source code and the tools you can use for your source repository.

1. In the navigation bar for your project, choose **Repository**.

   To view a list of commits in GitHub, choose **View commits**. This opens your commit history in GitHub.

   To view issues, choose the **Issues** tab for your project. To create a new issue in GitHub, choose **Create GitHub issue**. This opens your repository issue form in GitHub.
2. Under the **Repository** tab, choose the link under **Repository name**, and your project's repository opens in a new tab or window. This repository contains the source code for your project.

Step 3: Create a GitHub Pull Request

In this step, you make a minor change to your source code and create a pull request.

1. In GitHub, create a new feature branch in your repository. Choose the main branch drop-down field and enter a new branch in the field named `feature-branch`. Choose **Create new branch**. The branch is created and checked out for you.

2. In GitHub, make a change in the `feature-branch` branch. Open the public folder and open the `index.html` file.
3. In the AWS CodeStar console, under **Pull requests**, to create a pull request in GitHub, choose **Create pull request**. This opens your repository pull request form in GitHub. In GitHub, choose the pencil icon to edit the file.

   After `Congratulations!`, add the string `Well done, <name>!` and replace `<name>` with your name. Choose **Commit changes**. The change is committed to your feature branch.
4. In the AWS CodeStar console, choose your project. Choose the **Repository** tab. Under Pull requests, choose **Create pull request**.

   The form opens in GitHub. Leave the main branch in the base branch. For **Compare to**, choose your feature branch. View the diff.
5. In GitHub, choose **Create pull request**. A pull request named *Update index.html* is created.
6. In the AWS CodeStar console, view the new pull request. Choose **Merge changes** to commit the changes to the repository and merge the pull request with the main branch of your repository.
7. Return to the project in AWS CodeStar and check the **Pipeline** page. You should now see the pipeline deploying.
8. After your project is created, choose **View application** to view your web application.

AWS CodeStar Project Templates

AWS CodeStar project templates allow you to start with a sample application and deploy it using AWS resources created to support your development project. When you choose an AWS CodeStar project template, the application type, programming language, and compute platform are provisioned for you. After you create projects with web applications, web services, Alexa skills, and static web pages, you can replace the sample application with your own.

After AWS CodeStar creates your project, you can modify the AWS resources that support delivery of your application. AWS CodeStar works with AWS CloudFormation to allow you to use code to create support services and servers/serverless platforms in the cloud. AWS CloudFormation allows you to model your entire infrastructure in a text file.

Topics

- AWS CodeStar Project Files and Resources
- Get Started: Choose a Project Template
- How to Make Changes to Your AWS CodeStar Project

AWS CodeStar Project Files and Resources

An AWS CodeStar project is a combination of source code and the resources created to deploy the code. The collection of resources that help you build, release, and deploy your code are called *toolchain resources*. At project creation, an AWS CloudFormation template provisions your toolchain resources in a continuous integration/continuous deployment (CI/CD) pipeline.

You can use AWS CodeStar to create projects in two ways, depending on your experience level with AWS resource creation:

- When you use the console to create a project, AWS CodeStar creates your toolchain resources, including your repository, and populates your repository with sample application code and project files. Use the console to quickly set up sample projects based on a set of preconfigured project options.
- When you use the CLI to create a project, you provide the AWS CloudFormation template that creates your toolchain resources and the application source code. Use the CLI to allow AWS CodeStar to create your project from your template and then populate your repository with your sample code.

An AWS CodeStar project provides a single point of management. You can use the **Create project** wizard in the console to set up a sample project. You can then use it as a collaboration platform for managing permissions and resources for your team. For more information, see What Is AWS CodeStar?. When you use the console to create a project, your source code is provided as sample code, and your CI/CD toolchain resources are created for you

When you create a project in the console, AWS CodeStar provisions the following resources:

- A code repository in GitHub or CodeCommit.
- In the project repository, a `README.md` file that provides details of files and directories.
- In the project repository, a `template.yml` file that stores the definition for your application's runtime stack. You use this file to add or modify project resources that are not toolchain resources, such as AWS resources used for notifications, database support, monitoring, and tracing.
- AWS services and resources created in connection with your pipeline, such as the Amazon S3 artifact bucket, Amazon CloudWatch Events, and related service roles.
- A working sample application with full source code and a public HTTP endpoint.
- An AWS compute resource, based on the AWS CodeStar project template type:
  - A Lambda function.
  - An Amazon EC2 instance.
  - An AWS Elastic Beanstalk environment.
- Starting December 6, 2018 PDT:
  - A permissions boundary, which is a specialized IAM policy for controlling access to project resources. The permissions boundary is attached by default to roles in the sample project. For more information, see IAM Permissions Boundary for Worker Roles.
  - An AWS CloudFormation IAM role for creating project resources using AWS CloudFormation that includes permissions for all AWS CloudFormation supported resources, including IAM roles.
  - A toolchain IAM role.
  - Execution roles for Lambda defined in the application stack, which you can modify.
- Before December 6, 2018 PDT:
  - An AWS CloudFormation IAM role for creating project resources with support for a limited set of AWS CloudFormation resources.

- An IAM role for creating a CodePipeline resource.
- An IAM role for creating an CodeBuild resource.
- An IAM role for creating a CodeDeploy resource, if applicable to your project type.
- An IAM role for creating the Amazon EC2 web app, if applicable to your project type.
- An IAM role for creating a CloudWatch Events resource.
- An execution role for Lambda that is dynamically modified to include a partial set of resources.

The project includes detail pages that show status and contain links to team management, links to setup instructions for IDEs or your repository, and a commit history of source code changes in the repository. You can also select tools for connecting to external issue tracking tools, such as Jira.

Get Started: Choose a Project Template

When you choose an AWS CodeStar project in the console, you are choosing from a set of preconfigured options with sample code and resources to get you started quickly. These options are called *project templates*. Each AWS CodeStar project template consists of a programming language, application type, and compute platform. The combination you select determines the project template.

Choose a Template Compute Platform

Each template configures one of the following compute platform types:

- When you choose an AWS Elastic Beanstalk project, you deploy to an AWS Elastic Beanstalk environment on Amazon Elastic Compute Cloud instances in the cloud.
- When you choose an Amazon EC2 project, AWS CodeStar creates Linux EC2 instances to host your application in the cloud. Your project team members can access the instances, and your team uses the key pair you provide to SSH into your Amazon EC2 instances. AWS CodeStar also has a managed SSH that uses team member permissions to manage key pair connections.
- When you choose AWS Lambda, AWS CodeStar creates a serverless environment accessed through Amazon API Gateway, with no instances or servers to maintain.

Choose a Template Application Type

Each template configures one of the following application types:

- **Web service**

  A web service is used for tasks that run in the background, such as calling APIs. After AWS CodeStar creates your sample web service project, you can choose the endpoint URL to see Hello World output, but the primary use of this application type is not as a user interface (UI). The AWS CodeStar project templates in this category support development in Ruby, Java, ASP.NET, PHP, Node.js, and more.

- **Web application**

  A web application features a UI. After AWS CodeStar creates your sample web application project, you can choose the endpoint URL to see an interactive web application. The AWS CodeStar project templates in this category support development in Ruby, Java, ASP.NET, PHP, Node.js, and more.

- **Static web page**

  Choose this template if you want a project for an HTML website. The AWS CodeStar project templates in this category support development in HTML5.

- **Alexa skill**

  Choose this template if you want a project for an Alexa skill with an AWS Lambda function. When you create the skill project, AWS CodeStar returns an Amazon Resource Name (ARN) that you can use as a service endpoint. For more information, see Host a Custom Skill as an AWS Lambda Function.

  > **Note**
  >
  > Lambda functions for Alexa skills are supported in the US East (N. Virginia), US West (Oregon), EU (Ireland), and Asia Pacific (Tokyo) Regions only.

- **Config rule**

  Choose this template if you want a project for an AWS Config rule that lets you automate rules across AWS resources in your account. The function returns an ARN that you can use as a service endpoint for your rule.

Choose a Template Programming Language

When you choose a project template, you select a programming language, such as Ruby, Java, ASP.NET, PHP, Node.js, and more.

How to Make Changes to Your AWS CodeStar Project

You can update your project by modifying:

- Sample code and programming language resources for your application.
- Resources that make up the infrastructure where your application is stored and deployed (operating systems, support applications and services, deployment parameters, and the cloud compute platform). You can modify application resources in the `template.yml` file. This is the AWS CloudFormation file that models your application's runtime environment.

> **Note**
>
> If you are working with an Alexa Skills AWS CodeStar project, you cannot make changes to the skill outside of the AWS CodeStar source repository (CodeCommit or GitHub). If you edit the skill in the Alexa developer portal, the change might not be visible in the source repository and the two versions will be out of sync.

Change Application Source Code and Push Changes

To modify sample source code, scripts, and other application source files, edit files in your source repository by:

- Using the Edit mode in CodeCommit or GitHub.
- Opening the project in an IDE, such as AWS Cloud9.
- Cloning the repository locally and then committing and pushing your changes. For information, see Step 4: Commit a Change.

Change Application Resources with the Template.yml File

Instead of manually modifying an infrastructure resource, use AWS CloudFormation to model and deploy your application's runtime resources.

You can modify or add an application resource, such as a Lambda function, in your runtime stack by editing the `template.yml` file in your project repository. You can add any resource that is available as an AWS CloudFormation resource.

To change the code or settings of an AWS Lambda function, see Add a Resource to a Project.

Modify the `template.yml` file in your project's repository to add the type of AWS CloudFormation resources that are application resources. When you add an application resource to the `Resources` section of the `template.yml` file, AWS CloudFormation and AWS CodeStar create the resource for you. For a list of AWS CloudFormation resources and their required properties, see AWS Resource Types Reference. For more information, see this example in Step 1: Edit the CloudFormation Worker Role in IAM.

AWS CodeStar allows you to implement best practices by configuring and modeling your application's runtime environment.

How to Manage Permissions to Change Application Resources

When you use AWS CloudFormation to add runtime application resources, such as a Lambda function, the AWS CloudFormation worker role can use the permissions it already has. For some runtime application resources, you must manually adjust the AWS CloudFormation worker role's permissions before you edit the `template.yml` file.

For an example of changing the AWS CloudFormation worker role's permissions, see Step 5: Add Resource Permissions with an Inline Policy.

AWS CodeStar Best Practices

AWS CodeStar is integrated with a number of products and services. The following sections describe best practices for AWS CodeStar and these related products and services.

Topics

- Security Best Practices for AWS CodeStar Resources
- Best Practices for Setting Versions for Dependencies
- Monitoring and Logging Best Practices for AWS CodeStar Resources

Security Best Practices for AWS CodeStar Resources

You should regularly apply patches and review security best practices for the dependencies used by your application. Use these security best practices to update your sample code and maintain your project in a production environment:

- Track ongoing security announcements and updates for your framework.
- Before you deploy your project, follow the best practices developed for your framework.
- Review dependencies for your framework on a regular basis and update as needed.
- Each AWS CodeStar template contains configuration instructions for your programming language. See the `README.md` file in your project's source repository.
- As a best practice for isolating project resources, manage least-privilege access to AWS resources using a *multi-account strategy* as introduced in Security in AWS CodeStar.

Best Practices for Setting Versions for Dependencies

The sample source code in your AWS CodeStar project uses dependencies that are listed in the `package.json` file in your source repository. As a best practice, always set your dependencies to point to a specific version. This is known as pinning the version. We do not recommend that you set the version to `latest` because that can introduce changes that might break your application without notice.

Monitoring and Logging Best Practices for AWS CodeStar Resources

You can use logging features in AWS to determine the actions users have taken in your account and the resources that were used. The log files show:

- The time and date of actions.
- The source IP address for an action.
- Which actions failed due to inadequate permissions.

AWS CloudTrail can be used to log AWS API calls and related events made by or on behalf of an AWS account. For more information, see Logging AWS CodeStar API Calls with AWS CloudTrail.

Working with Projects in AWS CodeStar

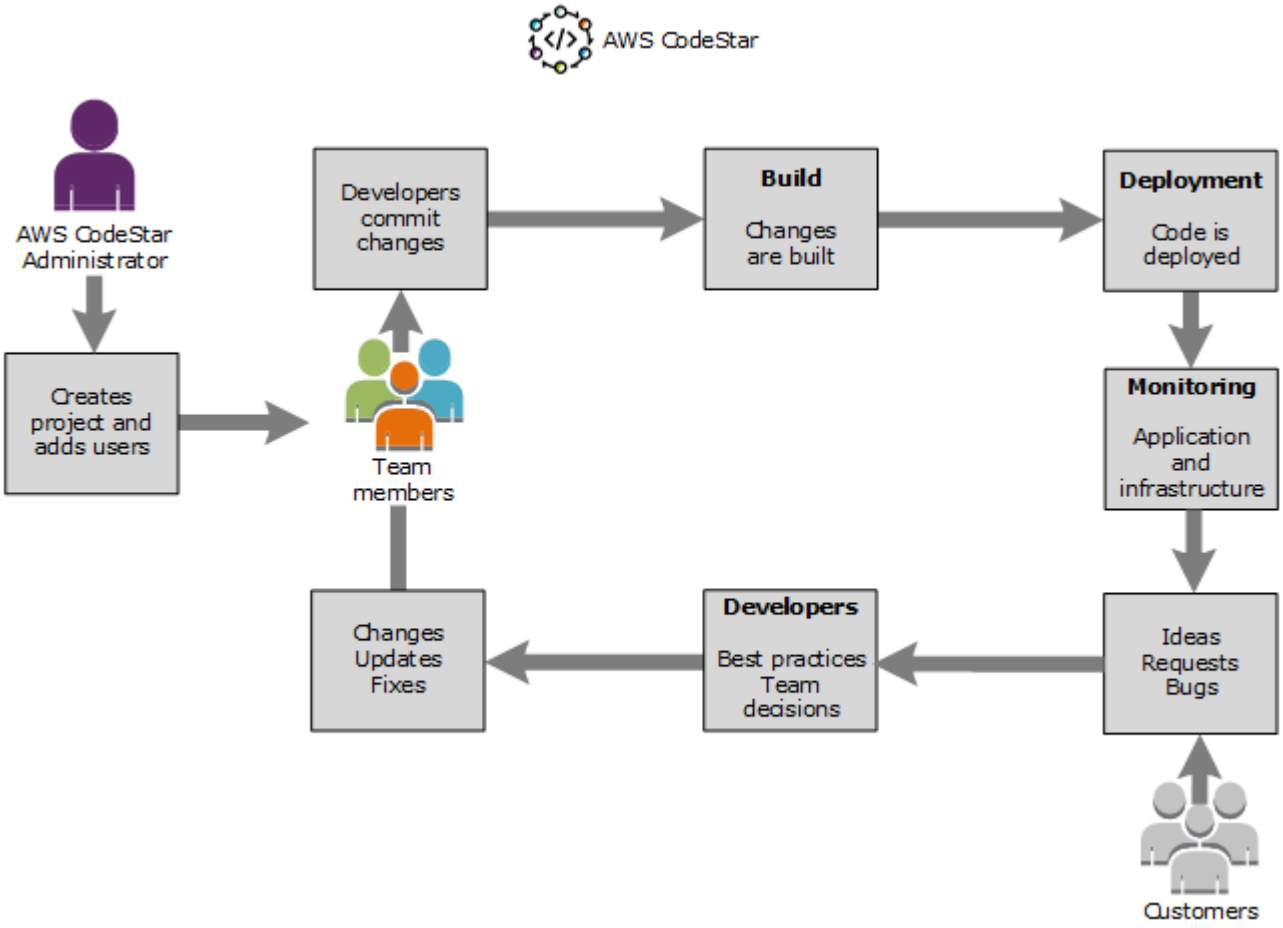
When you use an AWS CodeStar project template, you can quickly create a project that is already configured with the resources you need, including:

- Source repository
- Build environment
- Deployment and hosting resources
- Programming language

The template even includes sample source code so you can start working with your project right away.

After you have a project, you can add or remove resources, customize your project dashboard, and monitor progress.

The following diagram shows a basic workflow in an AWS CodeStar project.



The basic workflow in the diagram shows a developer with the `AWSCodeStarFullAccess` policy applied that creates a project and adds team members to it. Together they write, build, test, and deploy code. The project dashboard provides tools that can be used in real time to view application activity and monitor builds, the flow of code through the deployment pipeline, and more. The team uses the team wiki tile to share information, best practices, and links. They integrate their issue-tracking software to help them track progress and tasks. As customers provide requests and feedback, the team adds this information to the project and integrates it into their project planning and development. As the project grows, the team adds more team members to support their code base.

Create a Project in AWS CodeStar

You use the AWS CodeStar console to create a project. If you use a project template, it sets up the required resources for you. The template also includes sample code that you can use to start coding.

To create a project, sign in to the AWS Management Console with an IAM user that has the `AWSCodeStarFullAccess` policy or equivalent permissions. For more information, see Setting Up AWS CodeStar.

> **Note**
>
> You must complete the steps in Setting Up AWS CodeStar before you can complete the procedures in this topic.

Topics

- Create a Project in AWS CodeStar (Console)
- Create a Project in AWS CodeStar (AWS CLI)

Create a Project in AWS CodeStar (Console)

Use the AWS CodeStar console to create a project.

To create a project in AWS CodeStar

1. Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.

   Make sure that you are signed in to the AWS Region where you want to create the project and its resources. For example, to create a project in US East (Ohio), make sure you have selected that AWS Region. For information about AWS Regions where AWS CodeStar is available, see Regions and Endpoints in the *AWS General Reference* .
2. On the **AWS CodeStar** page, choose **Create project**.
3. On the **Choose a project template** page, choose the project type from the list of AWS CodeStar project templates. You can use the filter bar to narrow your choices. For example, for a web application project written in Node.js to be deployed to Amazon EC2 instances, select the **Web application**, **Node.js**, and **Amazon EC2** check boxes. Then choose from the templates available for that set of options.

   For more information, see AWS CodeStar Project Templates.
4. Choose **Next**.

5. In the **Project name** text input field, enter a name for the project, such as *My First Project*. In **Project ID**, the ID for the project is derived from this project name, but is limited to 15 characters.

   For example, the default ID for a project named *My First Project* is *my-first-projec*. This project ID is the basis for the names of all resources associated with the project. AWS CodeStar uses this project ID as part of the URL for your code repository and for the names of related security access roles and policies in IAM. After the project is created, the project ID cannot be changed. To edit the project ID before you create the project, in **Project ID**, enter the ID you want to use.

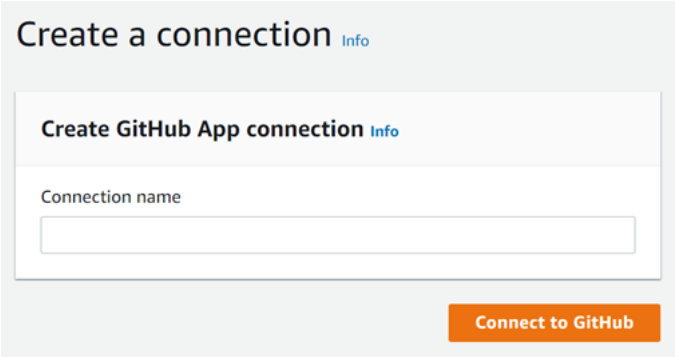
   For information about the limits on project names and project IDs, see Limits in AWS CodeStar.

   > **Note**
   >
   > Project IDs must be unique for your AWS account in an AWS Region.

6. Choose the repository provider, **AWS CodeCommit** or **GitHub**.
7. If you chose **AWS CodeCommit**, for **Repository name**, accept the default AWS CodeCommit repository name, or enter a different one. Then skip ahead to step 9.
8. If you chose **GitHub**, you need to choose or create a connection resource. If you have an existing connection, choose it in the search field. Otherwise, create a new connection now. Choose **Connect to GitHub**.

   The **Create a connection** page displays.

   > **Note**
   >
   > To create a connection, you must have a GitHub account. If you are creating a connection for an organization, you must be the organization owner.



a. Under **Create GitHub App connection**, in the **Connection name** input text field, enter a name for your connection. Choose **Connect to GitHub**.

   The **Connect to GitHub** page displays and shows the **GitHub Apps** field.

b. Under **GitHub Apps**, choose an app installation or choose **Install a new app** to create one.

> **Note**
>
> You install one app for all of your connections to a particular provider. If you have already installed the AWS Connector for GitHub app, choose it and skip this step.

c. On the **Install AWS Connector for GitHub** page, choose the account where you want to install the app.

> **Note**
>
> If you previously installed the app, you can choose **Configure** to proceed to a modification page for your app installation, or you can use the back button to return to the console.

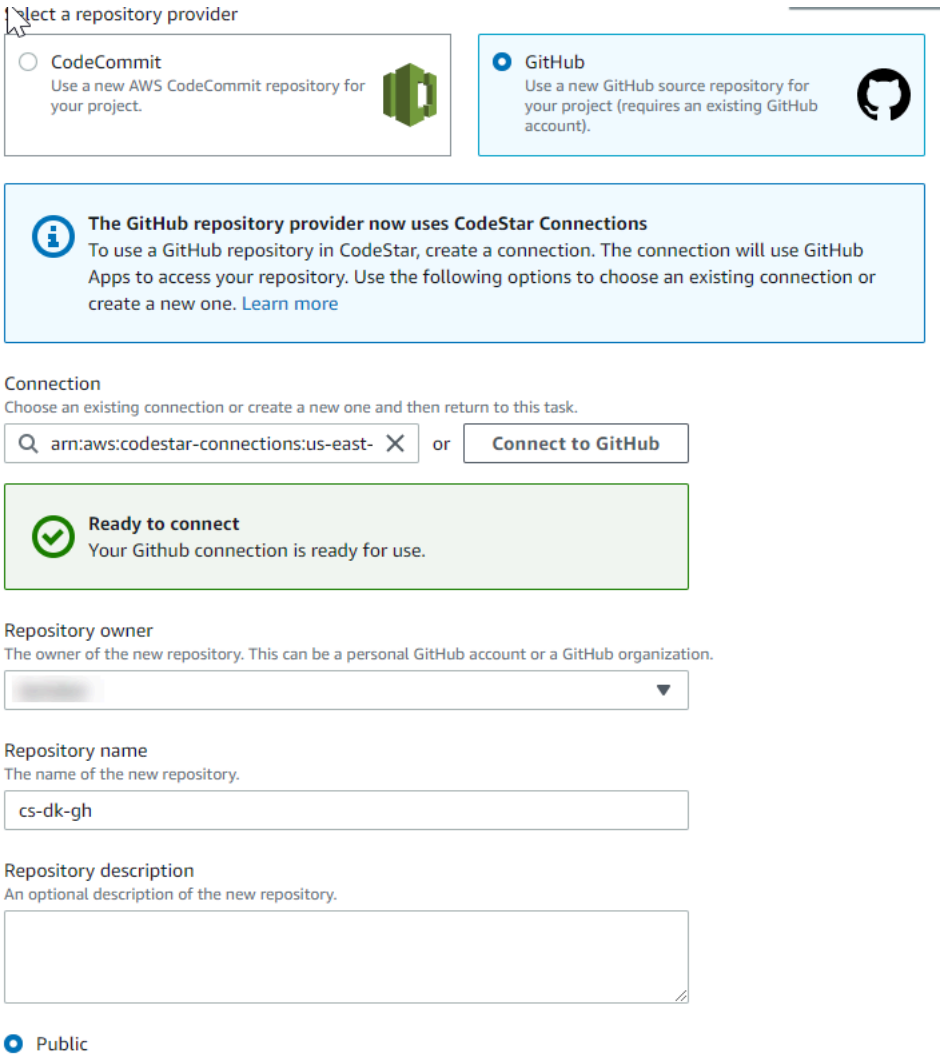
d. If the **Confirm password to continue** page is displayed, enter your GitHub password, and then choose **Sign in**.

e. On the **Install AWS Connector for GitHub** page, keep the defaults, and choose **Install**.

f. On the **Connect to GitHub** page, the installation ID for your new installation appears in the **GitHub Apps** text input field.

   After the connection is created, in the CodeStar create project page, the message **Ready to connect** displays.

> **Note**
>
> You can view your connection under **Settings** in the **Developer Tools** console. For more information, see Getting started with connections.



g. For **Repository owner**, choose the GitHub organization or your personal GitHub account.

h. For **Repository name**, accept the default GitHub repository name, or enter a different one.

i. Choose **Public** or **Private**.

> **Note**
>
> To use AWS Cloud9 as your development environment, you must choose **Public**.

j. (Optional) For **Repository description**, enter a description for the GitHub repository.

> **Note**
>
> If you choose an Alexa Skill project template, you need to connect an Amazon developer account. For more information about working with Alexa Skill projects, see Tutorial: Create an Alexa Skill Project in AWS CodeStar.

9. If your project is deployed to Amazon EC2 instances and you want to make changes, configure your Amazon EC2 instances in **Amazon EC2 Configuration**. For example, you can choose from available instance types for your project.

> **Note**
>
> Different Amazon EC2 instance types provide different levels of computing power and might have different associated costs. For more information, see Amazon EC2 Instance Types and Amazon EC2 Pricing.
> If you have more than one virtual private cloud (VPC) or multiple subnets created in Amazon Virtual Private Cloud, you can also choose the VPC and subnet to use. However, if you choose an Amazon EC2 instance type that is not supported on dedicated instances, you cannot choose a VPC whose instance tenancy is set to **Dedicated**.
> For more information, see What Is Amazon VPC? and Dedicated Instance Basics.

In **Key pair**, choose the Amazon EC2 key pair you created in Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects. Select **I acknowledge that I have access to the private key file**.

10. Choose **Next**.
11. Review the resources and configuration details.
12. Choose **Next** or **Create project**. (The displayed choice depends on your project template.)

It might take a few minutes to create the project, including the repository.

13. After your project has a repository, you can use the **Repository** page to configure access to it. Use the links in **Next steps** to configure an IDE, set up issue tracking, or add team members to your project.

While your project is being created, you can add team members or configure access to your project repository from the command line or your favorite IDE.

Create a Project in AWS CodeStar (AWS CLI)

An AWS CodeStar project is a combination of source code and the resources created to deploy the code. The collection of resources that help you build, release, and deploy your code are called toolchain resources. At project creation, an AWS CloudFormation template provisions your toolchain resources in a continuous integration/continuous deployment (CI/CD) pipeline.

When you use the console to create a project, the toolchain template is created for you. When you use the AWS CLI to create a project, you create the toolchain template that creates your toolchain resources.

A complete toolchain requires the following recommended resources:

1. A CodeCommit or GitHub repository that contains your source code.
2. A CodePipeline pipeline that is configured to listen to changes to your repository.
   a. When you use CodeBuild to run unit or integration tests, we recommend that you add a build stage to your pipeline to create build artifacts.
   b. We recommend that you add a deployment stage to your pipeline that uses CodeDeploy or AWS CloudFormation to deploy your build artifact and source code to your runtime infrastructure.

> **Note**
>
> Because CodePipeline requires at least two stages in a pipeline, and the first stage must be the source stage, add a build or deploy stage as the second stage.

AWS CodeStar toolchains are defined as a CloudFormation template.

For a tutorial that walks through this task and sets up sample resources, see Tutorial: Create a Project in AWS CodeStar with the AWS CLI.

Prerequisites:

When you create a project, you provide the following parameters in an input file. If the following are not provided, AWS CodeStar creates an empty project.

- **Source code**. If this parameter is included in your request, then you must also include a toolchain template.
  - Your source code must include the application code required to run your project.
  - Your source code must include any required configuration files, such as a buildspec.yml for a CodeBuild project or an appspec.yml for a CodeDeploy deployment.
  - You can include optional items in your source code such as a README or a template.yml for non-toolchain AWS resources.
- **Toolchain template**. Your toolchain template provisions the AWS resources and IAM roles to be managed for your project.
- **Source locations**. If you specify source code and a toolchain template for your project, you must provide a location. Upload your source files and your toolchain template to the Amazon S3 bucket. AWS CodeStar retrieves the files and uses them to create the project.

> **Important**
>
> Make sure you configure the preferred AWS Region in the AWS CLI. Your project is created in the AWS Region configured in the AWS CLI.

1. Run the **create-project** command and include the `--generate-cli-skeleton` parameter:

   ```
   aws codestar create-project --generate-cli-skeleton
   ```

   JSON-formatted data appears in the output. Copy the data to a file (for example, *`input.json`*) in a location on your local computer or instance where the AWS CLI is installed. Modify the copied data as follows, and save your results.

   ```
   {
       "name": "project-name",
   ```

```
    "id": "project-id",
    "description": "description",
    "sourceCode": [
        {
            "source": {
                "s3": {
                    "bucketName": "s3-bucket-name",
                    "bucketKey": "s3-bucket-object-key"
                }
            },
            "destination": {
                "codeCommit": {
                    "name": "codecommit-repository-name"
                },
                "gitHub": {
                    "name": "github-repository-name",
                    "description": "github-repository-description",
                    "type": "github-repository-type",
                    "owner": "github-repository-owner",
                    "privateRepository": true,
                    "issuesEnabled": true,
                    "token": "github-personal-access-token"
                }
            }
        }
    ],
    "toolchain": {
        "source": {
            "s3": {
                "bucketName": "s3-bucket-name",
                "bucketKey": "s3-bucket-object-key"
            }
        },
        "roleArn": "service-role-arn",
        "stackParameters": {
            "KeyName": "key-name"
        }
    },
    "tags": {
        "KeyName": "key-name"
    }
}
```

Replace the following:

- *project-name*: Required. The friendly name for this AWS CodeStar project.
- *project-id*: Required. The project ID for this AWS CodeStar project.

  > **Note**
  >
  > You must have a unique project ID when you create a project. You receive an error if you submit an input file with a project ID that already exists.

- *description*: Optional. The description for this AWS CodeStar project.
- *sourceCode*: Optional. The configuration information for the source code provided for the project. Currently, only a single `sourceCode` object is supported. Each `sourceCode` object contains information about the location where source code is retrieved by AWS CodeStar and the destination where the source code is populated.
  - *source*: Required. This defines the location where you uploaded your source code. The only supported source is Amazon S3. AWS CodeStar retrieves the source code and includes it in the repository after your project is created.
    - *S3*: Optional. The Amazon S3 location of your source code.
      - *bucket-name*: The bucket that contains your source code.
      - *bucket-key*: The bucket prefix and object key that points to the .zip file that contains your source code (for example, `src.zip`).
  - *destination*: Optional. The destination locations where your source code is to be populated when the project is created. The supported destinations for your source code are CodeCommit and GitHub.

    You can provide only one of these two options:

    - *codeCommit*: The only required attribute is the name of the CodeCommit repository that should contain your source code. This repository should be in your toolchain template.

> **Note**
>
> For CodeCommit, you must provide the name of the repository that you defined in your toolchain stack. AWS CodeStar initializes this repository with the source code you provided in Amazon S3.

- *gitHub*: This object represents information required to create the GitHub repository and seed it with source code. If you choose a GitHub repository, the following values are required.

> **Note**
>
> For GitHub, you cannot specify an existing GitHub repository. AWS CodeStar creates one for you and populates this repository with the source code you uploaded to Amazon S3. AWS CodeStar uses the following information to create your repository in GitHub.

  - *name*: Required. The name of your GitHub repository.
  - *description*: Required. The description of your GitHub repository.
  - *type*: Required. The type of GitHub repository. Valid values are User or Organization.
  - *owner*: Required. The GitHub user name for the owner of your repository. If the repository should be owned by a GitHub organization, provide the organization name.
  - *privateRepository*: Required. Whether you want this repository to be private or public. Valid values are true or false.
  - *issuesEnabled*: Required. Whether you want to enable issues in GitHub with this repository. Valid values are true or false.
  - *token*: Optional. This is a personal access token AWS CodeStar uses to access your GitHub account. This token must contain the following scopes: **repo**, **user**, and **admin:repo_hook**. To retrieve a personal access token from GitHub, see Creating a Personal Access Token for the Command Line on the GitHub website.

> **Note**
>
> If you use the CLI to create a project with a GitHub source repository, AWS CodeStar uses your token to access the repository through OAuth apps. If you use the console to create a project with a GitHub source repository, AWS CodeStar uses a connection resource, which accesses the repository with GitHub apps.

- *`toolchain`*: Information about the CI/CD toolchain to be set up when the project is created. This includes the location where you uploaded the toolchain template. The template creates the AWS CloudFormation stack that contains your toolchain resources. This also includes any parameter overrides for AWS CloudFormation to reference and the role to be used to create the stack. AWS CodeStar retrieves the template and uses AWS CloudFormation to run the template.
  - *`source`*: Required. The location of your toolchain template. Amazon S3 is the only supported source location.
    - *`S3`*: Optional. The Amazon S3 location where you uploaded your toolchain template.
      - *`bucket-name`*: The Amazon S3 bucket name.
      - *`bucket-key`*: The bucket prefix and object key that points to the .yml or .json file that contains your toolchain template (for example, `files/toolchain.yml`).
  - *`stackParameters`*: Optional. Contains key-value pairs to be passed to AWS CloudFormation. These are the parameters, if any, your toolchain template is set up to reference.
  - *`role`*: Optional. The role used to create your toolchain resources in your account. The role is required as follows:
    - If the role is not provided, AWS CodeStar uses the default service role created for your account if the toolchain is an AWS CodeStar quickstart template. If the service role doesn't exist in your account, you can create one. For information, see Step 2: Create the AWS CodeStar Service Role.
    - You must provide the role must if you are uploading and using your own custom toolchain template. You can create a role based on the AWS CodeStar service role and policy statement. For an example of this policy statement, see AWSCodeStarServiceRole Policy .

- *tags*: Optional. The tags attached to your AWS CodeStar project.

> **Note**
>
> These tags are not attached to the resources contained in the project.

2. Switch to the directory that contains the file you just saved, and run the **create-project** command again. Include the `--cli-input-json` parameter.

```
aws codestar create-project --cli-input-json file://input.json
```

3. If successful, data similar to the following appears in the output:

```
{
    "id": "project-ID",
    "arn": "arn"
}
```

- The output contains information about the new project:
  - The `id` value represents the project ID.
  - The `arn` value represents the ARN of the project.

4. Use the **describe-project** command to check the status of your project creation. Include the `--id` parameter.

```
aws codestar describe-project --id <project_ID>
```

Data similar to the following appears in the output:

```
{
    "name": "MyProject",
    "id": "myproject",
    "arn": "arn:aws:codestar:us-east-1:account_ID:project/myproject",
    "description": "",
    "createdTimeStamp": 1539700079.472,
    "stackId": "arn:aws:cloudformation:us-east-1:account_ID:stack/awscodestar-
myproject/stack-ID",
    "status": {
        "state": "CreateInProgress"
    }
```

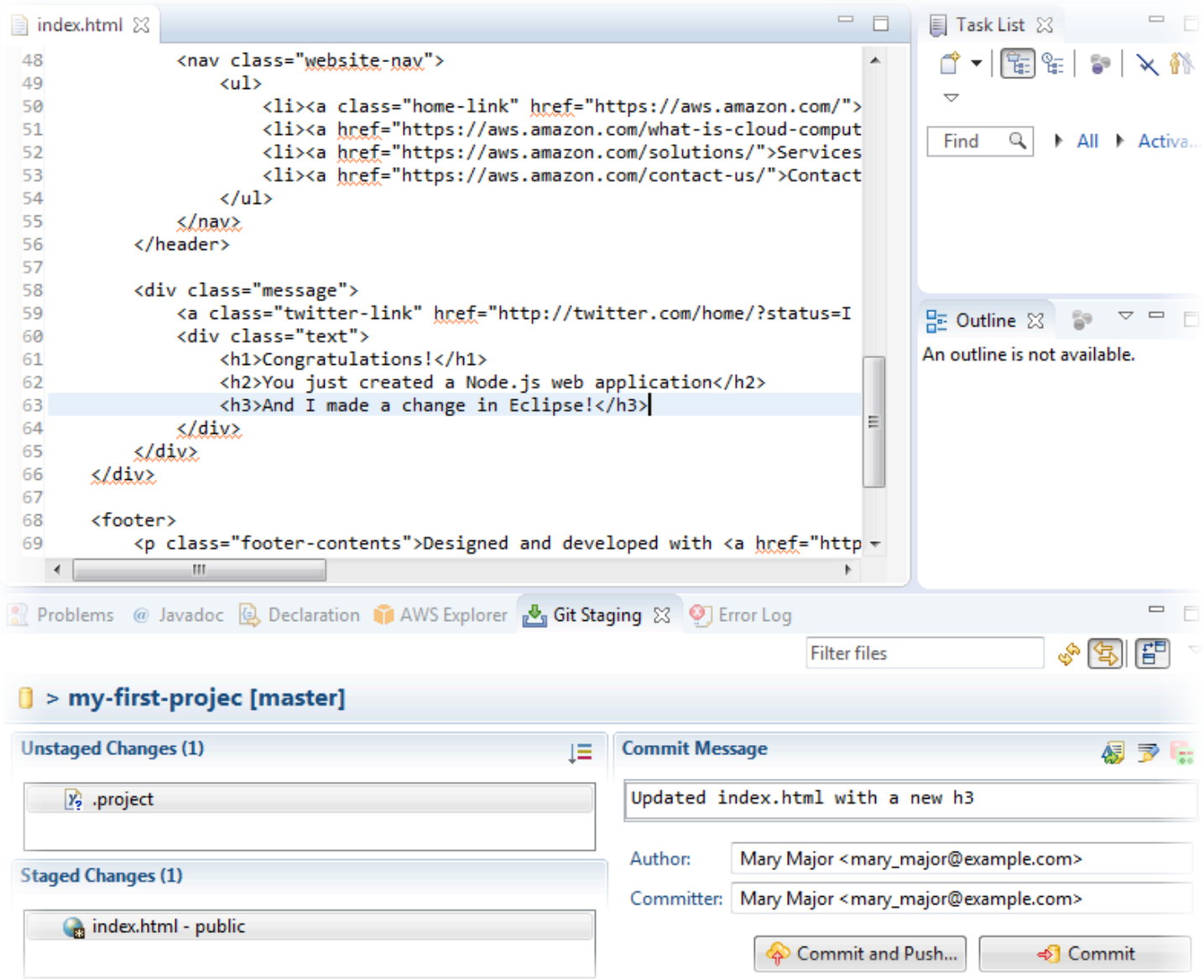
```
}
```

- The output contains information about the new project:
  - The `state` value represents the status of the project creation, such as `CreateInProgress` or `CreateComplete`.

While your project is being created, you can add team members or configure access to your project repository from the command line or your favorite IDE.

Use an IDE with AWS CodeStar

When you integrate an IDE with AWS CodeStar, you can continue to write and develop code in your preferred environment. The changes you make are included in the AWS CodeStar project each time you commit and push your code.



Topics

- Use AWS Cloud9 with AWS CodeStar
- Use Eclipse with AWS CodeStar
- Use Visual Studio with AWS CodeStar

Use AWS Cloud9 with AWS CodeStar

You can use AWS Cloud9 to make code changes and develop software in an AWS CodeStar project. AWS Cloud9 is an online IDE, which you access through your web browser. The IDE offers a rich code editing experience with support for several programming languages and runtime debuggers,

as well as a built-in terminal. In the background, an Amazon EC2 instance hosts an AWS Cloud9 development environment. This environment provides the AWS Cloud9 IDE and access to the AWS CodeStar project's code files. For more information, see the *AWS Cloud9 User Guide*.

You can use the AWS CodeStar console or AWS Cloud9 console to create AWS Cloud9 development environments for projects that store their code in CodeCommit. For AWS CodeStar projects that store their code in GitHub, you can only use the AWS Cloud9 console. This topic describes how to use both consoles.

To use AWS Cloud9, you need:

- An IAM user that has been added as a team member to an AWS CodeStar project.
- If the AWS CodeStar project stores its source code in CodeCommit, AWS credentials for the IAM user.

Topics

- Create an AWS Cloud9 Environment for a Project
- Open an AWS Cloud9 Environment for a Project
- Share an AWS Cloud9 Environment with a Project Team Member
- Delete an AWS Cloud9 Environment from a Project
- Use GitHub with AWS Cloud9
- Additional Resources

Create an AWS Cloud9 Environment for a Project

Follow these steps to create an AWS Cloud9 development environment for a AWS CodeStar project.

1. Follow the steps in Create a Project if you want to create a new project.
2. Open the project in the AWS CodeStar console. On the navigation bar, choose **IDE**. Choose **Create environment**, and then use the following steps.

> **Important**
>
> If the project is in an AWS Region where AWS Cloud9 isn't supported, you won't see AWS Cloud9 options in the **IDE** tab on the navigation bar. However, you can

use the AWS Cloud9 console to create a development environment, open the new environment, and then connect it to the project's AWS CodeCommit repository. Skip the following steps and see Creating an Environment, Opening an Environment, and the AWS CodeCommit Sample in the *AWS Cloud9 User Guide*. For the list of supported AWS Regions, see AWS Cloud9 in the *Amazon Web Services General Reference*.

On **Create AWS Cloud9 environment**, customize the project defaults.

1. To change the default type of Amazon EC2 instance to host the environment, for **Instance type**, choose the instance type.
2. AWS Cloud9 uses Amazon Virtual Private Cloud (Amazon VPC) in your AWS account to communicate with the instance. Depending on how Amazon VPC is set up in your AWS account, do one of the following.

Does the account have a VPC with at least one subnet in that VPC?	Is the VPC you want AWS Cloud9 to use the default VPC in the account?	Does the VPC have a single subnet?	Do this
No	—	—	If no VPC exists, create one. Expand **Network settings**. For **Network (VPC)**, choose **Create VPC** and follow the instructions on the page. For more information, see Create an Amazon VPC for AWS Cloud9 in the *AWS Cloud9 User Guide*. If a VPC exists but has no subnet, create one. Expand **Network settings**. For **Network (VPC)**, choose **Create subnet**, and then follow the instructions. For more information, see Create a

Does the account have a VPC with at least one subnet in that VPC?	Is the VPC you want AWS Cloud9 to use the default VPC in the account?	Does the VPC have a single subnet?	Do this
			Subnet for AWS Cloud9 in the *AWS Cloud9 User Guide*.
Yes	Yes	Yes	Skip ahead to step 4 in this procedure. (AWS Cloud9 uses the default VPC with its single subnet.)
Yes	Yes	No	For **Subnet**, choose the subnet you want AWS Cloud9 to use in the preselected default VPC.
Yes	No	Yes or No	For **Network (VPC)**, choose the VPC that you want AWS Cloud9 to use. For **Subnet**, choose the subnet you want AWS Cloud9 to use in that VPC.

For more information, see Amazon VPC Settings for AWS Cloud9 Development Environments in the *AWS Cloud9 User Guide*.

3. Enter an **Environment name** and optionally add an **Environment description**.

> **Note**
>
> Environment names must be unique per user.

4. To change the default time period after which AWS Cloud9 shuts down the environment when it has not been used, expand **Cost-saving settings**, and then change the setting.
5. Choose **Create environment**.

To open the environment, see Open an AWS Cloud9 Environment for a Project.

You can use these steps to create more than one environment for a project. For example, you might want to use one environment to work on one portion of the code and another environment to work on the same portion of the code with different settings.

Open an AWS Cloud9 Environment for a Project

Follow these steps to open an AWS Cloud9 development environment that you created for an AWS CodeStar project.

1. With the project open in the AWS CodeStar console, on the navigation bar, choose **IDE**.

   > **Important**
   >
   > If the project's source code is stored in GitHub, you won't see **IDE** on the navigation bar. However, you can use the AWS Cloud9 console to open an existing environment. Skip the rest of this procedure and see Opening an Environment in the *AWS Cloud9 User Guide* and Use GitHub with AWS Cloud9.

2. For **Your AWS Cloud9 environments** or **Shared AWS Cloud9 environments**, choose **Open IDE** for the environment you want to open.

You can use the AWS Cloud9 IDE to start working with code in the project's AWS CodeCommit repository right away. For more information, see The Environment Window, The Editor, Tabs, and Panes, and The Terminal in the *AWS Cloud9 User Guide* and Basic Git Commands in the *AWS CodeCommit User Guide*.

Share an AWS Cloud9 Environment with a Project Team Member

After you create an AWS Cloud9 development environment for an AWS CodeStar project, you can invite other users across your AWS account, including project team members, to access that same environment. This is especially useful for pair programming, where two programmers take turns coding and giving advice about the same code through screen sharing or while sitting at the same workstation. Environment members can use the shared AWS Cloud9 IDE to see each member's code changes highlighted in the code editor and to text chat with other members while coding.

Adding a team member to a project doesn't automatically allow that member to participate in any related AWS Cloud9 development environments for the project. To invite a project team member to access an environment for a project, you need to determine the correct environment member access role, apply AWS managed policies to the user, and invite the user to your environment. For

more information, see About Environment Member Access Roles and Invite an IAM user to Your Environment in the *AWS Cloud9 User Guide*.

When you invite a project team member to access an environment for a project, the AWS CodeStar console displays the environment to that team member. The environment is displayed in the **Shared environments** list on the **IDE** tab in the AWS CodeStar console for the project. To display this list, have the team member open the project in the console, and then choose **IDE** in the navigation bar.

> **Important**
>
> If the project's source code is stored in GitHub, you won't see **IDE** on the navigation bar. However, you can use the AWS Cloud9 console to invite other users across your AWS account, including project team members, to access an environment. To do this, see Use GitHub with AWS Cloud9 in this guide, and see About Environment Member Access Roles and Invite an IAM user to Your Environment in the *AWS Cloud9 User Guide*.

You can also invite a user who is not a project team member to access an environment. For example, you might want a user to work on a project's code but have no other access to that project. To invite this type of user, see About Environment Member Access Roles and Invite an IAM user to Your Environmentin the *AWS Cloud9 User Guide*. When you invite a user who is not a project team member to access an environment for a project, that user can use the AWS Cloud9 console to access the environment. For more information, see Open an Environment in the *AWS Cloud9 User Guide*.

Delete an AWS Cloud9 Environment from a Project

When you delete a project and all its AWS resources from AWS CodeStar, all related AWS Cloud9 development environments that were created with the AWS CodeStar console are also deleted and cannot be recovered. You can delete a development environment from a project without deleting the project.

1. With the project open in the AWS CodeStar console, in the navigation bar, choose **IDE**.

   > **Important**
   >
   > If the project's source code is stored in GitHub, you won't see **IDE** on the navigation bar. However, you can use the AWS Cloud9 console to delete a development environment.

Skip the rest of this procedure and see Deleting an Environment in the *AWS Cloud9 User Guide*.

2. Choose the environment you want to delete in **Cloud9 environments** and choose **Delete**
3. Enter **`delete`** to confirm deletion for the development environment, and then choose **Delete**.

> **Warning**
>
> You cannot recover a development environment after it has been deleted. All uncommitted code changes in the environment are lost.

Use GitHub with AWS Cloud9

For AWS CodeStar projects that have their source code stored in GitHub, the AWS CodeStar console doesn't support working with AWS Cloud9 development environments directly. However, you can use the AWS Cloud9 console to work with source code in GitHub repositories.

1. Use the AWS Cloud9 console to create an AWS Cloud9 development environment. For information, see Creating an Environment in the *AWS Cloud9 User Guide*.
2. Use the AWS Cloud9 console to open the development environment. For information, see Opening an Environment in the *AWS Cloud9 User Guide*.
3. In the IDE, use a terminal session to connect to the GitHub repository (a process known as *cloning*). If a terminal session isn't running, on the menu bar in the IDE, choose **Window, New Terminal**. For the commands to use to clone the GitHub repository, see Cloning a Repository on the GitHub Help website.

   To navigate to the main page of the GitHub repository, with the project open in the AWS CodeStar console, on the side navigation bar, choose **Code**.
4. Use the **Environment** window and editor tabs in the IDE to view, change, and save code. For more information, see The Environment Window and The Editor, Tabs, and Panes in the *AWS Cloud9 User Guide*.
5. Use Git in the IDE's terminal session to push your code changes to the repository and periodically pull code changes from others from the repository. For more information, see Pushing to a Remote Respository and Fetching a Remote Repository on the GitHub Help website. For Git commands, see Git Cheatsheet on the GitHub Help website.

> **Note**
>
> To keep Git from prompting you for your GitHub sign-in credentials every time you push or pull code from the repository, you can use a *credential helper*. For more information, see Caching Your GitHub Password in Git on the GitHub Help website.

Additional Resources

For more information about using AWS Cloud9, see the following in the *AWS Cloud9 User Guide*:

- Tutorial
- Working with Environments
- Working with the IDE
- Samples

Use Eclipse with AWS CodeStar

You can use Eclipse to make code changes and develop software in an AWS CodeStar project. You can edit your AWS CodeStar project code with Eclipse and then commit and push your changes to the source repository for the AWS CodeStar project.

> **Note**
>
> The information in this topic applies only to AWS CodeStar projects that store their source code in CodeCommit. If your AWS CodeStar project stores its source code in GitHub, you can use a tool such as EGit for Eclipse. For more information, see the EGit Documentation on the EGit website.

If the AWS CodeStar project stores its source code in CodeCommit, you must install a version of the AWS Toolkit for Eclipse that supports AWS CodeStar. You must also be a member of the AWS CodeStar project team with the owner or contributor role.

To use Eclipse, you also need:

- An IAM user that has been added to an AWS CodeStar project as a team member.

- If the AWS CodeStar project stores its source code in CodeCommit, Git credentials (sign-in credentials) for the IAM user.
- Sufficient permissions to install Eclipse and the AWS Toolkit for Eclipse on your local computer.

Topics

- Step 1: Install AWS Toolkit for Eclipse
- Step 2: Import Your AWS CodeStar Project to Eclipse
- Step 3: Edit AWS CodeStar Project Code in Eclipse

Step 1: Install AWS Toolkit for Eclipse

The Toolkit for Eclipse is a software package you can add to Eclipse. It is installed and managed in the same way as other software packages in Eclipse. The AWS CodeStar toolkit is included as part of the Toolkit for Eclipse.

To install the Toolkit for Eclipse with the AWS CodeStar module

1. Install Eclipse on your local computer. Supported versions of Eclipse include Luna, Mars, and Neon.
2. Download and install the Toolkit for Eclipse. For more information, see the AWS Toolkit for Eclipse Getting Started Guide.
3. In Eclipse, choose **Help**, and then choose **Install New Software**.
4. In **Available Software**, choose **Add**.
5. In **Add Repository**, choose **Archive**, browse to the location where you saved the .zip file, and open the file. Leave **Name** blank, and then choose **OK**.
6. In **Available Software**, choose **Select all** to select **AWS Core Management Tools** and **Developer Tools**, and then choose **Next**.
7. In **Install Details**, choose **Next**.
8. In **Review Licenses**, review the license agreements. Choose **I accept the terms of the license agreement**, and then choose **Finish**. Restart Eclipse.

Step 2: Import Your AWS CodeStar Project to Eclipse

After you have installed the Toolkit for Eclipse, you can import AWS CodeStar projects and edit, commit, and push code from the IDE.

> **Note**
>
> You can add multiple AWS CodeStar projects to a single workspace in Eclipse, but you must update your project credentials when you change from one project to another.

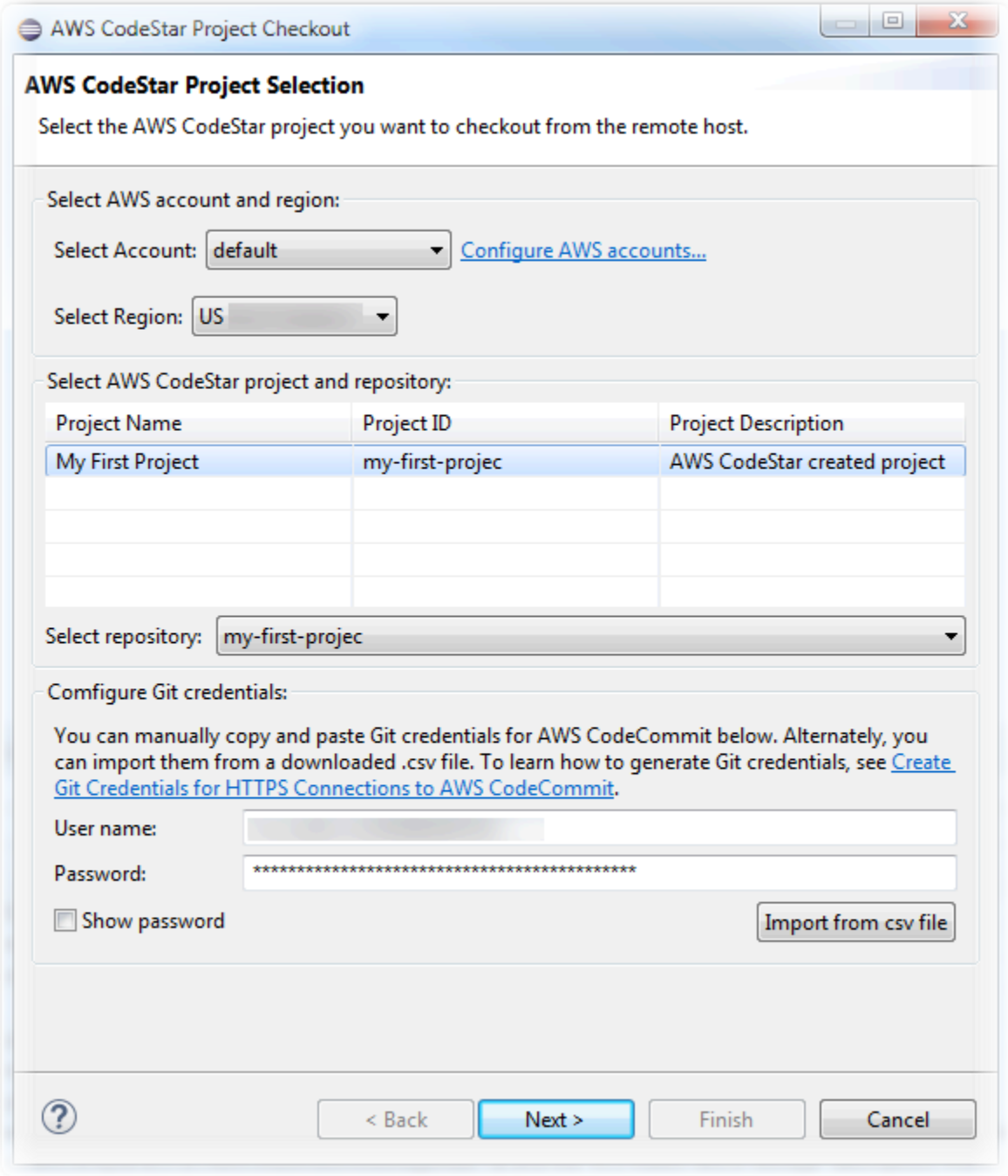
To import an AWS CodeStar project

1. From the AWS menu, choose **Import AWS CodeStar Project**. Alternatively, choose **File**, and then choose **Import**. In **Select**, expand **AWS**, and then choose **AWS CodeStar Project**.

   Choose **Next**.

2. In **AWS CodeStar Project Selection**, choose your AWS profile and the AWS Region where the AWS CodeStar project is hosted. If you do not have an AWS profile configured with an access key and secret key on your computer, choose **Configure AWS accounts** and follow the instructions.

   In **Select AWS CodeStar project and repository**, choose your AWS CodeStar project. In **Configure Git credentials**, enter the sign-in credentials you generated for access to the project's repository. (If you don't have Git credentials, see Getting Started.) Choose **Next**.



3. All branches of the project's repository are selected by default. If you don't want to import one or more branches, clear the boxes, and then choose **Next**.
4. In **Local Destination**, choose a destination where the import wizard creates the local repo on your computer, and then choose **Finish**.
5. In **Project Explorer**, expand the project tree to browse the files in the AWS CodeStar project.

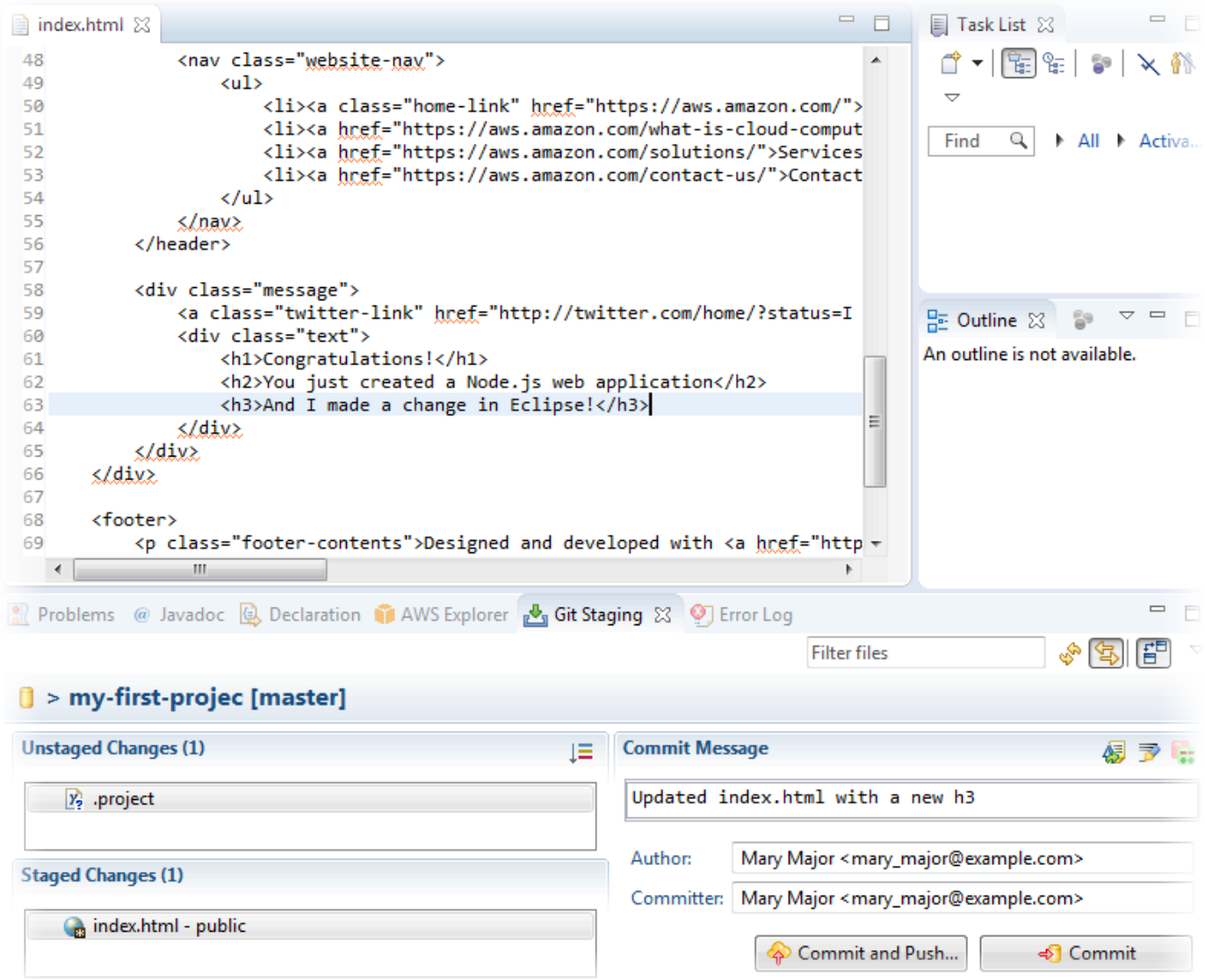
Step 3: Edit AWS CodeStar Project Code in Eclipse

After you have imported an AWS CodeStar project into an Eclipse workspace, you can edit the code for the project, save your changes, and commit and push your code to the source repository for the project. This is the same process you follow for any Git repository using the EGit plugin for Eclipse. For more information, see the EGit User Guide on the Eclipse website.

To edit project code and make your first commit to the source repository for an AWS CodeStar project

1. In **Project Explorer**, expand the project tree to browse the files in the AWS CodeStar project.
2. Edit one or more code files and save your changes.
3. When you are ready to commit your changes, open the context menu for that file, choose **Team**, and then choose **Commit**.

   You can skip this step if the **Git Staging** window is already open in your project view.
4. In **Git Staging**, stage your changes by moving changed files into **Staged Changes**. Enter a commit message in **Commit Message**, and then choose **Commit and Push**.



To view the deployment of your code changes, return to the dashboard for your project. For more information, see [Step 3: View Your Project](#).

Use Visual Studio with AWS CodeStar

You can use Visual Studio to make code changes and develop software in an AWS CodeStar project.

> **Note**
>
> Visual Studio for Mac does not support the AWS Toolkit so it cannot be used with AWS CodeStar.

> The information in this topic applies only to AWS CodeStar projects that store their source code in CodeCommit. If your AWS CodeStar project stores its source code in GitHub, you can use a tool such as the GitHub Extension for Visual Studio. For more information, see the Overview page on the GitHub Extension for Visual Studio website and Getting Started with GitHub for Visual Studio on the GitHub website.

To use Visual Studio to edit code in the source repository for an AWS CodeStar project, you must install a version of the AWS Toolkit for Visual Studio that supports AWS CodeStar. You must be a member of the AWS CodeStar project team with the owner or contributor role.

To use Visual Studio, you also need:

- An IAM user that has been added to an AWS CodeStar project as a team member.
- AWS credentials for your IAM user (for example, your access key and secret key).
- Sufficient permissions to install Visual Studio and the AWS Toolkit for Visual Studio on your local computer.

The Toolkit for Visual Studio is a software package you can add to Visual Studio. It is installed and managed in the same way as other software packages in Visual Studio.

To install the Toolkit for Visual Studio with the AWS CodeStar module and configure access to your project repository

1. Install Visual Studio on your local computer.
2. Download and install the Toolkit for Visual Studio and save the .zip file to a local folder or directory. On the **Getting Started with the AWS Toolkit for Visual Studio** page, enter or import your AWS credentials, and then choose **Save and Close**.
3. In **Visual Studio**, open **Team Explorer**. In **Hosted Service Providers**, find **CodeCommit**, and choose **Connect**.
4. In **Manage Connections**, choose **Clone**. Choose your project's repository and the folder on your local computer where you want to clone the repository, and then choose **OK**.
5. If you are prompted to create Git credentials, choose **Yes**. The toolkit attempts to create credentials on your behalf. Save the credentials file in a secure location. This is the only opportunity you have to save these credentials. If the toolkit cannot create credentials on your behalf, or if you chose **No**, you must create and provide your own Git credentials. For more

information, see To set up your computer to commit changes (IAM user) or follow the online directions.

When you have finished cloning the project, you're ready to start editing your code in Visual Studio and committing and pushing your changes to your project's repository in CodeCommit.

Change AWS Resources in an AWS CodeStar Project

After you create a project in AWS CodeStar, you can change the default set of AWS resources that AWS CodeStar adds to the project.

Supported Resource Changes

The following table lists the supported changes to default AWS resources in an AWS CodeStar project.

Change	Notes
Add a stage to AWS CodePipeline.	See Add a Stage to AWS CodePipeline.
Change Elastic Beanstalk environment settings.	See Change AWS Elastic Beanstalk Environment Settings.
Change an AWS Lambda function's code or settings, its IAM role, or its API in Amazon API Gateway.	See Change an AWS Lambda Function in Source Code.
Add a resource to an AWS Lambda project and expand permissions to create and access the new resource.	See Add a Resource to a Project.
Add traffic shifting with CodeDeploy for an AWS Lambda function.	See Shift Traffic for an AWS Lambda Project.
Add AWS X-Ray support	See Enable Tracing for a Project.
Edit the buildspec.yml file for your project to add a unit test build phase for AWS CodeBuild to run.	See Step 7: Add a Unit Test to the Web Service in the Serverless Project tutorial.

Change	Notes
Add your own IAM role to your project.	See Add an IAM Role to a Project.
Change an IAM role definition.	For roles defined in the application stack. You cannot change roles defined in the toolchain or AWS CloudFormation stacks.
Modify your Lambda project to add an endpoint.	
Modify your EC2 project to add an endpoint.	
Modify your Elastic Beanstalk project to add an endpoint.	
Edit your project to add a Prod stage and endpoint.	See Add a Prod Stage and Endpoint to a Project.
Securely use SSM parameters in an AWS CodeStar project.	See the section called "Securely Use SSM Parameters in an AWS CodeStar Project".

The following changes are not supported.

- Switch to a different deployment target (for example, deploy to AWS Elastic Beanstalk instead of AWS CodeDeploy).
- Add a friendly web endpoint name.
- Change the CodeCommit repository name (for an AWS CodeStar project connected to CodeCommit).
- For an AWS CodeStar project connected to GitHub, disconnect the GitHub repository, and then reconnect the repository to that project, or connect any other repository to that project. You can use the CodePipeline console (not the AWS CodeStar console) to disconnect and reconnect to GitHub in a pipeline's **Source** stage. However, if you reconnect the **Source** stage to a different GitHub repository, in the AWS CodeStar dashboard for the project, the information in the **Repository** and **Issues** tiles might be wrong or out of date. Disconnecting the GitHub repository does not remove that repository's information from the commit history and GitHub issues tiles in the AWS CodeStar project dashboard. To remove this information, use the GitHub website

to disable access to GitHub from the AWS CodeStar project. To revoke access, on the GitHub website, use the **Authorized OAuth Apps** section of the settings page for your GitHub account profile.

- Disconnect the CodeCommit repository (for an AWS CodeStar project connected to CodeCommit) and then reconnect the repository to that project, or connect any other repository to that project.

Add a Stage to AWS CodePipeline

You can add a new stage to a pipeline that AWS CodeStar creates in a project. For more information, see Edit a Pipeline in AWS CodePipeline in the *AWS CodePipeline User Guide*.

> **Note**
>
> If the new stage depends on any AWS resources that AWS CodeStar did not create, the pipeline might break. This is because the IAM role that AWS CodeStar created for AWS CodePipeline might not have access to those resources by default.
> To attempt to give AWS CodePipeline access to AWS resources that AWS CodeStar did not create, you might want to change the IAM role that AWS CodeStar created. This is not supported because AWS CodeStar might remove your IAM role changes when it performs regular update checks on the project.

Change AWS Elastic Beanstalk Environment Settings

You can change the settings of an Elastic Beanstalk environment that AWS CodeStar creates in a project. For example, you might want to change the default Elastic Beanstalk environment in your AWS CodeStar project from Single Instance to Load Balanced. To do this, edit the `template.yml` file in your project's repository. You might also need to change permissions for your project's worker roles. After you push the template change, AWS CodeStar and AWS CloudFormation provision the resources for you.

For more information about editing the `template.yml` file, see Change Application Resources with the Template.yml File. For more information about Elastic Beanstalk environments, see AWS Elastic Beanstalk Environment Management Console in the *AWS Elastic Beanstalk Developer Guide*.

Change an AWS Lambda Function in Source Code

You can change the code or settings of a Lambda function, or its IAM role or API Gateway API, that AWS CodeStar creates in a project. To do this, we recommend that you use the AWS Serverless Application Model (AWS SAM) along with the `template.yaml` file in your project's CodeCommit repository. This `template.yaml` file defines your function's name, handler, runtime, IAM role, and API in API Gateway. For more information, see How to Create Serverless Applications Using AWS SAM on the GitHub website.

Enable Tracing for a Project

AWS X-Ray offers tracing, which you can use to analyze performance behavior of distributed applications (for example, latencies in response times). After you add traces to your AWS CodeStar project, you can use the AWS X-Ray console to view application views and response times.

> **Note**
>
> You can use these steps for the following projects, created with the following project support changes:
>
> - Any Lambda project.
> - For Amazon EC2 or Elastic Beanstalk projects created after August 3, 2018, AWS CodeStar provisioned a `/template.yml`file in the project repository.

Each AWS CodeStar template includes an AWS CloudFormation file that models your application's AWS runtime dependencies, such as database tables and Lambda functions. This file is stored in your source repository in the file `/template.yml`.

You can modify this file to add tracing by adding the AWS X-Ray resource to the `Resources` section. You then modify the IAM permissions for your project to allow AWS CloudFormation to create the resource. For information about template elements and formatting, see AWS Resource Types Reference.

These are the high-level steps to follow to customize your template.

1. Step 1: Edit the Worker Role in IAM for Tracing
2. Step 2: Modify the template.yml File for Tracing

3. Step 3: Commit and Push Your Template Change for Tracing
4. Step 4: Monitor the AWS CloudFormation Stack Update for Tracing

Step 1: Edit the Worker Role in IAM for Tracing

You must be signed in as an administrator to perform steps 1 and 4. This step shows an example of editing permissions for a Lambda project.

> **Note**
>
> You can skip this step if your project was provisioned with a permissions boundary policy. For projects created after December 6, 2018 PDT, AWS CodeStar provisioned your project with a permissions boundary policy.

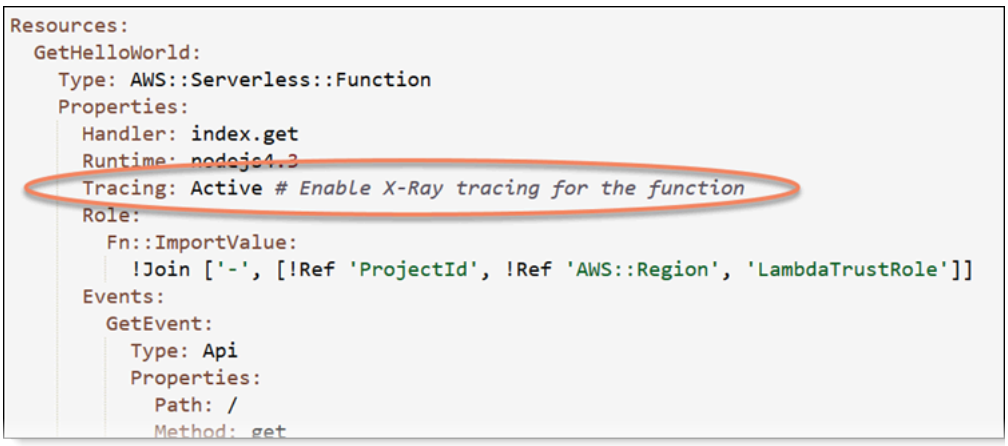
1. Sign in to the AWS Management Console and open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Create a project or choose an existing project with a `template.yml file`, and then open the **Project resources** page.
3. Under **Project Resources**, locate the IAM role created for the CodeStarWorker/Lambda role in the resource list. The role name follows this format: `role/CodeStarWorker-`*`Project_name`*`-lambda-`*`Function_name`*. Choose the ARN for the role.
4. The role opens in the IAM console. Choose **Attach policies**. Search for the `AWSXrayWriteOnlyAccess` policy, select the box next to it, and then choose **Attach Policy**.

Step 2: Modify the template.yml File for Tracing

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose your serverless project and then open the **Code** page. In the top level of your repository, locate and edit the `template.yml` file. Under `Resources`, paste the resource into the `Properties` section .

   ```
       Tracing: Active
   ```

   This example shows a modified template:



Step 3: Commit and Push Your Template Change for Tracing

- Commit and push the changes in the `template.yml` file.

> **Note**
>
> This starts your pipeline. If you commit the changes before you update IAM permissions, your pipeline starts, the AWS CloudFormation stack update encounters errors, and the stack update is rolled back. If this happens, correct the permissions and then restart your pipeline.

Step 4: Monitor the AWS CloudFormation Stack Update for Tracing

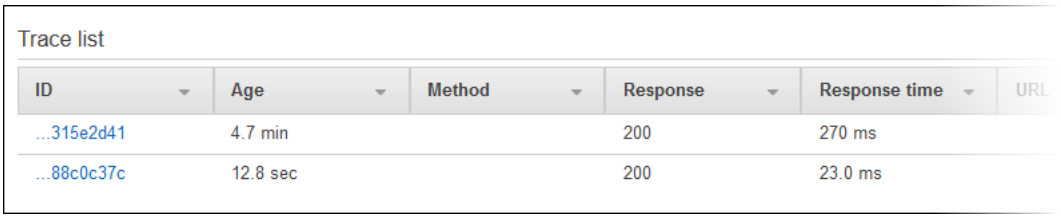
1. The AWS CloudFormation stack update starts when the pipeline for your project starts the Deploy stage. To see the status of the stack update, on your AWS CodeStar dashboard, choose the AWS CloudFormation stage in your pipeline.

   If the stack update in AWS CloudFormation returns errors, see troubleshooting guidelines in AWS CloudFormation: Stack Creation Rolled Back for Missing Permissions. If permissions are missing from the worker role, edit the policy attached to your project's Lambda worker role. See Step 1: Edit the Worker Role in IAM for Tracing.

2. Use the dashboard to view the successful completion of your pipeline. Tracing is now enabled on your application.

3. Verify that tracing is enabled by viewing the details for your function in the Lambda console.

4. Choose the application endpoint for your project. This interaction with your application is traced. You can view tracing information in the AWS X-Ray console.



Add a Resource to a Project

Each AWS CodeStar template for all projects comes with an AWS CloudFormation file that models your application's AWS runtime dependencies, such as database tables and Lambda functions. This is stored in your source repository in the file `/template.yml`.

> **Note**
>
> You can use these steps for the following projects, created with the following project support changes:
>
> - Any Lambda project.
> - For Amazon EC2 or Elastic Beanstalk projects created after August 3, 2018, AWS CodeStar provisioned a `/template.yml` file in the project repository.

You can modify this file by adding AWS CloudFormation resources to the `Resources` section. Modifying the `template.yml` file allows AWS CodeStar and AWS CloudFormation to add the new resource to your project. Some resources require you to add other permissions to the policy for your project's CloudFormation worker role. For information about template elements and formatting, see AWS Resource Types Reference.

After you determine which resources you must add to your project, these are the high-level steps to follow to customize a template. For a list of AWS CloudFormation resources and their required properties, see AWS Resource Types Reference.

1. Step 1: Edit the CloudFormation Worker Role in IAM (if necessary)
2. Step 2: Modify the template.yml File
3. Step 3: Commit and Push Your Template Change

4. Step 4: Monitor the AWS CloudFormation Stack Update
5. Step 5: Add Resource Permissions with an Inline Policy

Use the steps in this section to modify your AWS CodeStar project template to add a resource and then expand the project's CloudFormation worker role's permissions in IAM. In this example, the AWS::SQS::Queue resource is added to the `template.yml` file. The change starts an automated response in AWS CloudFormation that adds an Amazon Simple Queue Service queue to your project.

Step 1: Edit the CloudFormation Worker Role in IAM

You must be signed in as an administrator to perform steps 1 and 5.

> **Note**
>
> You can skip this step if your project was provisioned with a permissions boundary policy. For projects created after December 6, 2018 PDT, AWS CodeStar provisioned your project with a permissions boundary policy.

1. Sign in to the AWS Management Console and open the AWS CodeStar console, at https://console.aws.amazon.com/codestar/.
2. Create a project or choose an existing project with a `template.yml file`, and then open the **Project resources** page.
3. Under **Project Resources**, locate the IAM role created for the CodeStarWorker/AWS CloudFormation role in the resource list. The role name follows this format: `role/CodeStarWorker-`*`Project_name`*`-CloudFormation`.
4. The role opens in the IAM console. On the **Permissions** tab, in **Inline Policies**, expand the row for your service role policy, and choose **Edit Policy**.
5. Choose the **JSON** tab to edit the policy.

   > **Note**
   >
   > The policy attached to the worker role is `CodeStarWorkerCloudFormationRolePolicy`.

6. In the **JSON** field, add the following policy statement in the `Statement` element.

```
{
  "Action": [
    "sqs:CreateQueue",
    "sqs:DeleteQueue",
    "sqs:GetQueueAttributes",
    "sqs:SetQueueAttributes",
    "sqs:ListQueues",
    "sqs:GetQueueUrl"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow"
}
```

7. Choose **Review policy** to ensure the policy contains no errors, and then choose **Save changes**.

Step 2: Modify the template.yml File

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose your serverless project and then open the **Code** page. In the top level of your repository, make a note of the location of `template.yml`.
3. Use an IDE, the console, or the command line in your local repository to edit the `template.yml` file in your repository. Paste the resource into the `Resources` section. In this example, when the following text is copied, it adds the `Resources` section.

```
Resources:
  TestQueue:
    Type: AWS::SQS::Queue
```

This example shows a modified template:

```
Resources:
  HelloWorld:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.6
      Role:
        Fn::ImportValue:
          !Join ['-', [!Ref 'ProjectId', !Ref 'AWS::Region', 'LambdaTrustRole']]
      Events:
        GetEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
        PostEvent:
          Type: Api
          Properties:
            Path: /
            Method: post

  TestQueue:
    Type: AWS::SQS::Queue
```

Step 3: Commit and Push Your Template Change

- Commit and push the changes in the `template.yml` file that you saved in step 2.

> **Note**
>
> This starts your pipeline. If you commit the changes before you update IAM permissions, your pipeline starts and the AWS CloudFormation stack update encounters errors, which causes the stack update to be rolled back. If this happens, correct the permissions and then restart your pipeline.

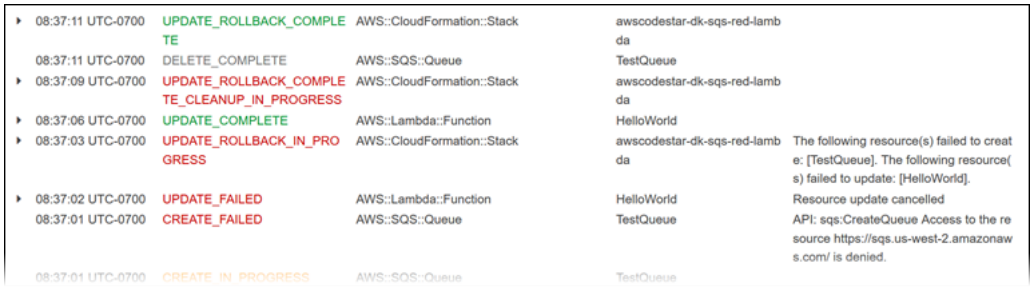
Step 4: Monitor the AWS CloudFormation Stack Update

1. When the pipeline for your project starts the Deploy stage, the AWS CloudFormation stack update starts. You can choose the AWS CloudFormation stage in your pipeline on your AWS CodeStar dashboard to see the stack update.

   Troubleshooting:

   The stack update fails if required resource permissions are missing. View the failure status in the AWS CodeStar dashboard view for your project's pipeline.

Choose the **CloudFormation** link in your pipeline's Deploy stage to troubleshoot the failure in the AWS CloudFormation console. In the console, in the **Events** list, choose your project to view stack creation details. There is a message with the failure details. In this example, the `sqs:CreateQueue` permission is missing.



08:37:11 UTC-0700	UPDATE_ROLLBACK_COMPLETE	AWS::CloudFormation::Stack	awscodestar-dk-sqs-red-lambda	
08:37:11 UTC-0700	DELETE_COMPLETE	AWS::SQS::Queue	TestQueue	
08:37:09 UTC-0700	UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	AWS::CloudFormation::Stack	awscodestar-dk-sqs-red-lambda	
08:37:06 UTC-0700	UPDATE_COMPLETE	AWS::Lambda::Function	HelloWorld	
08:37:03 UTC-0700	UPDATE_ROLLBACK_IN_PROGRESS	AWS::CloudFormation::Stack	awscodestar-dk-sqs-red-lambda	The following resource(s) failed to create: [TestQueue]. The following resource(s) failed to update: [HelloWorld].
08:37:02 UTC-0700	UPDATE_FAILED	AWS::Lambda::Function	HelloWorld	Resource update cancelled
08:37:01 UTC-0700	CREATE_FAILED	AWS::SQS::Queue	TestQueue	API: sqs:CreateQueue Access to the resource https://sqs.us-west-2.amazonaws.com/ is denied.

Add any missing permissions by editing the policy attached to your project's AWS CloudFormation worker role. See Step 1: Edit the CloudFormation Worker Role in IAM.

2. After a successful run of your pipeline, the resources are created in your AWS CloudFormation stack. In the **Resources** list in AWS CloudFormation, view the resource created for your project. In this example, the TestQueue queue is listed in the **Resources** section.

   The queue URL is available in AWS CloudFormation. The queue URL follows this format:

   ```
   https://{REGION_ENDPOINT}/queue.|api-domain|/{YOUR_ACCOUNT_NUMBER}/
   {YOUR_QUEUE_NAME}
   ```

   For more information, see Send an Amazon SQS Message, Receive a Message from an Amazon SQS Queue, and Delete a Message from an Amazon SQS Queue.

Step 5: Add Resource Permissions with an Inline Policy

Grant team members access to your new resource by adding the appropriate inline policy to the user's role. Not all resources require that you add permissions. To perform the following steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the `AdministratorAccess` managed policy or equivalent.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. At the top of the page, choose **Create policy**.

4. In the **Policy editor** section, choose the **JSON** option.

5. Enter the following JSON policy document:

```
 {
  "Action": [
    "sqs:CreateQueue",
    "sqs:DeleteQueue",
    "sqs:GetQueueAttributes",
    "sqs:SetQueueAttributes",
    "sqs:ListQueues",
    "sqs:GetQueueUrl"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow"
}
```

6. Choose **Next**.

> **Note**
>
> You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy restructuring in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.

8. Choose **Create policy** to save your new policy.

Add an IAM Role to a Project

As of December 6, 2018 PDT you can define your own roles and polices in the application stack (template.yml). To mitigate risks of privilege escalation and destructive actions, you are required to set the project-specific permissions boundary for every IAM entity you create. If you have a Lambda project with multiple functions, it is a best practice to create an IAM role for each function.

To add an IAM role to your project

1. Edit the `template.yml` file for your project.
2. In the `Resources:` section, add your IAM resource, using the format in the following example:

```
  SampleRole:
  Description: Sample Lambda role
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
      - Effect: Allow
        Principal:
          Service: [lambda.amazonaws.com]
        Action: sts:AssumeRole
    ManagedPolicyArns:
      -  arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
    PermissionsBoundary: !Sub 'arn:${AWS::Partition}:iam::${AWS::AccountId}:policy/
CodeStar_${ProjectId}_PermissionsBoundary'
```

3. Release your changes through the pipeline and verify success.

Add a Prod Stage and Endpoint to a Project

Use the procedures in this section to add a new production (Prod) stage to your pipeline and a manual approval stage between your pipeline's Deploy and Prod stages. This creates an additional resource stack when your project pipeline runs.

> **Note**
>
> You can use these procedures if:

- For projects created after August 3, 2018, AWS CodeStar provisioned your Amazon EC2, Elastic Beanstalk, or Lambda project with a `/template.yml` file in the project repository.
- For projects created after December 6, 2018 PDT, AWS CodeStar provisioned your project with a permissions boundary policy.

All AWS CodeStar projects use an AWS CloudFormation template file that models your application's AWS runtime dependencies, such as Linux instances and Lambda functions. The `/template.yml` file is stored in your source repository.

In the `/template.yml` file, use the `Stage` parameter to add a resource stack for a new stage in the project pipeline.

```
  Stage:
    Type: String
    Description: The name for a project pipeline stage, such as Staging or Prod, for
which resources are provisioned and deployed.
    Default: ''
```

The `Stage` parameter is applied to all named resources with the project ID referenced in the resource. For example, the following role name is a named resource in the template:

```
RoleName: !Sub 'CodeStar-${ProjectId}-WebApp${Stage}'
```

Prerequisites

Use the template options in the AWS CodeStar console to create a project.

Make sure your IAM user has the following permissions:

- `iam:PassRole` on the project AWS CloudFormation role.
- `iam:PassRole` on the project toolchain role.
- `cloudformation:DescribeStacks`
- `cloudformation:ListChangeSets`

For Elastic Beanstalk or Amazon EC2 projects only:

- `codedeploy:CreateApplication`
- `codedeploy:CreateDeploymentGroup`
- `codedeploy:GetApplication`
- `codedeploy:GetDeploymentConfig`
- `codedeploy:GetDeploymentGroup`
- `elasticloadbalancing:DescribeTargetGroups`

Topics

- Step 1: Create a new Deployment Group in CodeDeploy (Amazon EC2 Projects Only)
- Step 2: Add a New Pipeline Stage for the Prod Stage
- Step 3: Add a Manual Approval Stage
- Step 4: Push a Change and Monitor the AWS CloudFormation Stack Update

Step 1: Create a new Deployment Group in CodeDeploy (Amazon EC2 Projects Only)

You choose your CodeDeploy application and then add a new deployment group associated with the new instance.

> **Note**
>
> If your project is a Lambda or Elastic Beanstalk project, you can skip this step.

1. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy.
2. Choose the CodeDeploy application that was generated for your project when it was created in AWS CodeStar.
3. Under **Deployment groups**, choose **Create deployment group**.
4. In **Deployment group name**, enter ***<project-id>-prod-Env***.
5. In **Service role**, choose the toolchain worker role for your AWS CodeStar project.
6. Under **Deployment type**, choose **In-place**.
7. Under **Environment configuration**, choose the **Amazon EC2 Instances** tab.

8. Under the tag group, under **Key**, choose `aws:cloudformation:stack-name`. Under **Value**, choose `awscodestar-<projectid>-infrastructure-prod` (the stack to be created for the **GenerateChangeSet** action).
9. In **Deployment settings**, choose `CodeDeployDefault.AllAtOnce`.
10. Clear **Choose a load balancer**.
11. Choose **Create deployment group**.

    Now your second deployment group has been created.

Step 2: Add a New Pipeline Stage for the Prod Stage

Add a stage with the same set of deployment actions as your project's Deploy stage. For example, the new Prod stage for an Amazon EC2 project should have the same actions as the Deploy stage created for the project.

To copy parameters and fields from the Deploy stage

1. From your AWS CodeStar project dashboard, choose **Pipeline Details** to open your pipeline in the CodePipeline console.
2. Choose **Edit**.
3. In the Deploy stage, choose **Edit stage**.
4. Choose the edit icon on the **GenerateChangeSet** action. Make a note of the values in the following fields. You use these values when you create your new action.

   - **Stack name**
   - **Change set name**
   - **Template**
   - **Template configuration**
   - **Input artifacts**
5. Expand **Advanced**, and in **Parameters**, copy the parameters for your project. You paste these parameters into your new action. For example, copy the parameters that are shown here in JSON format:

   - Lambda projects:

```
{
```

```
  "ProjectId":"MyProject"
}
```

- Amazon EC2 projects:

```
{

  "ProjectId":"MyProject",
  "InstanceType":"t2.micro",
  "WebAppInstanceProfile":"awscodestar-MyProject-WebAppInstanceProfile-
EXAMPLEY5VSFS",
  "ImageId":"ami-EXAMPLE1",
  "KeyPairName":"my-keypair",
  "SubnetId":"subnet-EXAMPLE",
  "VpcId":"vpc-EXAMPLE1"
}
```

- Elastic Beanstalk projects:

```
{
  "ProjectId":"MyProject",
  "InstanceType":"t2.micro",
  "KeyPairName":"my-keypair",
  "SubnetId":"subnet-EXAMPLE",
  "VpcId":"vpc-EXAMPLE",
  "SolutionStackName":"64bit Amazon Linux 2018.03 v3.0.5 running Tomcat 8 Java
 8",
  "EBTrustRole":"CodeStarWorker-myproject-EBService",
  "EBInstanceProfile":"awscodestar-myproject-EBInstanceProfile-11111EXAMPLE"
}
```

6. In the stage edit pane, choose **Cancel**.

To create a GenerateChangeSet action in your new Prod stage

> **Note**
>
> After you add the new action but while you are still in edit mode, if you reopen the new action for editing, some of the fields might not be displayed. You might also see following: Stack *stack-name* does not exist

This error does not prevent you from saving the pipeline. However, to restore the missing fields, you must delete the new action and add it again. After you save and run the pipeline, the stack is recognized and the error does not reappear.

1. If your pipeline is not already displayed, from your AWS CodeStar project dashboard, choose **Pipeline Details** to open your pipeline in the console.
2. Choose **Edit**.
3. At the bottom of the diagram, choose **+ Add stage**.
4. Enter a stage name (for example, **Prod**), and then choose **+ Add action group**.
5. In **Action name**, enter a name (for example, **GenerateChangeSet**).
6. In **Action provider**, choose **AWS CloudFormation**.
7. In **Action mode**, choose **Create or replace a change set**.
8. In **Stack name**, enter a new name for the AWS CloudFormation stack that is to be created by this action. Start with a name that is identical to the Deploy stack name, and then add **-prod**:
   - Lambda projects: `awscodestar-<project_name>-lambda-prod`
   - Amazon EC2 and Elastic Beanstalk projects: `awscodestar-<project_name>-infrastructure-prod`

   Note

   The stack name must start with **awscodestar-<project_name>-** exactly, or the stack creation fails.
9. In **Change set name**, enter the same change set name as provided in the existing Deploy stage (for example, **pipeline-changeset**).
10. In **Input artifacts**, choose the build artifact.
11. In **Template**, enter the same change template name as provided in the existing Deploy stage (for example, **<project-ID>-BuildArtifact::template.yml**).
12. In **Template configuration**, enter the same change template configuration file name as provided in the Deploy stage (for example, **<project-ID>-BuildArtifact::template-configuration.json**).
13. In **Capabilities**, choose **CAPABILITY_NAMED_IAM**.

14. In **Role name**, choose the name of your project's AWS CloudFormation worker role.
15. Expand **Advanced**, and in **Parameters**, paste the parameters for your project. Include the `Stage` parameter, shown here in JSON format, for an Amazon EC2 project:

```
{

  "ProjectId":"MyProject",
  "InstanceType":"t2.micro",
  "WebAppInstanceProfile":"awscodestar-MyProject-WebAppInstanceProfile-
EXAMPLEY5VSFS",
  "ImageId":"ami-EXAMPLE1",
  "KeyPairName":"my-keypair",
  "SubnetId":"subnet-EXAMPLE",
  "VpcId":"vpc-EXAMPLE1",
  "Stage":"Prod"
}
```

> **Note**
>
> Make sure to paste all of the parameters for the project, not just new parameters or parameters you want to change.

16. Choose **Save**.
17. In the AWS CodePipeline pane, choose **Save pipeline change**, and then choose **Save change**.

> **Note**
>
> A message might display that notifies you of change-detection resources being deleted and added. Acknowledge the message and continue to the next step in this tutorial.

View your updated pipeline.

To create an ExecuteChangeSet action in your new Prod stage

1. If you are not already viewing your pipeline, from your AWS CodeStar project dashboard, choose **Pipeline Details** to open your pipeline in the console.
2. Choose **Edit**.

3. In your new Prod stage, after the new **GenerateChangeSet** action, choose **+ Add action group**.
4. In **Action name**, enter a name (for example, **`ExecuteChangeSet`**).
5. In **Action provider**, choose **AWS CloudFormation**.
6. In **Action mode**, choose **Execute a change set**.
7. In **Stack name**, enter the new name for the AWS CloudFormation stack that you entered in the GenerateChangeSet action (for example, **`awscodestar-<project-ID>-infrastructure-prod`**).
8. In **Change set name**, enter the same change set name used in the Deploy stage (for example, **`pipeline-changeset`**).
9. Choose **Done**.
10. In the AWS CodePipeline pane, choose **Save pipeline change**, and then choose **Save change**.

> **Note**
>
> A message might display that notifies you of change-detection resources being deleted and added. Acknowledge the message and continue to the next step in this tutorial.

View your updated pipeline.

To create a CodeDeploy Deploy action in your new Prod stage (Amazon EC2 projects only)

1. After the new actions in your Prod stage, choose **+ Action**.
2. In **Action name**, enter a name (for example, **`Deploy`**).
3. In **Action provider**, choose **AWS CodeDeploy**.
4. In **Application name**, choose the name of the CodeDeploy application for your project.
5. In **Deployment group**, choose the name of the new CodeDeploy deployment group you created in step 2.
6. In **Input artifacts**, choose the same build artifact used in the existing stage.
7. Choose **Done**.
8. In the AWS CodePipeline pane, choose **Save pipeline change**, and then choose **Save change**. View your updated pipeline.

Step 3: Add a Manual Approval Stage

As a best practice, add a manual approval stage in front of your new production stage.

1. In the upper left, choose **Edit**.
2. In your pipeline diagram, between the Deploy and Prod deployment stages, choose **+ Add stage**.
3. On **Edit stage**, enter a stage name (for example, **`Approval`**), and then choose **+ Add action group**.
4. In **Action name**, enter a name (for example, **`Approval`**).
5. In **Approval type**, choose **Manual approval**.
6. (Optional) Under **Configuration**, in **SNS Topic ARN**, choose the SNS topic that you have created and subscribed to.
7. Choose **Add Action**.
8. In the AWS CodePipeline pane, choose **Save pipeline change**, and then choose **Save change**. View your updated pipeline.
9. To submit your changes and start a pipeline build, choose **Release change**, and then choose **Release**.

Step 4: Push a Change and Monitor the AWS CloudFormation Stack Update

1. While your pipeline is running, you can use the steps here to follow the stack and endpoint creation for your new stage.
2. When the pipeline starts the Deploy stage, the AWS CloudFormation stack update starts. You can choose the AWS CloudFormation stage in your pipeline on your AWS CodeStar dashboard to see the stack update notification. To view stack creation details, in the console, choose your project from the **Events** list.
3. After successful completion of your pipeline, the resources are created in your AWS CloudFormation stack. In the AWS CloudFormation console, choose the infrastructure stack for your project. Stack names follow this format:

   - Lambda projects: `awscodestar-<project_name>-lambda-prod`
   - Amazon EC2 and Elastic Beanstalk projects: `awscodestar-<project_name>-infrastructure-prod`

In the **Resources** list in the AWS CloudFormation console, view the resource created for your project. In this example, the new Amazon EC2 instance appears in the **Resources** section.

4. Access the endpoint for your production stage:

   - For an Elastic Beanstalk project, open the new stack in the AWS CloudFormation console and expand **Resources**. Choose the Elastic Beanstalk application. The link opens in the Elastic Beanstalk console. Choose **Environments**. Choose the URL in **URL** to open the endpoint in a browser.
   - For a Lambda project, open the new stack in the AWS CloudFormation console and expand **Resources**. Choose the API Gateway resource. The link opens in the API Gateway console. Choose **Stages**. Choose the URL in **Invoke URL** to open the endpoint in a browser.
   - For an Amazon EC2 project, choose the new Amazon EC2 instance in your project resources list in the AWS CodeStar console. The link opens on the **Instance** page of the Amazon EC2 console. Choose the **Description** tab, copy the URL in **Public DNS (IPv4)**, and open the URL in a browser.

5. Verify that your change is deployed.

Securely Use SSM Parameters in an AWS CodeStar Project

Many customers store secrets, such as credentials, in Systems Manager Parameter Store parameters. Now you can securely use these parameters in an AWS CodeStar project. For example, you might want to use SSM parameters in your build spec for CodeBuild or when defining application resources in your toolchain stack (template.yml).

In order to use SSM parameters in a AWS CodeStar project, you must manually tag the parameters with the AWS CodeStar project ARN. You must also provide appropriate permissions to the AWS CodeStar toolchain worker role to access the parameters that you've tagged.

Before You Begin

- Create a new or identify an existing Systems Manager parameter that contains the information you want to access.
- Identify which AWS CodeStar project you want to use, or create a new project.
- Make a note of the CodeStar project ARN. It looks like this: `arn:aws:codestar:`*`region-id`*`:`*`account-id`*`:project/`*`project-id`*.

Tag a Parameter with the AWS CodeStar Project ARN

See Tagging Systems Manager Parameters for step by step instructions.

1. In **Key**, enter `awscodestar:projectArn`.
2. In **Value**, enter the project ARN from CodeStar: `arn:aws:codestar:`*`region-id`*`:`*`account-id`*`:project/`*`project-id`*.
3. Choose **Save**.

Now you can reference the SSM parameter in your template.yml file. If you want to use it with a toolchain worker role, you will need to grant additional permissions.

Grant Permissions to Use Tagged Parameters in your AWS CodeStar Project Toolchain

> **Note**
>
> These steps are applicable only to projects created after December 6, 2018 PDT.

1. Open the AWS CodeStar project dashboard for the project you want to use.
2. Click **Project** to view the list of created resources, and locate the toolchain worker role. It is an IAM resource with a name of the format: `role/CodeStarWorker-`*`project-id`*`-ToolChain`.
3. Click the ARN to open it in the IAM console.
4. Locate the ToolChainWorkerPolicy and expand it, if necessary.
5. Click **Edit Policy**.
6. Under `Action:` add the following line:

   ```
   ssm:GetParameter*
   ```

7. Click Review policy, then click Save changes.

For projects created before December 6, 2018 PDT, you will need to add the following permissions to the worker roles for each service.

```
        {
            "Action": [
                "ssm:GetParameter*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ssm:ResourceTag/awscodestar:projectArn": "arn:aws:codestar:region-
id:account-id:project/project-id"
                }
            }
        }
```

Shift Traffic for an AWS Lambda Project

AWS CodeDeploy supports function version deployments for AWS Lambda functions in your AWS CodeStar serverless projects. An AWS Lambda deployment shifts incoming traffic from an existing Lambda function to an updated Lambda function version. You might want to test an updated Lambda function by deploying a separate version and then rolling back the deployment to the first version if needed.

Use the steps in this section to modify your AWS CodeStar project template and update your CodeStarWorker roles IAM permissions. This task starts an automated response in AWS CloudFormation that creates aliased AWS Lambda functions and then instructs AWS CodeDeploy to shift traffic to an updated environment.

> **Note**
>
> Complete these steps only if you created your AWS CodeStar project before December 12, 2018.

AWS CodeDeploy has three deployment options that allow you to shift traffic to versions of your AWS Lambda function in your application:

- **Canary:** Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the

first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.

- **Linear:** Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment. Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- **All-at-once:** All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

Deployment Preference Type
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery10Minutes
Linear10PercentEvery1Minute
Linear10PercentEvery2Minutes
Linear10PercentEvery3Minutes
AllAtOnce

For more information about AWS CodeDeploy deployments on an AWS Lambda compute platform, see Deployments on an AWS Lambda Compute Platform.

For more information about AWS SAM, see AWS Serverless Application Model (AWS SAM) on GitHub.

Prerequisites:

When you create a serverless project, select any template with the Lambda compute platform. You must be signed in as an administrator to perform steps 4-6.

Step 1: Modify the SAM template to add AWS Lambda version deployment parameters

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Create a project or choose an existing project with a `template.yml` file, and then open the **Code** page. In the top level of your repository, note the location of the SAM template named `template.yml` to be modified.
3. Open the `template.yml` file in your IDE or local repository. Copy the following text to add a `Globals` section to the file. The sample text in this tutorial chooses the `Canary10Percent5Minutes` option.

```
Globals:
  Function:
    AutoPublishAlias: live
    DeploymentPreference:
      Enabled: true
      Type: Canary10Percent5Minutes
```

This example shows a modified template after the `Globals` section has been added:

```
AWSTemplateFormatVersion: 2010-09-09
Transform:
- AWS::Serverless-2016-10-31
- AWS::CodeStar

Parameters:
  ProjectId:
    Type: String
    Description: CodeStar projectId used to associate new resources to team members

Globals:
  Function:
    AutoPublishAlias: live
    DeploymentPreference:
      Enabled: true
      Type: Canary10Percent5Minutes

Resources:
  HelloWorld:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.6
      Role:
        Fn::ImportValue:
          !Join ['-', [!Ref 'ProjectId', !Ref 'AWS::Region', 'LambdaTrustRole']]
      Events:
```

For more information, see the Globals Section reference guide for SAM templates.

Step 2: Edit the AWS CloudFormation role to add permissions

1. Sign in to the AWS Management Console and open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.

   > **Note**
   >
   > You must sign in to the AWS Management Console using credentials associated with the IAM user you created or identified in Setting Up AWS CodeStar. This user must have the AWS managed policy named **`AWSCodeStarFullAccess`** attached.

2. Choose your existing serverless project and then open the **Project resources** page.
3. Under **Resources**, choose the IAM role created for the CodeStarWorker/AWS CloudFormation role. The role opens in the IAM console.
4. On the **Permissions** tab, in **Inline Policies**, in the row for your service role policy, choose **Edit Policy**. Choose the **JSON** tab to edit the policy in JSON format.

   > **Note**
   >
   > Your service role is named `CodeStarWorkerCloudFormationRolePolicy`.

5. In the **JSON** field, add the following policy statements within the `Statement` element. Replace the *`region`* and *`id`* placeholders with your region and account ID.

```
{
  "Action": [
    "s3:GetObject",
    "s3:GetObjectVersion",
    "s3:GetBucketVersioning"
  ],
  "Resource": "*",
  "Effect": "Allow"
},
{
  "Action": [
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::codepipeline*"
```

```
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "lambda:*"
    ],
    "Resource": [
      "arn:aws:lambda:region:id:function:*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "apigateway:*"
    ],
    "Resource": [
      "arn:aws:apigateway:region::*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:GetRole",
      "iam:CreateRole",
      "iam:DeleteRole",
      "iam:PutRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::id:role/*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:AttachRolePolicy",
      "iam:DeleteRolePolicy",
      "iam:DetachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::id:role/*"
    ],
    "Effect": "Allow"
  },
```

```
{
  "Action": [
    "iam:PassRole"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "codedeploy:CreateApplication",
    "codedeploy:DeleteApplication",
    "codedeploy:RegisterApplicationRevision"
  ],
  "Resource": [
    "arn:aws:codedeploy:region:id:application:*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "codedeploy:CreateDeploymentGroup",
    "codedeploy:CreateDeployment",
    "codedeploy:DeleteDeploymentGroup",
    "codedeploy:GetDeployment"
  ],
  "Resource": [
    "arn:aws:codedeploy:region:id:deploymentgroup:*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "codedeploy:GetDeploymentConfig"
  ],
  "Resource": [
    "arn:aws:codedeploy:region:id:deploymentconfig:*"
  ],
  "Effect": "Allow"
}
```

6. Choose **Review policy** to ensure the policy contains no errors. When the policy is error-free, choose **Save changes**.

Step 3: Commit and push your template change to start the AWS Lambda version shift

1. Commit and push the changes in the `template.yml` file that you saved in step 1.

   > **Note**
   >
   > This starts your pipeline. If you commit the changes before you update IAM permissions, your pipeline starts and the AWS CloudFormation stack update encounters errors that roll back the stack update. If this happens, restart your pipeline after permissions have been corrected.

2. The AWS CloudFormation stack update starts when the pipeline for your project starts the **Deploy** stage. To see the stack update notification when the deployment starts, on your AWS CodeStar dashboard, select the AWS CloudFormation stage in your pipeline.

   During stack update, AWS CloudFormation automatically updates the project resources as follows:

   - AWS CloudFormation processes the `template.yml` file by creating aliased Lambda functions, event hooks, and resources.
   - AWS CloudFormation calls Lambda to create the new version of the function.
   - AWS CloudFormation creates an AppSpec file and calls AWS CodeDeploy to shift the traffic.

   For more information about publishing aliased Lambda functions in SAM, see the AWS Serverless Application Model (SAM) template reference. For more information about event hooks and resources in the AWS CodeDeploy AppSpec file, see AppSpec 'resources' Section (AWS Lambda Deployments Only) and AppSpec 'hooks' Section for an AWS Lambda Deployment.

3. After successful completion of your pipeline, the resources are created in your AWS CloudFormation stack. On the **Project** page, in the **Project Resources** list, view the AWS CodeDeploy application, the AWS CodeDeploy deployment group, and the AWS CodeDeploy service role resources created for your project.

4. To create a new version, make a change to the Lambda function in your repository. The new deployment starts and shifts traffic according to the deployment type indicated in the SAM template. To view the status of the traffic that is being shifted to the new version, on the **Project** page, in the **Project Resources** list, choose the link to the AWS CodeDeploy deployment.

5. To view details about each revision, under **Revisions**, choose the link to the AWS CodeDeploy deployment group.
6. In your local working directory, you can make changes to your AWS Lambda function and commit the change to your project repository. AWS CloudFormation supports AWS CodeDeploy in managing the next revision in the same way. For more information about redeploying, stopping, or rolling back a Lambda deployment, see Deployments on an AWS Lambda Compute Platform.

Transition your AWS CodeStar Project to Production

After you have created your application using an AWS CodeStar project and seen what AWS CodeStar provides, you might want to transition your project to production use. One way to do this is to replicate your application's AWS resources outside of AWS CodeStar. You will still need a repository, a build project, a pipeline, and a deployment, but rather than having AWS CodeStar create them for you, you will recreate them using AWS CloudFormation.

> **Note**
>
> It can be helpful to create or view a similar project using one of the AWS CodeStar quick starts first and use that as a template for your own project to make sure you include the resources and policies you need.

An AWS CodeStar project is a combination of source code and the resources created to deploy the code. The collection of resources that help you build, release, and deploy your code are called toolchain resources. At project creation, an AWS CloudFormation template provisions your toolchain resources in a continuous integration/continuous deployment (CI/CD) pipeline.

When you use the console to create a project, the toolchain template is created for you. When you use the AWS CLI to create a project, you create the toolchain template that creates your toolchain resources.

A complete toolchain requires the following recommended resources:

1. A CodeCommit or GitHub repository that contains your source code.
2. A CodePipeline pipeline that is configured to listen to changes to your repository.

a. When you use AWS CodeBuild to run unit or integration tests, we recommend that you add a build stage to your pipeline to create build artifacts.

b. We recommend that you add a deployment stage to your pipeline that uses CodeDeploy or AWS CloudFormation to deploy your build artifact and source code to your runtime infrastructure.

> **Note**
>
> Because CodePipeline requires at least two stages in a pipeline, and the first stage must be the source stage, add a build or deploy stage as the second stage.

Topics

- Create a GitHub Repository

Create a GitHub Repository

You create a GitHub repository by defining it in your toolchain template. Make sure that you have already created a location for a ZIP file containing your source code, so the code can be uploaded to the repository. Also, you must have already created a personal access token in GitHub so that AWS can connect to GitHub on your behalf. In addition to the personal access token for GitHub, you also must have `s3.GetObject` permission for the `Code` object you pass in.

To specify a public GitHub repository, add code like the following to your toolchain template in AWS CloudFormation.

```
GitHubRepo:
Condition: CreateGitHubRepo
Description: GitHub repository for application source code
Properties:
  Code:
    S3:
      Bucket: MyCodeS3Bucket
      Key: MyCodeS3BucketKey
  EnableIssues: true
  IsPrivate: false
  RepositoryAccessToken: MyGitHubPersonalAccessToken
  RepositoryDescription: MyAppCodeRepository
```

```
      RepositoryName: MyAppSource
      RepositoryOwner: MyGitHubUserName
    Type: AWS::CodeStar::GitHubRepository
```

This code specifies the following information:

- The location of the code you want to include, which must be an Amazon S3 bucket.
- Whether you want to enable issues on the GitHub repository.
- Whether the GitHub repository is private.
- The GitHub personal access token you created.
- A description, name, and owner for the repository you are creating.

For complete details about what information to specify, see AWS::CodeStar::GitHubRepository in the *AWS CloudFormation User Guide*.

Working with Project Tags in AWS CodeStar

You can associate tags with projects in AWS CodeStar. Tags can help you manage your projects. For example, you might add a tag with a key of `Release` and a value of `Beta` to any project your organization is working on for a beta release.

Add a Tag to a Project

1. With the project open in the AWS CodeStar console, in the side navigation pane, choose **Settings**.
2. In **Tags**, choose **Edit**.
3. In **Key**, enter the tag's name. In **Value**, enter the tag's value.
4. **Optional:** Choose **Add tag** to add more tags.
5. Once you're done adding tags, choose **Save**.

Remove a Tag from a Project

1. With the project open in the AWS CodeStar console, in the side navigation pane, choose **Settings**.

2. In **Tags**, choose **Edit**.
3. In **Tags**, find the tag you want to remove and choose **Remove tag**.
4. Choose **Save**.

Get a List of Tags for a Project

Use the AWS CLI to run the AWS CodeStar **list-tags-for-project** command, specifying the name of the project:

```
aws codestar list-tags-for-project --id my-first-projec
```

If successful, a list of tags appears in the output, similar to the following:

```
{
  "tags": {
    "Release": "Beta"
  }
}
```

Delete an AWS CodeStar Project

If you no longer need a project, you can delete it and its resources so that you do not incur any further charges in AWS. When you delete a project, all team members are removed from that project. Their project roles are removed from their IAM users, but their user profiles in AWS CodeStar are not changed. You can use the AWS CodeStar console or AWS CLI to delete a project. Deleting a project requires the AWS CodeStar service role, `aws-codestar-service-role`, which must be unmodified and assumable by AWS CodeStar.

Important

Deleting a project in AWS CodeStar cannot be undone. By default all AWS resources for the project are deleted in your AWS account, including:

- The CodeCommit repository for the project along with anything stored in that repository.
- The AWS CodeStar project roles and the associated IAM policies configured for the project and its resources.
- Any Amazon EC2 instances created for the project.

- The deployment application and associated resources, such as:
  - A CodeDeploy application and associated deployment groups.
  - An AWS Lambda function and associated API Gateway APIs.
  - An AWS Elastic Beanstalk application and associated environment.
- The continuous deployment pipeline for the project in CodePipeline.
- The AWS CloudFormation stacks associated with the project.
- Any AWS Cloud9 development environments created with the AWS CodeStar console. All uncommitted code changes in the environments are lost.

To delete all project resources along with the project, select the **Delete resources** check box. If you clear this option, the project is deleted in AWS CodeStar, and the project roles that enabled access to those resources are deleted in IAM, but all other resources are retained. You might continue to incur charges for these resources in AWS. If you decide you no longer want one or more of these resources, you must manually delete them. For more information, see Project deletion: An AWS CodeStar project was deleted, but resources still exist.

If you decide to keep resources when you delete a project, as a best practice, copy the list of resources from the project details page. This way, you have a record of all resources that you have kept, even though the project no longer exists.

Topics

- Delete a Project in AWS CodeStar (Console)
- Delete a Project in AWS CodeStar (AWS CLI)

Delete a Project in AWS CodeStar (Console)

You can use the AWS CodeStar console to delete a project.

To delete a project in AWS CodeStar

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** in the navigation pane.
3. Select the project you want to delete and choose **Delete**.

Or, open the project and choose **Settings** from the navigation pane on the left side of the console. On the project details page, choose **Delete project**.

4. In the **Delete confirmation page**, enter *delete*. Keep **Delete resources** selected if you wish to delete project resources. Choose **Delete**.

   Deleting a project can take several minutes. After it's deleted, the project no longer appears in the list of projects in the AWS CodeStar console.

   > **Important**
   >
   > If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), those resources are not deleted, even if you select the check box.
   > Your project cannot be deleted if any AWS CodeStar managed policies have been manually attached to roles that are not IAM users. If you have attached your project's managed policies to a federated user's role, you must detach the policy before you can delete the project. For more information, see ???.

Delete a Project in AWS CodeStar (AWS CLI)

You can use the AWS CLI to delete a project.

To delete a project in AWS CodeStar

1. At a terminal (Linux, macOS, or Unix) or command prompt (Windows), run the **delete-project** command, including the name of the project. For example, to delete a project with the ID *my-2nd-project*:

   ```
   aws codestar delete-project --id my-2nd-project
   ```

   This command returns output similar to the following:

   ```
   {
       "projectArn":"arn:aws:codestar:us-east-2:111111111111:project/my-2nd-project"
   }
   ```

Projects are not deleted immediately.

2. Run the **describe-project** command, including the name of the project. For example, to check the status of a project with the ID *my-2nd-project*:

```
aws codestar describe-project --id my-2nd-project
```

if the project isn't deleted yet, this command returns output similar to the following:

```
{
    "name": "my project",
    "id": "my-2nd-project",
    "arn": "arn:aws:codestar:us-west-2:123456789012:project/my-2nd-project",
    "description": "My second CodeStar project.",
    "createdTimeStamp": 1572547510.128,
    "status": {
        "state": "CreateComplete"
    }
}
```

If the project is deleted, this command returns output similar to the following:

```
   An error occurred (ProjectNotFoundException) when calling the DescribeProject
operation: The project ID was not found: my-2nd-project. Make sure that the
project ID is correct and then try again.
```

3. Run the **list-projects** command and verify that the deleted project no longer appears in the list of projects associated with your AWS account.

```
aws codestar list-projects
```

Working with AWS CodeStar Teams

After you create a development project, grant access to others so you can work together. In AWS CodeStar, each project has a *project team*. A user can belong to multiple AWS CodeStar projects and have different AWS CodeStar roles (and thus, different permissions) in each. In the AWS CodeStar console, users see all projects associated with your AWS account, but they can view and work only on those projects in which they are team members.

Team members can choose a friendly name for themselves. They can also add an email address so other team members can contact them. Team members who are not owners cannot change their AWS CodeStar role for the project.

Each project in AWS CodeStar has three roles:

Roles and Permissions in an AWS CodeStar Project

Role Name	View Project Dashboard and Status	Add/Remove/ Access Project Resources	Add/Remove Team Members	Delete Project
Owner	x	x	x	x
Contributor	x	x		
Viewer	x			

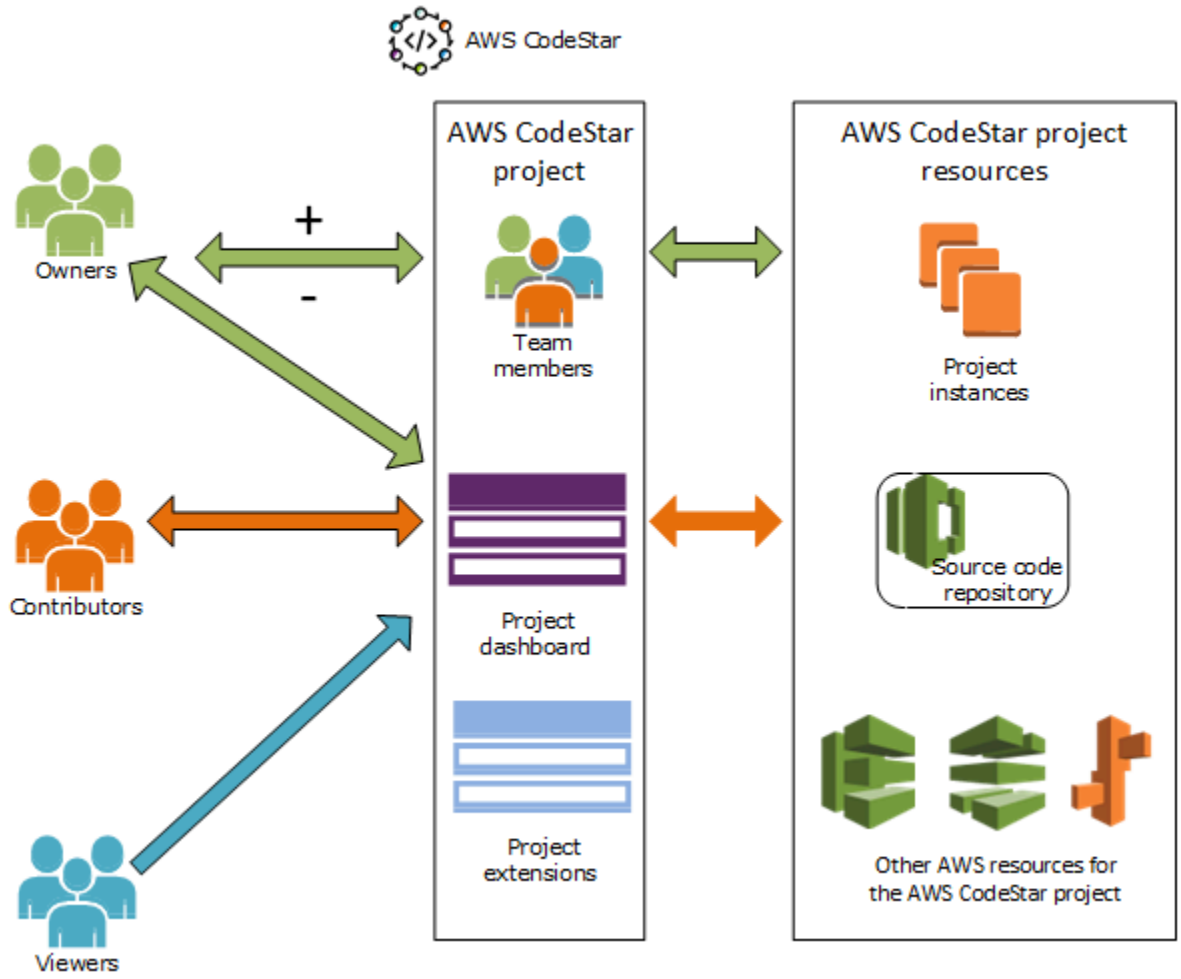
- **Owner**: Can add and remove other team members, contribute code to a project repository if the code is stored in CodeCommit, grant or deny other team members remote access to any Amazon EC2 instances running Linux associated with the project, configure the project dashboard, and delete the project.
- **Contributor**: Can add and remove dashboard resources such as a JIRA tile, contribute code to the project repository if the code is stored in CodeCommit, and interact fully with the dashboard. Cannot add or remove team members, grant or deny remote access to resources, or delete the project. This is the role you should choose for most team members.
- **Viewer**: Can view the project dashboard, the code if is stored in CodeCommit, and, on the dashboard tiles, the state of the project and its resources.

Important

If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), access to those resources is controlled by the resource provider, not AWS CodeStar. For more information, see the resource provider's documentation.
Anyone who has access to an AWS CodeStar project can use the AWS CodeStar console to access resources that are outside of AWS but related to the project.
AWS CodeStar does not automatically allow project team members to participate in any related AWS Cloud9 development environments for a project. To allow a team member to participate in a shared environment, see Share an AWS Cloud9 Environment with a Project Team Member.

An IAM policy is associated with each project role. This policy is customized for your project to reflect its resources. For more information about these policies, see AWS CodeStar Identity-Based Policy Examples.

The following diagram shows the relationship between each role and an AWS CodeStar project.



Topics

- Add Team Members to an AWS CodeStar Project
- Manage Permissions for AWS CodeStar Team Members
- Remove Team Members from an AWS CodeStar Project

Add Team Members to an AWS CodeStar Project

If you have the owner role in an AWS CodeStar project or have the `AWSCodeStarFullAccess` policy applied to your IAM user, you can add other IAM users to the project team. This is a simple process that applies an AWS CodeStar role (owner, contributor, or viewer) to the user. These roles are per-project and customized. For example, a contributor team member in project A might have permissions to resources that are different from those of a contributor team member in project B. A team member can have only one role in a project. After you've added a team member, he or she can interact immediately with your project at the level defined by the role.

Benefits of AWS CodeStar roles and team membership include:

- You do not have to manually configure permissions in IAM for your team members.
- You can easily change a team member's level of access to a project.
- Users can access projects in the AWS CodeStar console only if they are team members.
- User access to a project is defined by role.

For more information about teams and AWS CodeStar roles, see Working with AWS CodeStar Teams and Working with Your AWS CodeStar User Profile .

To add a team member to a project, you must either have the AWS CodeStar owner role for the project or the `AWSCodeStarFullAccess` policy.

Important

Adding a team member does not affect that member's access to resources that are outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA). Those access permissions are controlled by the resource provider, not AWS CodeStar. For more information, see the resource provider's documentation.

Anyone who has access to an AWS CodeStar project can use the AWS CodeStar console to access resources that are outside of AWS but related to that project.

Adding a team member to a project does not automatically allow that member to participate in any related AWS Cloud9 development environments for the project. To allow a team member to participate in a shared environment, see Share an AWS Cloud9 Environment with a Project Team Member.

Granting federated user access to a project involves manually attaching the AWS CodeStar owner, contributor, or viewer managed policy to the role assumed by the federated user. For more information, see Federated User Access to AWS CodeStar.

Topics

- Add a Team Member (Console)
- Add and View Team Members (AWS CLI)

Add a Team Member (Console)

You can use the AWS CodeStar console to add a team member to your project. If an IAM user already exists for the person you want to add, you can add the IAM user. Otherwise, you can create an IAM user for that person when you add them to your project.

To add a team member to an AWS CodeStar project (console)

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose your project.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose **Add team member**.
5. In **Choose user**, do one of the following:

   - If an IAM user already exists for the person you want to add, choose the IAM user from the list.

     > **Note**
     >
     > Users who have already been added to another AWS CodeStar project appear in the **Existing AWS CodeStar users** list.

     In **Project role**, choose the AWS CodeStar role (Owner, Contributor, or Viewer) for this user. This is an AWS CodeStar project-level role that can only be changed by an owner of the project. When applied to an IAM user, the role provides all permissions required to access AWS CodeStar project resources. It applies policies required for creating and managing Git credentials for code stored in CodeCommit in IAM or uploading Amazon EC2 SSH keys for the user in IAM.

     > **Important**
     >
     > You cannot provide or change the display name or email information for an IAM user unless you are signed in to the console as that user. For more information, see Manage Display Information for Your AWS CodeStar User Profile .

Choose **Add team member**.

- If an IAM user does not exist for the person you want to add to the project, choose **Create new IAM user**. You will be redirected to the IAM console where you can create a new IAM user, see Creating IAM users in the *IAM user guide* for more information. After you create your IAM user, return to the AWS CodeStar console, refresh the list of users, and choose the IAM user you created from the dropdown list. Enter the AWS CodeStardisplay name, email address, and project role you want to apply to this new user, and then choose **Add team member**.

> **Note**
>
> For ease of management, at least one user should be assigned the Owner role for the project.

6. Send the new team member the following information:

- Connection information for your AWS CodeStar project.
- If the source code is stored in CodeCommit, instructions for setting up access with Git credentials to the CodeCommit repository from their local computers.
- Information about how the user can manage their display name, email address, and public Amazon EC2 SSH key, as described in Working with Your AWS CodeStar User Profile .
- One-time password and connection information, if the user is new to AWS and you created an IAM user for that person. The password expires the first time the user signs in. The user must choose a new password.

Add and View Team Members (AWS CLI)

You can use the AWS CLI to add team members to your project team. You can also view information about all of the team members in your project.

To add a team member

1. Open a terminal or command window.
2. Run the **associate-team-member** command with the `--project-id`, `-user-arn`, and `--project-role` parameters. You can also specify whether the user has remote access to

project instances by including the `--remote-access-allowed` or `--no-remote-access-allowed` parameters. For example:

```
aws codestar associate-team-member --project-id my-first-projec --user-arn
 arn:aws:iam:111111111111:user/Jane_Doe --project-role Contributor --remote-access-
allowed
```

This command returns no output.

To view all team members (AWS CLI)

1. Open a terminal or command window.
2. Run the **list-team-members** command with the `--project-id` parameter. For example:

```
aws codestar list-team-members --project-id my-first-projec
```

This command returns output similar to the following:

```
{
    "teamMembers":[

 {"projectRole":"Owner","remoteAccessAllowed":true,"userArn":"arn:aws:iam::111111111111:use
Mary_Major"},

 {"projectRole":"Contributor","remoteAccessAllowed":true,"userArn":"arn:aws:iam::1111111111
Jane_Doe"},

 {"projectRole":"Contributor","remoteAccessAllowed":true,"userArn":"arn:aws:iam::1111111111
John_Doe"},

 {"projectRole":"Viewer","remoteAccessAllowed":false,"userArn":"arn:aws:iam::111111111111:u
John_Stiles"}
 ]
}
```

Manage Permissions for AWS CodeStar Team Members

You change permissions for team members by changing their AWS CodeStar role. Each team member can be assigned to only one role in an AWS CodeStar project, but many users can be assigned to the same role. You can use the AWS CodeStar console or AWS CLI to manage permissions.

Important

To change a role for a team member, you must either have the AWS CodeStar owner role for that project or have the `AWSCodeStarFullAccess` policy applied.
Changing a team member's permissions does not affect that team member's access to any resources that are outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA). Those access permissions are controlled by the resource provider, not AWS CodeStar. For more information, see the resource provider's documentation.
Anyone who has access to an AWS CodeStar project may be able to use the AWS CodeStar console to access resources that are outside of AWS but are related to that project.
Changing a team member's role for a project does not automatically allow or prevent that member from participating in any AWS Cloud9 development environments for the project. To allow or prevent a team member from participating in a shared environment, see Share an AWS Cloud9 Environment with a Project Team Member.

You can also grant permissions for users to remotely access any Amazon EC2 Linux instances associated with the project. After you grant this permission, the user must upload an SSH public key that is associated with their AWS CodeStar user profile across all team projects. To successfully connect to the Linux instances, the user must have SSH configured and the private key on the local computer.

Topics

- Manage Team Permissions (Console)
- Manage Team Permissions (AWS CLI)

Manage Team Permissions (Console)

You can use the AWS CodeStar console to manage the roles of team members. You can also manage whether team members have remote access to the Amazon EC2 instances associated with your project.

To change the role of a team member

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose your project.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose the team member and choose **Edit**.
5. In **Project role**, choose the AWS CodeStar role (owner, contributor, or viewer) you want to grant this user.

   For more information about AWS CodeStar roles and their permissions, see Working with AWS CodeStar Teams.

   Choose **Edit team member**.

To grant a team member remote access permissions to Amazon EC2 instances

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose your project.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose the team member and choose **Edit**.
5. Select **Allow SSH access to project instances**, and then choose **Edit team member**.
6. (Optional) Notify the team members that they should upload an SSH public key for their AWS CodeStar users, if they have not already done so. For more information, see Add a Public Key to Your AWS CodeStar User Profile .

Manage Team Permissions (AWS CLI)

You can use the AWS CLI to manage the project role assigned to a team member. You can use the same AWS CLI commands to manage whether that team member has remote access to Amazon EC2 instances associated with your project.

To manage the permissions for a team member

1. Open a terminal or command window.
2. Run the **update-team-member** command with the `--project-id`, `-user-arn`, and `--project-role` parameters. You can also specify whether the user has remote access to project instances by including the `--remote-access-allowed` or `--no-remote-access-allowed` parameters. For example, to update the project role of an IAM user named John_Doe and change his permissions to viewer with no remote access to project Amazon EC2 instances:

```
aws codestar update-team-member --project-id my-first-projec --user-arn
 arn:aws:iam:111111111111:user/John_Doe --project-role Viewer --no-remote-access-
allowed
```

   This command returns output similar to the following:

```
{
 "projectRole":"Viewer",
 "remoteAccessAllowed":false,
 "userArn":"arn:aws:iam::111111111111:user/John_Doe"
}
```

Remove Team Members from an AWS CodeStar Project

After you remove a user from an AWS CodeStar project, the user still appears in the commit history for the project repository, but no longer has access to the CodeCommit repository or any other project resources, such as the project pipeline. (The exception to this rule is an IAM user who has other policies that grant access to those resources.) The user can't access the project dashboard, and the project no longer appears in the list of projects that user sees on the AWS CodeStar dashboard. You can use the AWS CodeStar console or AWS CLI to remove team members from your project team.

Important

Although removing a team member from a project denies remote access to project Amazon EC2 instances, it does not close any of the user's active SSH sessions.
Removing a team member does not affect that team member's access to any resources that are outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA). Those

access permissions are controlled by the resource provider, not AWS CodeStar. For more information, see the resource provider's documentation.
Removing a team member from a project does not automatically delete that team member's related AWS Cloud9 development environments or prevent that member from participating in any related AWS Cloud9 development environments they have been invited to. To delete a development environment, see Delete an AWS Cloud9 Environment from a Project. To prevent a team member from participating in a shared environment, see Share an AWS Cloud9 Environment with a Project Team Member.

To remove a team member from a project, you must have the AWS CodeStar owner role for that project or have the `AWSCodeStarFullAccess` policy applied to your account.

Topics

- Remove Team Members (Console)
- Remove Team Members (AWS CLI)

Remove Team Members (Console)

You can use the AWS CodeStar console to remove team members from your project team.

To remove a team member from a project

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose your project.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose the team member and choose **Remove**.

Remove Team Members (AWS CLI)

You can use the AWS CLI to remove team members from your project team.

To remove a team member

1. Open a terminal or command window.
2. Run the **disassociate-team-member** command with the `--project-id` and `-user-arn`. For example:

```
aws codestar disassociate-team-member --project-id my-first-projec --user-arn
 arn:aws:iam:111111111111:user/John_Doe
```

This command returns output similar to the following:

```
{
    "projectId": "my-first-projec",
    "userArn": "arn:aws:iam::111111111111:user/John_Doe"
}
```

Working with Your AWS CodeStar User Profile

Your AWS CodeStar user profile is associated with your IAM user. This profile contains a display name and email address that is used in all AWS CodeStar projects you belong to. You can upload an SSH public key to be associated with your profile. This public key is part of the SSH public-private key pair you use when you connect to Amazon EC2 instances associated with AWS CodeStar projects you belong to.

> **Note**
>
> The information in these topics covers only your AWS CodeStar user profile. If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), those resource providers might use their own user profiles, which might have different settings. For more information, see the resource provider's documentation.

Topics

- Manage Display Information for Your AWS CodeStar User Profile
- Add a Public Key to Your AWS CodeStar User Profile

Manage Display Information for Your AWS CodeStar User Profile

You can use the AWS CodeStar console or AWS CLI to change the display name and email address in your user profile. A user profile is not project-specific. It is associated with your IAM user, and is applied across the AWS CodeStar projects you belong to in an AWS Region. If you belong to projects in more than one AWS Region, you have separate user profiles.

You can only manage your own user profile in the AWS CodeStar console. If you have the `AWSCodeStarFullAccess` policy, you can use the AWS CLI to view and manage other profiles.

> **Note**
>
> The information in this topic covers only your AWS CodeStar user profile. If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA),

those resource providers might use their own user profiles, which might have different settings. For more information, see the resource provider's documentation.

Topics

- Manage Your User Profile (Console)
- Manage User Profiles (AWS CLI)

Manage Your User Profile (Console)

You can manage your user profile in the AWS CodeStar console by navigating to any project where you are a team member and changing your profile information. Because user profiles are user-specific, not project-specific, your user profile changes appear in every project in an AWS Region where you are a team member.

> **Important**
>
> To use the console to change the display information for a user, you must be signed in as that IAM user. No other user, even those with the AWS CodeStar owner role for a project or with the `AWSCodeStarFullAccess` policy applied, can change your display information.

To change your display information in all projects in an AWS region

1. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
2. Choose **Projects** from the navigation pane and choose a project where you are a team member.
3. In the side navigation pane for the project, choose **Team**.
4. On the **Team members** page, choose the IAM user, then choose **Edit**.
5. Edit the display name, the email address, or both, and then choose **Edit team member**.

> **Note**
>
> A display name and email address are required. For more information, see Limits in AWS CodeStar.

Manage User Profiles (AWS CLI)

You can use the AWS CLI to create and manage your user profile in AWS CodeStar. You can also use the AWS CLI to view your user profile information, and to view all user profiles configured for your AWS account in an AWS Region.

Make sure that your AWS profile is configured for the region where you want to create, manage, or view user profiles.

To create a user profile

1. Open a terminal or command window.
2. Run the **create-user-profile** command with the `user-arn`, `display-name`, and `email-address` parameters. For example:

```
aws codestar create-user-profile --user-arn arn:aws:iam:111111111111:user/
John_Stiles --display-name "John Stiles" --email-address "john_stiles@example.com"
```

This command returns output similar to the following:

```
{
 "createdTimestamp":1.491439687681E9,"
 displayName":"John Stiles",
 "emailAddress":"john.stiles@example.com",
 "lastModifiedTimestamp":1.491439687681E9,
 "userArn":"arn:aws:iam::111111111111:user/Jane_Doe"
}
```

To view your display information

1. Open a terminal or command window.
2. Run the **describe-user-profile** command with the `user-arn` parameter. For example:

```
aws codestar describe-user-profile --user-arn arn:aws:iam:111111111111:user/
Mary_Major
```

This command returns output similar to the following:

```
{
```

```
 "createdTimestamp":1.490634364532E9,
 "displayName":"Mary Major",
 "emailAddress":"mary.major@example.com",
 "lastModifiedTimestamp":1.491001935261E9,
 "sshPublicKey":"EXAMPLE=",
 "userArn":"arn:aws:iam::111111111111:user/Mary_Major"
}
```

To change your display information

1. Open a terminal or command window.
2. Run the **update-user-profile** command with the `user-arn` parameter and the profile parameters you want to change, such as `display-name` or `email-address`. For example, if a user with the display name Jane Doe wants to change her display name to Jane Mary Doe:

```
aws codestar update-user-profile --user-arn arn:aws:iam:111111111111:user/Jane_Doe
 --display-name "Jane Mary Doe"
```

This command returns output similar to the following:

```
{
 "createdTimestamp":1.491439687681E9,
 "displayName":"Jane Mary Doe",
 "emailAddress":"jane.doe@example.com",
 "lastModifiedTimestamp":1.491442730598E9,
 "sshPublicKey":"EXAMPLE1",
 "userArn":"arn:aws:iam::111111111111:user/Jane_Doe"
}
```

To list all user profiles in an AWS region in your AWS account

1. Open a terminal or command window.
2. Run the **aws codestar list-user-profiles** command. For example:

```
aws codestar list-user-profiles
```

This command returns output similar to the following:

```
{
  "userProfiles":[
 {
  "displayName":"Jane Doe",
  "emailAddress":"jane.doe@example.com",
  "sshPublicKey":"EXAMPLE1",
  "userArn":"arn:aws:iam::111111111111:user/Jane_Doe"
 },
 {
  "displayName":"John Doe",
  "emailAddress":"john.doe@example.com",
  "sshPublicKey":"EXAMPLE2",
  "userArn":"arn:aws:iam::111111111111:user/John_Doe"
 },
 {
  "displayName":"Mary Major",
  "emailAddress":"mary.major@example.com",
  "sshPublicKey":"EXAMPLE=",
  "userArn":"arn:aws:iam::111111111111:user/Mary_Major"
 },
 {
  "displayName":"John Stiles",
  "emailAddress":"john.stiles@example.com",
  "sshPublicKey":"",
  "userArn":"arn:aws:iam::111111111111:user/John_Stiles"
 }
  ]
}
```

Add a Public Key to Your AWS CodeStar User Profile

You can upload a public SSH key as part of the public-private key pair you create and manage. You use this SSH public-private key pair to access Amazon EC2 instances running Linux. If a project owner has granted you remote access permission, you can access only those instances associated with the project. You can use the AWS CodeStar console or AWS CLI to manage your public key.

Important

An AWS CodeStar project owner can grant project owners, contributors, and viewers SSH access to Amazon EC2 instances for the project, but only the individual (owner, contributor,

or viewer) can set the SSH key. To do this, the user must be signed in as the individual owner, contributor, or viewer.
AWS CodeStar does not manage SSH keys for AWS Cloud9 environments.

Topics

- Manage Your Public Key (Console)
- Manage Your Public Key (AWS CLI)
- Connect to Amazon EC2 Instance with Your Private Key

Manage Your Public Key (Console)

Although you cannot generate a public-private key pair in the console, you can create one locally and then add or manage it as part of your user profile through the AWS CodeStar console.

To manage your public SSH key

1. From a terminal or Bash emulator window, run the **ssh-keygen** command to generate an SSH public-private key pair on your local computer. You can generate a key in any format allowed by Amazon EC2. For information about acceptable formats, see Importing Your Own Public Key to Amazon EC2. Ideally, generate a key that is SSH-2 RSA, in OpenSSH format, and contains 2048 bits. The public key is stored in a file with the .pub extension.
2. Open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.

   Choose a project where you are a team member.
3. In the navigation pane, choose **Team**.
4. On the **Team members** page, find the name of your IAM user, and then choose **Edit**.
5. On the **Edit team member** page, under **Remote access**, enable **Allow SSH access to project instances**.
6. In the **SSH Public Key** box, paste the public key, and then choose **Edit team member**.

> **Note**
>
> You can change your public key by deleting the old key in this field and pasting in a new one. You can delete a public key by deleting the contents of this field, and then choosing **Edit team member**.

When you change or delete a public key, you are changing your user profile. It is not a per-project change. Because your key is associated with your profile, it changes (or is deleted) in all projects where you have been granted remote access.

Deleting your public key removes your access to Amazon EC2 instances running Linux in all projects where you were granted remote access. However, it does not close any open SSH sessions using that key. Make sure that you close any open sessions.

Manage Your Public Key (AWS CLI)

You can use the AWS CLI to manage your SSH public key as part of your user profile.

To manage your public key

1. From a terminal or Bash emulator window, run the **ssh-keygen** command to generate an SSH public-private key pair on your local computer. You can generate a key in any format allowed by Amazon EC2. For information about acceptable formats, see Importing Your Own Public Key to Amazon EC2. Ideally, generate a key that is SSH-2 RSA, in OpenSSH format, and contains 2048 bits. The public key is stored in a file with the .pub extension.
2. To add or change your SSH public key in your AWS CodeStar user profile, run the **update-user-profile** command with the `--ssh-public-key` parameter. For example:

```
aws codestar update-user-profile --user-arn arn:aws:iam:111111111111:user/Jane_Doe
 --ssh-key-id EXAMPLE1
```

This command returns output similar to the following:

```
{
 "createdTimestamp":1.491439687681E9,
 "displayName":"Jane Doe",
```

```
 "emailAddress":"jane.doe@example.com",
 "lastModifiedTimestamp":1.491442730598E9,
 "sshPublicKey":"EXAMPLE1",
 "userArn":"arn:aws:iam::111111111111:user/Jane_Doe"
}
```

Connect to Amazon EC2 Instance with Your Private Key

Make sure that you have created an Amazon EC2 key pair. Add your public key to your user profile in AWS CodeStar. To create a key pair, see Step 4: Create an Amazon EC2 Key Pair for AWS CodeStar Projects. To add your public key to your user profile, see the instructions earlier in this topic.

To connect to an Amazon EC2 Linux instance by using your private key

1. With your project open in the AWS CodeStar console, in the navigation pane, choose **Project**.
2. In **Project Resources**, choose the **ARN** link in the row where **Type** is **Amazon EC2** and **Name** starts with **instance**.
3. In the Amazon EC2 console, choose **Connect**.
4. Follow the instructions in the **Connect To Your Instance** dialog box.

   For the user name, use `ec2-user`. If you use the wrong user name, you cannot connect to the instance.

For more information, see the following resources in the *Amazon EC2 User Guide*.

- Connecting to Your Linux Instance Using SSH
- Connecting to Your Linux Instance from Windows Using PuTTY
- Connecting to Your Linux Instance Using MindTerm

Security in AWS CodeStar

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

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

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs . To learn about the compliance programs that apply to AWS CodeStar, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS CodeStar. The following topics show you how to configure AWS CodeStar to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS CodeStar resources.

When you create custom policies and use permission boundaries in AWS CodeStar, ensure least-privilege access by granting only the permissions required to perform a task and scoping down the permissions to targeted resources. To prevent members of other projects from accessing resources in your project, grant organization members separate permissions for each AWS CodeStar project. As a best practice, create a project account for each member and then assign role-based access to that account.

For example, you can use a service such as AWS Control Tower with AWS Organizations to provision accounts for each developer role under a DevOps group. Then you can assign permissions to those accounts. The overall permissions apply to the account but the user has limited access to resources outside the project.

For more information about managing least-privilege access to AWS resources using a *multi-account strategy*, refer to AWS multi-account strategy for your landing zone in the *AWS Control Tower User Guide*.

Topics

- Data Protection in AWS CodeStar
- Identity and Access Management for AWS CodeStar
- Logging AWS CodeStar API Calls with AWS CloudTrail
- Compliance Validation for AWS CodeStar
- Resilience in AWS CodeStar
- Infrastructure Security in AWS CodeStar

Data Protection in AWS CodeStar

The AWS shared responsibility model applies to data protection in AWS CodeStar. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.

- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

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

Data Encryption in AWS CodeStar

By default, AWS CodeStar encrypts the information it stores about your project. Everything other than your project ID is encrypted at rest, such as project name, description, and user emails. Avoid putting personal information in your project IDs. AWS CodeStar also encrypts information in transit by default. No customer action is required for either encryption at rest or encryption in transit.

Identity and Access Management for AWS CodeStar

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS CodeStar resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating With Identities
- Managing Access Using Policies
- How AWS CodeStar Works with IAM
- AWS CodeStar Project-Level Policies and Permissions
- AWS CodeStar Identity-Based Policy Examples
- Troubleshooting AWS CodeStar Identity and Access

Audience

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

Service user – If you use the AWS CodeStar service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS CodeStar features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS CodeStar, see Troubleshooting AWS CodeStar Identity and Access.

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

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

Authenticating With Identities

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

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

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

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If

you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

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

AWS account root user

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

IAM users and Groups

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

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

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

IAM Roles

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

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

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

Managing Access Using Policies

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

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

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

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A

user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

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

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

Resource-Based Policies

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

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

Access Control Lists (ACLs)

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

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

Other Policy Types

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

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple Policy Types

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

How AWS CodeStar Works with IAM

Before you use IAM to manage access to AWS CodeStar, you should understand what IAM features are available to use with AWS CodeStar. To get a high-level view of how AWS CodeStar and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

- AWS CodeStar Identity-Based Policies
- AWS CodeStar Resource-Based Policies
- Authorization Based on AWS CodeStar Tags
- AWS CodeStar IAM Roles
- IAM user Access to AWS CodeStar
- Federated User Access to AWS CodeStar
- Using Temporary Credentials with AWS CodeStar
- Service-Linked Roles
- Service Roles

AWS CodeStar Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources and the conditions under which actions are allowed or denied. AWS CodeStar creates several identity-based policies on your behalf, which allow AWS CodeStar to create and manage resources within the scope of an AWS CodeStar project. AWS CodeStar supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

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

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

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

Policy actions in AWS CodeStar use the following prefix before the action: `codestar:`. For example, to allow a specified IAM user to edit the attributes of an AWS CodeStar project, such as its project description, you could use the following policy statement:

```
{
  "Version": "2012-10-17",
```

```
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codestar:UpdateProject"
      ],
      "Resource" : "arn:aws:codestar:us-east-2:project/my-first-projec"
    }
  ]
}
```

Policy statements must include either an `Action` or `NotAction` element. AWS CodeStar defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "codestar:action1",
      "codestar:action2"
```

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

```
"Action": "codestar:List*"
```

To see a list of AWS CodeStar actions, see Actions Defined by AWS CodeStar in the *IAM User Guide*.

Resources

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

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

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

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

The AWS CodeStar project resource has the following ARN:

```
arn:aws:codestar:region:account:project/resource-specifier
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, the following specifies the AWS CodeStar project named *my-first-projec* registered to the AWS account 111111111111 in the AWS Region us-east-2:

```
arn:aws:codestar:us-east-2:111111111111:project/my-first-projec
```

The following specifies any AWS CodeStar project that begins with the name my-proj registered to the AWS account 111111111111 in the AWS Region us-east-2:

```
arn:aws:codestar:us-east-2:111111111111:project/my-proj*
```

Some AWS CodeStar actions, such as for listing projects, cannot be performed on a resource. In those cases, you must use the wildcard (*).

```
"LisProjects": "*"
```

To see a list of AWS CodeStar resource types and their ARNs, see Resources Defined by AWS CodeStar in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS CodeStar.

Condition Keys

AWS CodeStar does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of AWS CodeStar identity-based policies, see AWS CodeStar Identity-Based Policy Examples.

AWS CodeStar Resource-Based Policies

AWS CodeStar does not support resource-based policies.

Authorization Based on AWS CodeStar Tags

You can attach tags to AWS CodeStar projects or pass tags in a request to AWS CodeStar. To control access based on tags, you provide tag information in the condition element of a policy using the `codestar:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys. For more information about tagging AWS CodeStar resources, see the section called "Working with Project Tags".

To view an example identity-based policy for limiting access to an AWS CodeStar project based on the tags on that project, see Viewing AWS CodeStar Projects Based on Tags.

AWS CodeStar IAM Roles

An IAM role is an entity in your AWS account that has specific permissions.

You can use AWS CodeStar as an IAM user, a federated user, the root user, or an assumed role. All user types with the appropriate permissions can manage project permissions to their AWS resources, but AWS CodeStar manages project permissions automatically for IAM users. IAM policies and roles grant permissions and access to that user based on the project role. You can use the IAM console to create other policies that assign AWS CodeStar and other permissions to an IAM user.

For example, you might want to allow a user to view, but not change, an AWS CodeStar project. In this case, you add the IAM user to an AWS CodeStar project with the viewer role. Every AWS CodeStar project has a set of policies that help you control access to the project. In addition, you can control which users have access to AWS CodeStar.

AWS CodeStar access is handled differently for IAM users and federated users. Only IAM users can be added to teams. To grant IAM users permissions to projects, you add the user to the project team and assign the user a role. To grant federated users permissions to projects, you manually attach the AWS CodeStar project role's managed policy to the federated user's role.

This table summarizes the tools available for each type of access.

Permissions feature	IAM user	Federated user	Root user
SSH key management for remote access for Amazon EC2 and Elastic Beanstalk projects	✓		

Permissions feature	IAM user	Federated user	Root user
AWS CodeCommit SSH access	✓		
IAM user permissions managed by AWS CodeStar	✓		
Project permissions managed manually		✓	✓
Users can be added to project as team members	✓		

IAM user Access to AWS CodeStar

When you add an IAM user to a project and choose a role for the user, AWS CodeStar applies the appropriate policy to the IAM user automatically. For IAM users, you don't need to directly attach or manage policies or permissions in IAM. For information about adding an IAM user to an AWS CodeStar project, see Add Team Members to an AWS CodeStar Project . For information about removing an IAM user from an AWS CodeStar project, see Remove Team Members from an AWS CodeStar Project .

Attach an Inline Policy to an IAM user

When you add a user to a project, AWS CodeStar automatically attaches the managed policy for the project that matches the user's role. You should not manually attach an AWS CodeStar managed policy for a project to an IAM user. With the exception of `AWSCodeStarFullAccess`, we do not recommend that you attach policies that change an IAM user's permissions in an AWS CodeStar project. If you decide to create and attach your own policies, see Adding and Removing IAM Identity Permissions in the *IAM User Guide*.

Federated User Access to AWS CodeStar

Instead of creating an IAM user or using the root user, you can use user identities from AWS Directory Service, your enterprise user directory, a web identity provider, or IAM users assuming roles. These are known as *federated users*.

Grant federated users access to your AWS CodeStar project by manually attaching the managed policies described in AWS CodeStar Project-Level Policies and Permissions to the user's IAM role.

You attach the owner, contributor, or viewer policy after AWS CodeStar creates your project resources and IAM roles.

Prerequisites:

- You must have set up an identity provider. For example, you could set up a SAML identity provider and set up AWS authentication through the provider. For more information about setting up an identity provider, see Creating IAM Identity Providers. For more information about SAML federation, see About SAML 2.0-based Federation.
- You must have created a role for a federated user to assume when access is requested through an identity provider. An STS trust policy must be attached to the role that allows federated users to assume the role. For more information, see Federated Users and Roles in the *IAM User Guide*.
- You must have created your AWS CodeStar project and know the project ID.

For more information about creating a role for identity providers, see Creating a Role for a Third-Party Identity Provider (Federation).

Attach the AWSCodeStarFullAccess Managed Policy to the Federated User's Role

Grant a federated user permissions to create a project by attaching the `AWSCodeStarFullAccess` managed policy. To perform these steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the associated `AdministratorAccess` managed policy or equivalent.

> **Note**
>
> After you create the project, your project owner permissions are not applied automatically. Using a role with administrative permissions for your account, attach the owner managed policy, as described in Attach Your Project's AWS CodeStar Viewer/Contributor/Owner Managed Policy to the Federated User's Role.

1. Open the IAM console. In the navigation pane, choose **Policies**.
2. Enter `AWSCodeStarFullAccess` in the search field. The policy name is displayed, with a policy type of **AWS managed**. You can expand the policy to see the permissions in the policy statement.

3. Select the circle next to the policy, and then under **Policy actions**, choose **Attach**.
4. On the **Summary** page, choose the **Attached entities** tab. Choose **Attach**.
5. On the **Attach Policy** page, filter for the federated user's role in the search field. Select the box next to the name of the role, and then choose **Attach policy**. The **Attached entities** tab displays the new attachment.

Attach Your Project's AWS CodeStar Viewer/Contributor/Owner Managed Policy to the Federated User's Role

Grant federated users access to your project by attaching the appropriate owner, contributor, or viewer managed policy to the user's role. The managed policy gives the appropriate level of permissions. Unlike IAM users, you must manually attach and detach managed policies for federated users. This is equivalent to assigning project permissions to team members in AWS CodeStar. To perform these steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the associated `AdministratorAccess` managed policy or equivalent.

Prerequisites:

- You must have created a role or have an existing role that your federated user assumes.
- You must know which level of permissions you want to grant. The managed policies attached to the owner, contributor, and viewer roles provide role-based permissions for your project.
- Your AWS CodeStar project must have been created. The managed policy is not available in IAM until the project is created.

1. Open the IAM console. In the navigation pane, choose **Policies**.
2. Enter your project ID in the search field. The policy name matching your project is displayed, with a policy type of **Customer managed**. You can expand the policy to see the permissions in the policy statement.
3. Choose one of these managed policies. Select the circle next to the policy, and then under **Policy actions**, choose **Attach**.
4. On the **Summary** page, choose the **Attached entities** tab. Choose **Attach**.
5. On the **Attach Policy** page, filter for the federated user's role in the search field. Select the box next to the name of the role and then choose **Attach policy**. The **Attached entities** tab displays the new attachment.

Detach an AWS CodeStar Managed Policy from the Federated User's Role

Before you delete your AWS CodeStar project, you must manually detach any managed policies you attached to a federated user's role. To perform these steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the associated `AdministratorAccess` managed policy or equivalent.

1. Open the IAM console. In the navigation pane, choose **Policies**.
2. Enter your project ID in the search field.
3. Select the circle next to the policy, and then under **Policy actions**, choose **Attach**.
4. On the **Summary** page, choose the **Attached entities** tab.
5. Filter for the federated user's role in the search field. Choose **Detach**.

Attach an AWS Cloud9 Managed Policy to the Federated User's Role

If you are using an AWS Cloud9 development environment, grant federated users access to it by attaching the `AWSCloud9User` managed policy to the user's role. Unlike IAM users, you must manually attach and detach managed policies for federated users. To perform these steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the associated `AdministratorAccess` managed policy or equivalent.

Prerequisites:

- You must have created a role or have an existing role that your federated user assumes.
- You must know which level of permissions you want to grant:
  - The `AWSCloud9User` managed policy allows the user to do the following:
    - Create their own AWS Cloud9 development environments.
    - Get information about their environments.
    - Change the settings for their environments.
  - The `AWSCloud9Administrator` managed policy allows the user to do the following for themselves or others:
    - Create environments.
    - Get information about environments.
    - Delete environments.

- Change the settings of environments.

1. Open the IAM console. In the navigation pane, choose **Policies**.
2. Enter the policy name in the search field. The managed policy is displayed, with a policy type of **AWS managed**. You can expand the policy to see the permissions in the policy statement.
3. Choose one of these managed policies. Select the circle next to the policy, and then under **Policy actions**, choose **Attach**.
4. On the **Summary** page, choose the **Attached entities** tab. Choose **Attach**.
5. On the **Attach Policy** page, filter for the federated user's role in the search field. Choose the box next to the name of the role and then choose **Attach policy**. The **Attached entities** tab displays the new attachment.

Detach an AWS Cloud9 Managed Policy from the Federated User's Role

If you are using an AWS Cloud9 development environment, you can remove a federated user's access to it by detaching the policy that grants access. To perform these steps, you must have signed in to the console either as a root user, an administrator user in the account, or an IAM user or federated user with the associated `AdministratorAccess` managed policy or equivalent.

1. Open the IAM console. In the navigation pane, choose **Policies**.
2. Enter your project name in the search field.
3. Select the circle next to the policy, and then under **Policy actions**, choose **Attach**.
4. On the **Summary** page, choose the **Attached entities** tab.
5. Filter for the federated user's role in the search field. Choose **Detach**.

Using Temporary Credentials with AWS CodeStar

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

AWS CodeStar supports the use of temporary credentials, but the AWS CodeStar team member functionality doesn't work for federated access. AWS CodeStar team member functionality only supports adding an IAM user as a team member.

Service-Linked Roles

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

AWS CodeStar does not support service-linked roles.

Service Roles

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

AWS CodeStar supports service roles. AWS CodeStar uses a service role, aws-codestar-service-role, when it creates and manages the resources for your project. For more information, see Roles Terms and Concepts in the *IAM User Guide*.

> **Important**
>
> You must be signed in as an administrator user or root account to create this service role. For more information, see First-Time Access Only: Your root user Credentials and Creating Your First Admin User and Group in the *IAM User Guide*.

This role is created for you the first time you create a project in AWS CodeStar. The service role acts on your behalf to:

- Create the resources you choose when you create a project.
- Display information about those resources in the AWS CodeStar project dashboard.

It also acts on your behalf when you manage the resources for a project. For an example of this policy statement, see AWSCodeStarServiceRole Policy.

In addition, AWS CodeStar creates several project-specific service roles, depending on the project type. AWS CloudFormation and toolchain roles are created for each project type.

- AWS CloudFormation roles allow AWS CodeStar to access AWS CloudFormation to create and modify stacks for your AWS CodeStar project.

- Toolchain roles allow AWS CodeStar to access other AWS services to create and modify resources for your AWS CodeStar project.

AWS CodeStar Project-Level Policies and Permissions

When you create a project, AWS CodeStar creates the IAM roles and policies you need to manage your project resources. The policies fall into three categories:

- IAM policies for project team members.
- IAM policies for worker roles.
- IAM policies for a runtime execution role.

IAM Policies for Team Members

When you create a project, AWS CodeStar creates three customer managed policies for owner, contributor, and viewer access to the project. All AWS CodeStar projects contain IAM policies for these three access levels. These access levels are project-specific and defined by an IAM managed policy with a standard name, where *`project-id`* is the ID of the AWS CodeStar project (for example, *`my-first-projec`*):

- `CodeStar_`*`project-id`*`_Owner`
- `CodeStar_`*`project-id`*`_Contributor`
- `CodeStar_`*`project-id`*`_Viewer`

> **Important**
>
> These policies are subject to change by AWS CodeStar. They should not be edited manually. If you want to add or change permissions, attach additional policies to the IAM user.

As you add team members (IAM users) to the project and choose their access levels, the corresponding policy is attached to the IAM user, granting the user the appropriate set of permissions to act on the project resources. Under most circumstances, you don't need to directly attach or manage policies or permissions in IAM. Manually attaching an AWS CodeStar access level policy to an IAM user is not recommended. If absolutely necessary, as a supplement to an AWS

CodeStar access level policy, you can create your own managed or inline policies to apply your own level of permissions to an IAM user.

The policies are tightly scoped to project resources and specific actions. As new resources are added to the infrastructure stack, AWS CodeStar attempts to update the team member policies to include permissions to access the new resource, if they are one of the supported resource types.

> **Note**
>
> The policies for access levels in an AWS CodeStar project apply to that project only. This helps ensure that users can only see and interact with the AWS CodeStar projects they have permissions to, at the level determined by their role. Only users who create AWS CodeStar projects should have a policy applied that allows access to all AWS CodeStar resources, regardless of project.

All AWS CodeStar access level policies vary, depending on the AWS resources associated with the project with which the access levels are associated. Unlike other AWS services, these policies are customized when the project is created and updated as project resources change. Therefore, there is no one canonical owner, contributor, or viewer managed policy.

AWS CodeStar Owner Role Policy

The `CodeStar_`*`project-id`*`_Owner` customer managed policy allows a user to perform all actions in the AWS CodeStar project with no restrictions. This is the only policy that allows a user to add or remove team members. The contents of the policy vary, depending on the resources associated with the project. See AWS CodeStar Owner Role Policy for an example.

An IAM user with this policy can perform all AWS CodeStar actions in the project, but unlike an IAM user with the `AWSCodeStarFullAccess` policy, the user cannot create projects. The `codestar:*` permission is limited in scope to a specific resource (the AWS CodeStar project associated with that project ID).

AWS CodeStar Contributor Role Policy

The `CodeStar_`*`project-id`*`_Contributor` customer managed policy allows a user to contribute to the project and change the project dashboard, but does not allow a user to add or remove team members. The contents of the policy vary, depending on the resources associated with the project. See AWS CodeStar Contributor Role Policy for an example.

AWS CodeStar Viewer Role Policy

The `CodeStar_project-id_Viewer` customer managed policy allows a user to view a project in AWS CodeStar, but not change its resources or add or remove team members. The contents of the policy vary, depending on the resources associated with the project. See AWS CodeStar Viewer Role Policy for an example.

IAM Policies for Worker Roles

If you create your AWS CodeStar project after December 6, 2018 PDT, AWS CodeStar creates two worker roles, `CodeStar-project-id-ToolChain` and `CodeStar-project-id-CloudFormation`. A worker role is a project-specific IAM role that AWS CodeStar creates to pass to a service. It grants permissions so that the service can create resources and execute actions in the context of your AWS CodeStar project. The toolchain worker role has a trust relationship established with toolchain services such as CodeBuild, CodeDeploy, and CodePipeline. Project team members (owners and contributors) are granted access to pass the worker role to trusted downstream services. For an example of the inline policy statement for this role, see AWS CodeStar Toolchain Worker Role Policy (After December 6, 2018 PDT).

The CloudFormation worker role includes permissions for selected resources supported by AWS CloudFormation, as well as permissions to create IAM users, roles, and policies in your application stack. It also has a trust relationship established with AWS CloudFormation. To mitigate risks of privilege escalation and destructive actions, the AWS CloudFormation role policy includes a condition that requires the project-specific permissions boundary for every IAM entity (user or role) created in the infrastructure stack. For an example of the inline policy statement for this role, see AWS CloudFormation Worker Role Policy.

For AWS CodeStar projects created before December 6, 2018 PDT AWS CodeStar creates individual worker roles for toolchain resources such as CodePipeline, CodeBuild, and CloudWatch Events, and also creates a worker role for AWS CloudFormation that supports a limited set of resources. Each of these roles has a trust relationship established with the corresponding service. Project team members (owners and contributors) and some of the other worker roles are given access to pass the role to the trusted downstream services. Permissions for the worker roles are defined in an inline policy that is scoped down to a basic set of actions that the role can perform on a set of project resources. These permissions are static. They include permissions to resources that are included in the project at creation, but are not updated when new resources are added to the project. For examples of these policy statements, see:

- AWS CloudFormation Worker Role Policy (Before December 6, 2018 PDT)

- AWS CodePipeline Worker Role Policy (Before December 6, 2018 PDT)
- AWS CodeBuild Worker Role Policy (Before December 6, 2018 PDT)
- Amazon CloudWatch Events Worker Role Policy (Before December 6, 2018 PDT)

IAM Policy for the Execution Role

For projects created after December 6, 2018 PDT, AWS CodeStar creates a generic execution role for the sample project in your application stack. The role is scoped down to project resources using the permissions boundary policy. As you expand on the sample project, you can create additional IAM roles, and the AWS CloudFormation role policy requires that these roles be scoped down using the permission boundary to avoid escalation of privileges. For more information, see Add an IAM Role to a Project.

For Lambda projects created before December 6, 2018 PDT, AWS CodeStar creates a Lambda execution role that has an inline policy attached with permissions to act on the resources in the project AWS SAM stack. As new resources are added to the SAM template, AWS CodeStar attempts to update the Lambda execution role policy to include permissions to the new resource if they are one of the supported resource types.

IAM Permissions Boundary

After December 6, 2018 PDT, when you create a project AWS CodeStar creates a customer managed policy and assigns that policy as the IAM permissions boundary to IAM roles in the project. AWS CodeStar requires all IAM entities created in the application stack to have a permissions boundary. A permissions boundary controls the maximum permissions that the role can have, but does not provide the role with any permissions. Permissions policies define the permissions for the role. This means that no matter how many extra permissions are added to a role, anyone using the role cannot perform more than the actions included in the permissions boundary. For information about how permissions policies and permissions boundaries are evaluated, see Policy Evaluation Logic in the *IAM User Guide*.

AWS CodeStar uses a project-specific permissions boundary to prevent escalation of privileges to resources outside the project. The AWS CodeStar permissions boundary includes ARNs for project resources. For an example of this policy statement, see AWS CodeStar Permissions Boundary Policy.

The AWS CodeStar transform updates this policy when you add or remove a supported resource from the project through the application stack (`template.yml`).

Add an IAM Permissions Boundary to Existing Projects

If you have an AWS CodeStar project that was created before December 6, 2018 PDT, you should manually add a permission boundary to the IAM roles in the project. As a best practice, we recommend using a project-specific boundary that includes only resources in the project to avoid escalation of privilege to resources outside the project. Follow these steps to use the AWS CodeStar managed permission boundary that is updated as the project evolves.

1. Sign into the AWS CloudFormation console and locate the template for the toolchain stack in your project. This template is named `awscodestar-`*`project-id`*.
2. Choose the template, choose **Actions**, and then choose **View/Edit template in Designer**.
3. Locate the `Resources` section, and include the following snippet at the top of the section.

```
 PermissionsBoundaryPolicy:
   Description: Creating an IAM managed policy for defining the permissions boundary
for an AWS CodeStar project
   Type: AWS::IAM::ManagedPolicy
   Properties:
     ManagedPolicyName: !Sub 'CodeStar_${ProjectId }_PermissionsBoundary'
     Description: 'IAM policy to define the permissions boundary for IAM entities
created in an AWS CodeStar project'
     PolicyDocument:
       Version: '2012-10-17'
       Statement:
       - Sid: '1'
         Effect: Allow
         Action: ['*']
         Resource:
           - !Sub 'arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/awscodestar-${ProjectId}-*'
```

   You might need additional IAM permissions to update the stack from the AWS CloudFormation console.
4. (Optional) If you want to create application-specific IAM roles, complete this step. From the IAM console, update the inline policy attached to the AWS CloudFormation role for your project to include the following snippet. You might need additional IAM resources to update the policy.

```
       {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::{AccountId}:role/CodeStar-{ProjectId}*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:GetRole",
                "iam:DeleteRole",
                "iam:DeleteUser"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:AttachRolePolicy",
                "iam:AttachUserPolicy",
                "iam:CreateRole",
                "iam:CreateUser",
                "iam:DeleteRolePolicy",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:DetachRolePolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutRolePermissionsBoundary"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::{AccountId}:policy/
CodeStar_{ProjectId}_PermissionsBoundary"
                }
            },
            "Effect": "Allow"
        }
```

5. Push a change through your project pipeline so that AWS CodeStar updates the permissions boundary with appropriate permissions.

For more information, see Add an IAM Role to a Project.

AWS CodeStar Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS CodeStar resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

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

Topics

- Policy Best Practices
- AWSCodeStarServiceRole Policy
- AWSCodeStarFullAccess Policy
- AWS CodeStar Owner Role Policy
- AWS CodeStar Contributor Role Policy
- AWS CodeStar Viewer Role Policy
- AWS CodeStar Toolchain Worker Role Policy (After December 6, 2018 PDT)
- AWS CloudFormation Worker Role Policy
- AWS CloudFormation Worker Role Policy (Before December 6, 2018 PDT)
- AWS CodePipeline Worker Role Policy (Before December 6, 2018 PDT)
- AWS CodeBuild Worker Role Policy (Before December 6, 2018 PDT)
- Amazon CloudWatch Events Worker Role Policy (Before December 6, 2018 PDT)
- AWS CodeStar Permissions Boundary Policy
- Listing Resources for a Project
- Using the AWS CodeStar Console
- Allow Users to View Their Own Permissions
- Updating an AWS CodeStar Project
- Adding a Team Member to a Project

- Listing User Profiles Associated with an AWS Account
- Viewing AWS CodeStar Projects Based on Tags
- AWS CodeStarupdates to AWS managed policies

Policy Best Practices

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

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

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

AWSCodeStarServiceRole Policy

The `aws-codestar-service-role` policy is attached to the service role that allows AWS CodeStar to perform actions with other services. The first time you sign in to AWS CodeStar, you create the service role. You only need to create it once. The policy is automatically attached to the service role after you create it.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ProjectEventRules",
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:RemoveTargets",
                "events:PutRule",
                "events:DeleteRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/awscodestar-*"
            ]
        },
        {
            "Sid": "ProjectStack",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*Stack*",
                "cloudformation:CreateChangeSet",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:GetTemplate"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awscodestar-*",
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/aws-cloud9-*",
                "arn:aws:cloudformation:*:aws:transform/CodeStar*"
            ]
```

```
        },
        {
            "Sid": "ProjectStackTemplate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:GetTemplateSummary",
                "cloudformation:DescribeChangeSet"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ProjectQuickstarts",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::awscodestar-*/*"
            ]
        },
        {
            "Sid": "ProjectS3Buckets",
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::aws-codestar-*",
                "arn:aws:s3:::elasticbeanstalk-*"
            ]
        },
        {
            "Sid": "ProjectServices",
            "Effect": "Allow",
            "Action": [
                "codestar:*",
                "codecommit:*",
                "codepipeline:*",
                "codedeploy:*",
                "codebuild:*",
                "autoscaling:*",
                "cloudwatch:Put*",
                "ec2:*",
                "elasticbeanstalk:*",
```

```
                "elasticloadbalancing:*",
                "iam:ListRoles",
                "logs:*",
                "sns:*",
                "cloud9:CreateEnvironmentEC2",
                "cloud9:DeleteEnvironment",
                "cloud9:DescribeEnvironment*",
                "cloud9:ListEnvironments"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ProjectWorkerRoles",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:PassRole",
                "iam:GetRolePolicy",
                "iam:PutRolePolicy",
                "iam:SetDefaultPolicyVersion",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:AddRoleToInstanceProfile",
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": [
                "arn:aws:iam::*:role/CodeStarWorker*",
                "arn:aws:iam::*:policy/CodeStarWorker*",
                "arn:aws:iam::*:instance-profile/awscodestar-*"
            ]
        },
        {
            "Sid": "ProjectTeamMembers",
            "Effect": "Allow",
            "Action": [
                "iam:AttachUserPolicy",
                "iam:DetachUserPolicy"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "iam:PolicyArn": [
                        "arn:aws:iam::*:policy/CodeStar_*"
                    ]
                }
            }
        },
        {
            "Sid": "ProjectRoles",
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:CreatePolicyVersion",
                "iam:DeletePolicyVersion",
                "iam:ListEntitiesForPolicy",
                "iam:ListPolicyVersions",
                "iam:GetPolicy",
                "iam:GetPolicyVersion"
            ],
            "Resource": [
                "arn:aws:iam::*:policy/CodeStar_*"
            ]
        },
        {
            "Sid": "InspectServiceRole",
            "Effect": "Allow",
            "Action": [
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-codestar-service-role",
                "arn:aws:iam::*:role/service-role/aws-codestar-service-role"
            ]
        },
        {
            "Sid": "IAMLinkRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "cloud9.amazonaws.com"
                }
            }
        },
        {
            "Sid": "DescribeConfigRuleForARN",
            "Effect": "Allow",
            "Action": [
                "config:DescribeConfigRules"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "ProjectCodeStarConnections",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:UseConnection",
                "codestar-connections:GetConnection"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ProjectCodeStarConnectionsPassConnections",
            "Effect": "Allow",
            "Action": "codestar-connections:PassConnection",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "codestar-connections:PassedToService":
 "codepipeline.amazonaws.com"
                }
            }
        }
    ]
}
```

AWSCodeStarFullAccess Policy

In the Setting Up AWS CodeStar instructions, you attached a policy named `AWSCodeStarFullAccess` to your IAM user. This policy statement allows the user to perform all available actions in AWS CodeStar with all available AWS CodeStar resources associated with the AWS account. This includes creating and deleting projects. The following example is a snippet of a representative `AWSCodeStarFullAccess` policy. The actual policy differs depending on the template you select when you start a new AWS CodeStar project.

AWS CloudFormation requires `cloudformation::ListStacks` permission when calling `cloudformation::DescribeStacks` without a target stack.

Permissions details

This policiy includes permissions to do the following:

- `ec2`–Retrieve information about EC2 instances to create a AWS CodeStar project.
- `cloud9`–Retrieve information about AWS Command Line Interface environments.
- `cloudformation`–Retrieve information about AWS CodeStar project stacks.
- `codestar`–Perform actions within a AWS CodeStar project.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeStarEC2",
      "Effect": "Allow",
      "Action": [
        "codestar:*",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "cloud9:DescribeEnvironment*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarCF",
      "Effect": "Allow",
      "Action": [
```

```
        "cloudformation:DescribeStack*",
        "cloudformation:ListStacks*",
        "cloudformation:GetTemplateSummary"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/awscodestar-*"
      ]
    }
  ]
}
```

You might not want to give all users this much access. Instead, you can add project-level permissions using project roles managed by AWS CodeStar. The roles grant specific levels of access to AWS CodeStar projects and are named as follows:

- Owner
- Contributor
- Viewer

AWS CodeStar Owner Role Policy

The AWS CodeStar owner role policy allows a user to perform all actions in an AWS CodeStar project with no restrictions. AWS CodeStar applies the `CodeStar_`*`project-id`*`_Owner` policy to project team members with the owner access level..

```
...
{
  "Effect": "Allow",
  "Action": [
    ...
    "codestar:*",
    ...
  ],
  "Resource": [
    "arn:aws:codestar:us-east-2:111111111111:project/project-id",
    "arn:aws:iam::account-id:policy/CodeStar_project-id_Owner"
  ]
},
{
  "Effect": "Allow",
  "Action": [
```

```
    "codestar:DescribeUserProfile",
    "codestar:ListProjects",
    "codestar:ListUserProfiles",
    "codestar:VerifyServiceRole",
    ...
  ],
  "Resource": [
    "*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "codestar:*UserProfile",
    ...
  ],
  "Resource": [
    "arn:aws:iam::account-id:user/user-name"
  ]
}
...
```

AWS CodeStar Contributor Role Policy

The AWS CodeStar contributor role policy allows a user to contribute to the project and change the project dashboard.AWS CodeStar applies the `CodeStar_project-id_Contributor` policy to project team members with the contributor access level. Users with contributor access can contribute to the project and change the project dashboard, but cannot add or remove team members.

```
...
{
  "Effect": "Allow",
  "Action": [
    ...
    "codestar:Describe*",
    "codestar:Get*",
    "codestar:List*",
    "codestar:PutExtendedAccess",
    ...
  ],
  "Resource": [
    "arn:aws:codestar:us-east-2:111111111111:project/project-id",
```

```
      "arn:aws:iam::account-id:policy/CodeStar_project-id_Contributor"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "codestar:DescribeUserProfile",
      "codestar:ListProjects",
      "codestar:ListUserProfiles",
      "codestar:VerifyServiceRole",
      ...
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "codestar:*UserProfile",
      ...
    ],
    "Resource": [
      "arn:aws:iam::account-id:user/user-name"
    ]
  }
  ...
```

AWS CodeStar Viewer Role Policy

The AWS CodeStar viewer role policy allows a user to view a project in AWS CodeStar. AWS CodeStar applies the `CodeStar_project-id_Viewer` policy to project team members with the viewer access level. Users with viewer access can view a project in AWS CodeStar, but not change its resources or add or remove team members.

```
...
{
  "Effect": "Allow",
  "Action": [
    ...
    "codestar:Describe*",
    "codestar:Get*",
    "codestar:List*",
```

```
      ...
    ],
    "Resource": [
      "arn:aws:codestar:us-east-2:111111111111:project/project-id",
      "arn:aws:iam::account-id:policy/CodeStar_project-id_Viewer"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "codestar:DescribeUserProfile",
      "codestar:ListProjects",
      "codestar:ListUserProfiles",
      "codestar:VerifyServiceRole",
      ...
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "codestar:*UserProfile",
      ...
    ],
    "Resource": [
      "arn:aws:iam::account-id:user/user-name"
    ]
  }
  ...
```

AWS CodeStar Toolchain Worker Role Policy (After December 6, 2018 PDT)

For AWS CodeStar projects created after December 6, 2018 PDT, AWS CodeStar creates an inline policy for a worker role that creates resources for your project in other AWS services. The contents of the policy depend on the type of project you are creating. The following policy is an example. For more information, see IAM Policies for Worker Roles.

```
{
  "Statement": [
    {
      "Action": [
```

```
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning",
        "s3:PutObject*",
        "codecommit:CancelUploadArchive",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:GitPull",
        "codecommit:UploadArchive",
        "codebuild:StartBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:StopBuild",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeChangeSet",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "codepipeline:StartPipelineExecution",
        "lambda:ListFunctions",
        "lambda:InvokeFunction",
        "sns:Publish"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:GenerateDataKey*",
        "kms:Encrypt",
```

```
        "kms:Decrypt"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS CloudFormation Worker Role Policy

For AWS CodeStar projects created after December 6, 2018 PDT, AWS CodeStar creates an inline policy for a worker role that creates AWS CloudFormation resources for your AWS CodeStar project. The contents of the policy depend on the type of resources required for your project. The following policy is an example. For more information, see IAM Policies for Worker Roles.

```
{
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::aws-codestar-region-id-account-id-project-id",
                "arn:aws:s3:::aws-codestar-region-id-account-id-project-id/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "apigateway:DELETE",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:POST",
                "apigateway:PUT",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentConfig",
```

```
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:DeleteApplication",
                "codedeploy:DeleteDeployment",
                "codedeploy:DeleteDeploymentConfig",
                "codedeploy:DeleteDeploymentGroup",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:RegisterApplicationRevision",
                "codestar:SyncResources",
                "config:DeleteConfigRule",
                "config:DescribeConfigRules",
                "config:ListTagsForResource",
                "config:PutConfigRule",
                "config:TagResource",
                "config:UntagResource",
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:DescribeContinuousBackups",
                "dynamodb:DescribeTable",
                "dynamodb:DescribeTimeToLive",
                "dynamodb:ListTagsOfResource",
                "dynamodb:TagResource",
                "dynamodb:UntagResource",
                "dynamodb:UpdateContinuousBackups",
                "dynamodb:UpdateTable",
                "dynamodb:UpdateTimeToLive",
                "ec2:AssociateIamInstanceProfile",
                "ec2:AttachVolume",
                "ec2:CreateSecurityGroup",
                "ec2:createTags",
                "ec2:DescribeIamInstanceProfileAssociations",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DetachVolume",
                "ec2:DisassociateIamInstanceProfile",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyInstanceCreditSpecification",
                "ec2:ModifyInstancePlacement",
                "ec2:MonitorInstances",
                "ec2:ReplaceIamInstanceProfileAssociation",
                "ec2:RunInstances",
                "ec2:StartInstances",
```

```
                "ec2:StopInstances",
                "ec2:TerminateInstances",
                "events:DeleteRule",
                "events:DescribeRule",
                "events:ListTagsForResource",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "events:TagResource",
                "events:UntagResource",
                "kinesis:AddTagsToStream",
                "kinesis:CreateStream",
                "kinesis:DecreaseStreamRetentionPeriod",
                "kinesis:DeleteStream",
                "kinesis:DescribeStream",
                "kinesis:IncreaseStreamRetentionPeriod",
                "kinesis:RemoveTagsFromStream",
                "kinesis:StartStreamEncryption",
                "kinesis:StopStreamEncryption",
                "kinesis:UpdateShardCount",
                "lambda:CreateAlias",
                "lambda:CreateFunction",
                "lambda:DeleteAlias",
                "lambda:DeleteFunction",
                "lambda:DeleteFunctionConcurrency",
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:ListTags",
                "lambda:ListVersionsByFunction",
                "lambda:PublishVersion",
                "lambda:PutFunctionConcurrency",
                "lambda:TagResource",
                "lambda:UntagResource",
                "lambda:UpdateAlias",
                "lambda:UpdateFunctionCode",
                "lambda:UpdateFunctionConfiguration",
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:DeleteBucketWebsite",
                "s3:PutAccelerateConfiguration",
                "s3:PutAnalyticsConfiguration",
                "s3:PutBucketAcl",
                "s3:PutBucketCORS",
                "s3:PutBucketLogging",
```

```
                "s3:PutBucketNotification",
                "s3:PutBucketPublicAccessBlock",
                "s3:PutBucketVersioning",
                "s3:PutBucketWebsite",
                "s3:PutEncryptionConfiguration",
                "s3:PutInventoryConfiguration",
                "s3:PutLifecycleConfiguration",
                "s3:PutMetricsConfiguration",
                "s3:PutReplicationConfiguration",
                "sns:CreateTopic",
                "sns:DeleteTopic",
                "sns:GetTopicAttributes",
                "sns:ListSubscriptionsByTopic",
                "sns:ListTopics",
                "sns:SetSubscriptionAttributes",
                "sns:Subscribe",
                "sns:Unsubscribe",
                "sqs:CreateQueue",
                "sqs:DeleteQueue",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ListQueueTags",
                "sqs:TagQueue",
                "sqs:UntagQueue"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": [
                "arn:aws:lambda:region-id:account-id:function:awscodestar-*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/CodeStar-project-id*"
```

```
            ],
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "codedeploy.amazonaws.com"
                }
            },
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CodeDeploy"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:region-id:aws:transform/Serverless-2016-10-31",
                "arn:aws:cloudformation:region-id:aws:transform/CodeStar"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:GetRole",
                "iam:DeleteRole",
                "iam:DeleteUser"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::account-id:policy/
CodeStar_project-id_PermissionsBoundary"
                }
            },
```

```
            "Action": [
                "iam:AttachRolePolicy",
                "iam:AttachUserPolicy",
                "iam:CreateRole",
                "iam:CreateUser",
                "iam:DeleteRolePolicy",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:DetachRolePolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutRolePermissionsBoundary"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:CreateKey",
                "kms:CreateAlias",
                "kms:DeleteAlias",
                "kms:DisableKey",
                "kms:EnableKey",
                "kms:UpdateAlias",
                "kms:TagResource",
                "kms:UntagResource"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringEquals": {
                    "ssm:ResourceTag/awscodestar:projectArn":
 "arn:aws:codestar:project-id:account-id:project/project-id"
                }
            },
            "Action": [
                "ssm:GetParameter*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS CloudFormation Worker Role Policy (Before December 6, 2018 PDT)

If your AWS CodeStar project was created before December 6, 2018 PDT, AWS CodeStar created an inline policy for an AWS CloudFormation worker role. The following policy statement is an example.

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe",
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codestar:SyncResources",
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:AddPermission",
                "lambda:UpdateFunction",
                "lambda:UpdateFunctionCode",
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration",
                "lambda:UpdateFunctionConfiguration",
                "lambda:RemovePermission",
                "lambda:listTags",
                "lambda:TagResource",
                "lambda:UntagResource",
                "apigateway:*",
                "dynamodb:CreateTable",
                "dynamodb:DeleteTable",
                "dynamodb:DescribeTable",
                "kinesis:CreateStream",
                "kinesis:DeleteStream",
```

```
                "kinesis:DescribeStream",
                "sns:CreateTopic",
                "sns:DeleteTopic",
                "sns:ListTopics",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "config:DescribeConfigRules",
                "config:PutConfigRule",
                "config:DeleteConfigRule",
                "ec2:*",
                "autoscaling:*",
                "elasticloadbalancing:*",
                "elasticbeanstalk:*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/CodeStarWorker-project-id-Lambda"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:us-east-1:aws:transform/Serverless-2016-10-31",
                "arn:aws:cloudformation:us-east-1:aws:transform/CodeStar"
            ],
            "Effect": "Allow"
        }
    ]
}
```

AWS CodePipeline Worker Role Policy (Before December 6, 2018 PDT)

If your AWS CodeStar project was created before December 6, 2018 PDT, AWS CodeStar created an inline policy for a CodePipeline worker role. The following policy statement is an example.

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe",
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codecommit:CancelUploadArchive",
                "codecommit:GetBranch",
                "codecommit:GetCommit",
                "codecommit:GetUploadArchiveStatus",
                "codecommit:UploadArchive"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:account-id:project-id"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds",
                "codebuild:StopBuild"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-1:account-id:project/project-id"
            ],
            "Effect": "Allow"
```

```
        },
        {
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeChangeSet",
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:ExecuteChangeSet"
            ],
            "Resource": [
                "arn:aws:cloudformation:us-east-1:account-id:stack/awscodestar-project-
id-lambda/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CloudFormation"
            ],
            "Effect": "Allow"
        }
    ]
}
```

AWS CodeBuild Worker Role Policy (Before December 6, 2018 PDT)

If your AWS CodeStar project was created before December 6, 2018 PDT, AWS CodeStar created an inline policy for an CodeBuild worker role. The following policy statement is an example.

```
{
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe",
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe/*",
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-app",
                "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-app/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codecommit:GitPull"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:account-id:project-id"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:GenerateDataKey*",
                "kms:Encrypt",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:account-id:alias/aws/s3"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Amazon CloudWatch Events Worker Role Policy (Before December 6, 2018 PDT)

If your AWS CodeStar project was created before December 6, 2018 PDT, AWS CodeStar created an inline policy for an CloudWatch Events worker role. The following policy statement is an example.

```
{
```

```
    "Statement": [
        {
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-east-1:account-id:project-id-Pipeline"
            ],
            "Effect": "Allow"
        }
    ]
}
```

AWS CodeStar Permissions Boundary Policy

If you create an AWS CodeStar project after December 6, 2018 PDT, AWS CodeStar creates a permissions boundary policy for your project. This policy prevents the escalation of privileges to resources outside the project. It is a dynamic policy that updates as the project evolves. The contents of the policy depend on the type of project you are creating. The following policy is an example. For more information, see IAM Permissions Boundary.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/AWSLogs/*/Config/*"
      ]
    },
    {
      "Sid": "2",
      "Effect": "Allow",
      "Action": [
        "*"
      ],
      "Resource": [
        "arn:aws:codestar:us-east-1:account-id:project/project-id",
```

```
        "arn:aws:cloudformation:us-east-1:account-id:stack/awscodestar-project-id-
lambda/eefbbf20-c1d9-11e8-8a3a-500c28b4e461",
        "arn:aws:cloudformation:us-east-1:account-id:stack/awscodestar-project-
id/4b80b3f0-c1d9-11e8-8517-500c28b236fd",
        "arn:aws:codebuild:us-east-1:account-id:project/project-id",
        "arn:aws:codecommit:us-east-1:account-id:project-id",
        "arn:aws:codepipeline:us-east-1:account-id:project-id-Pipeline",
        "arn:aws:execute-api:us-east-1:account-id:7rlst5mrgi",
        "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CloudFormation",
        "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CloudWatchEventRule",
        "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CodeBuild",
        "arn:aws:iam::account-id:role/CodeStarWorker-project-id-CodePipeline",
        "arn:aws:iam::account-id:role/CodeStarWorker-project-id-Lambda",
        "arn:aws:lambda:us-east-1:account-id:function:awscodestar-project-id-lambda-
GetHelloWorld-KFKTXYNH9573",
        "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-app",
        "arn:aws:s3:::aws-codestar-us-east-1-account-id-project-id-pipe"
      ]
    },
    {
      "Sid": "3",
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "config:Describe*",
        "config:Get*",
        "config:List*",
        "config:Put*",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Listing Resources for a Project

In this example, you want to grant a specified IAM user in your AWS account access to list the resources of an AWS CodeStar project.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codestar:ListResources",
      ],
      "Resource" : "arn:aws:codestar:us-east-2:project/my-first-projec"
    }
  ]
}
```

Using the AWS CodeStar Console

No specific permissions are required to access the AWS CodeStar console, but you can't do anything useful unless you have either the `AWSCodeStarFullAccess` policy or one of the AWS CodeStar project-level role: Owner, Contributor, or Viewer. For more information on `AWSCodeStarFullAccess`, see AWSCodeStarFullAccess Policy. For more information on the project-level policies, see IAM Policies for Team Members.

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

Allow Users to View Their Own Permissions

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

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

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

Updating an AWS CodeStar Project

In this example, you want to grant a specified IAM user in your AWS account access to edit the attributes of an AWS CodeStar project, such as its project description.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codestar:UpdateProject"
      ],
      "Resource" : "arn:aws:codestar:us-east-2:project/my-first-projec"
```

```
    }
  ]
}
```

Adding a Team Member to a Project

In this example, you want to grant a specified IAM user the ability to add team members to an AWS CodeStar project with the project ID *my-first-projec*, but to explicitly deny that user the ability to remove team members:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codestar:AssociateTeamMember",
      ],
      "Resource" : "arn:aws:codestar:us-east-2:project/my-first-projec"
    },
    {
      "Effect" : "Deny",
      "Action" : [
        "codestar:DisassociateTeamMember",
      ],
      "Resource" : "arn:aws:codestar:us-east-2:project/my-first-projec"
    }
      ]

  ]
}
```

Listing User Profiles Associated with an AWS Account

In this example, you allow an IAM user who has this policy attached to list all AWS CodeStar user profiles associated with an AWS account:

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

```
      "Action" : [
        "codestar:ListUserProfiles",
              ],
      "Resource" : "*"
    }
  ]
}
```

Viewing AWS CodeStar Projects Based on Tags

You can use conditions in your identity-based policy to control access to AWS CodeStar projects based on tags. This example shows how you might create a policy that allows viewing a project. However, permission is granted only if the Project tag `Owner` has the value of that user's user name. This policy also grants the permissions necessary to complete this action on the console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListProjectsInConsole",
            "Effect": "Allow",
            "Action": "codestar:ListProjects",
            "Resource": "*"
        },
        {
            "Sid": "ViewProjectIfOwner",
            "Effect": "Allow",
            "Action": "codestar:GetProject,
            "Resource": "arn:aws:codestar:*:*:project/*",
            "Condition": {
                "StringEquals": {"codestar:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to view an AWS CodeStar project, the project must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise he is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

AWS CodeStarupdates to AWS managed policies

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

Change	Description	Date
AWSCodeStarFullAccess Policy – Update the AWSCodeStarFullAccess policy	The AWS CodeStar access role policy has been updated. Outcome of the policy is the same, but cloudformation requires ListStacks in addition to DescribeStacks, which is already required.	March 24, 2023
AWSCodeStarServiceRole Policy – Update the AWSCodeStarServiceRole policy	The policy for the AWS CodeStar service role has been updated to correct redundant actions in the policy statement. The service role policy allows the AWS CodeStar service to perform actions on your behalf.	September 23, 2021
AWS CodeStar started tracking changes	AWS CodeStar started tracking changes for its AWS managed policies.	September 23, 2021

Troubleshooting AWS CodeStar Identity and Access

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

Topics

- I am not authorized to perform an action in AWS CodeStar
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS CodeStar resources

I am not authorized to perform an action in AWS CodeStar

If the AWS Management Console tells you that you're not authorized to perform an action, contact your administrator for assistance. Your administrator is the person who provided you with your sign-in credentials.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`widget`* but does not have `codestar:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codestar:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `codestar:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

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

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS CodeStar. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

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

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

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

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

To learn more, consult the following:

- To learn whether AWS CodeStar supports these features, see How AWS CodeStar Works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Logging AWS CodeStar API Calls with AWS CloudTrail

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

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS CodeStar Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS CodeStar, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS CodeStar, create a trail. 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. You can configure other AWS services to further analyze and act on the event data collected in CloudTrail logs. For more information, see the following:

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

All AWS CodeStar actions are logged by CloudTrail and are documented in the AWS CodeStar API Reference. For example, calls to the `DescribeProject`, `UpdateProject`, and `AssociateTeamMember` actions generate entries in the CloudTrail log files.

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

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

For more information, see the CloudTrail userIdentity Element.

Understanding AWS CodeStar Log File Entries

CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action,

request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates a `CreateProject` operation being called in AWS CodeStar:

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAJLIN2OF3UBEXAMPLE:role-name",
    "arn": "arn:aws:sts::account-ID:assumed-role/role-name/role-session-name",
    "accountId": "account-ID",
    "accessKeyId": "ASIAJ44LFQS5XEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2017-06-04T23:56:57Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAJLIN2OF3UBEXAMPLE",
        "arn": "arn:aws:iam::account-ID:role/service-role/role-name",
        "accountId": "account-ID",
        "userName": "role-name"
      }
    },
    "invokedBy": "codestar.amazonaws.com"
  },
  "eventTime": "2017-06-04T23:56:57Z",
  "eventSource": "codestar.amazonaws.com",
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "codestar.amazonaws.com",
  "userAgent": "codestar.amazonaws.com",
  "requestParameters": {
    "clientRequestToken": "arn:aws:cloudformation:region-ID:account-ID:stack/stack-
name/additional-ID",
    "id": "project-ID",
    "stackId": "arn:aws:cloudformation:region-ID:account-ID:stack/stack-
name/additional-ID",
    "description": "AWS CodeStar created project",
    "name": "project-name",
```

```
    "projectTemplateId": "arn:aws:codestar:region-ID::project-template/project-
template-name"
  },
  "responseElements": {
    "projectTemplateId": "arn:aws:codestar:region-ID::project-template/project-
template-name",
    "arn": "arn:aws:codestar:us-east-1:account-ID:project/project-ID",
    "clientRequestToken": "arn:aws:cloudformation:region-ID:account-ID:stack/stack-
name/additional-ID",
    "id": "project-ID"
  },
  "requestID": "7d7556d0-4981-11e7-a3bc-dd5daEXAMPLE",
  "eventID": "6b0d6e28-7a1e-4a73-981b-c8fdbEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account-ID"
}
```

Compliance Validation for AWS CodeStar

AWS CodeStar is not in scope of any AWS compliance programs.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

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

Resilience in AWS CodeStar

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

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

Infrastructure Security in AWS CodeStar

As a managed service, AWS CodeStar 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 CodeStar 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.

By default, AWS CodeStar does not isolate service traffic. Projects created using AWS CodeStar are open to the public internet unless you manually modify the access settings through Amazon EC2, API Gateway or Elastic Beanstalk. This is intentional. You can modify the access settings in Amazon EC2, API Gateway, or Elastic Beanstalk to the degree you want, including preventing all internet access.

AWS CodeStar does not provide support for VPC endpoints (AWS PrivateLink) by default, but you can configure that support directly on the project resources.

Limits in AWS CodeStar

The following table describes limits in AWS CodeStar. AWS CodeStar depends on other AWS services for project resources. Some of those service limits can be changed. For information about limits that can be changed, see AWS Service Limits.

Number of projects	Maximum of 333 projects in an AWS account. Actual limit varies, depending on the level of other service dependencies (for example, the maximum number of pipelines in CodePipeline allowed for your AWS account).
Number of AWS CodeStar projects to which an IAM user can belong	Maximum of 10 per individual IAM user.
Project IDs	Project IDs must be unique in an AWS account. Project IDs must be at least 2 characters and cannot exceed 15 characters. Allowed characters include: Letters a through z, inclusive. Numbers 0 through 9, inclusive. The special character - (minus sign). Any other characters, such as capital letters, spaces, . (period), @ (at sign), or _ (underscore), are not allowed.
Project names	Project names cannot exceed 100 characters in length, and cannot begin or end with an empty space.
Project descriptions	Any combination of characters between 0 and 1,024 characters in length. Project descriptions are optional.

Team members in an AWS CodeStar project	100
Display name in a user profile	Any combination of characters between 1 and 100 characters in length. Display names must include at least one character. That character cannot be a space. Display names cannot begin or end with a space.
Email address in a user profile	The email address must include an @ and end in a valid domain extension.
Federated access, root account access, or temporary access to AWS CodeStar	AWS CodeStar supports federated users and use of temporary access credentials. Using AWS CodeStar with a root account is not recommended.
IAM roles	A maximum of 5,120 characters in any managed policy that is attached to an IAM role.

Troubleshooting AWS CodeStar

The following information might help you troubleshoot common issues in AWS CodeStar.

Topics

- Project creation failure: A project was not created
- Project creation: I see an error when I try to edit Amazon EC2 configuration when creating a project
- Project deletion: An AWS CodeStar project was deleted, but resources still exist
- Team management failure: An IAM user could not be added to a team in an AWS CodeStar project
- Access failure: A federated user cannot access an AWS CodeStar project
- Access failure: A federated user cannot access or create an AWS Cloud9 environment
- Access failure: A federated user can create an AWS CodeStar project, but cannot view project resources
- Service role issue: The service role could not be created
- Service role issue: The service role is not valid or missing
- Project role issue: AWS Elastic Beanstalk health status checks fail for instances in an AWS CodeStar project
- Project role issue: A project role is not valid or missing
- Project extensions: Can't connect to JIRA
- GitHub: Can't access a repository's commit history, issues, or code
- AWS CloudFormation: Stack Creation Rolled Back for Missing Permissions
- AWS CloudFormation is not authorized to perform iam:PassRole on Lambda execution role
- Unable to create the connection for a GitHub repository

Project creation failure: A project was not created

Problem: When you try to create a project, you see a message that says the creation failed.

Possible fixes: The most common reasons for failure are:

- A project with that ID already exists in your AWS account, possibly in a different AWS Region.

- The IAM user you used to sign in to the AWS Management Console doesn't have the permissions required to create a project.
- The AWS CodeStar service role is missing one or more required permissions.
- You have reached the maximum limit for one or more resources for a project (such as the limit on customer managed policies in IAM, Amazon S3 buckets, or pipelines in CodePipeline).

Before you create a project, verify that you have the `AWSCodeStarFullAccess` policy applied to your IAM user. For more information, see AWSCodeStarFullAccess Policy.

When you create a project, make sure that the ID is unique and meets the AWS CodeStar requirements. Be sure you selected the **AWS CodeStar would like permission to administer AWS resources on your behalf** check box.

To troubleshoot other issues, open the AWS CloudFormation console, choose the stack for the project you tried to create, and choose the **Events** tab. There might be more than one stack for a project. The stack names start with `awscodestar-` and are followed by the project ID. Stacks might be under the **Deleted** filter view. Review any failure messages in the stack events and correct the issue listed as the cause of those failures.

Project creation: I see an error when I try to edit Amazon EC2 configuration when creating a project

Problem: When you edit the Amazon EC2 configuration options during project creation, you see an error message or grayed-out option, and cannot continue with project creation.

Possible fixes: The most common reasons for an error message are:

- The VPC in the AWS CodeStar project template (either the default VPC or the one used when the Amazon EC2 configuration was edited) has dedicated instance tenancy, and the instance type is not supported for dedicated instances. Choose a different instance type or a different Amazon VPC.
- Your AWS account has no Amazon VPCs. You might have deleted the default VPC and not created any others. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/, choose **Your VPCs**, and make sure that you have at least one VPC configured. If not, create one. For more information, see Amazon Virtual Private Cloud Overview in the *Amazon VPC Getting Started Guide*.

- The Amazon VPC does not have any subnets. Choose a different VPC or create a subnet for the VPC. For more information, see VPC and Subnet Basics.

Project deletion: An AWS CodeStar project was deleted, but resources still exist

Problem: An AWS CodeStar project was deleted, but resources created for that project still exist. By default, AWS CodeStar deletes project resources when the project is deleted. Some resources, such as Amazon S3 buckets, are retained even if the user selects the **Delete resources** check box, because the buckets might contain data.

Possible fixes: Open the AWS CloudFormation console and find one or more of the AWS CloudFormation stacks used to create the project. The stack names start with `awscodestar-` and are followed by the project ID. The stacks might be under the **Deleted** filter view. Review the events associated with the stack to discover the resources created for the project. Open the console for each of those resources in the AWS Region where you created the AWS CodeStar project, and then manually delete the resources.

Project resources that might remain include:

- One or more project buckets in Amazon S3. Unlike other project resources, project buckets in Amazon S3 are not deleted when the **Delete associated AWS resources along with AWS CodeStar project** check box is selected.

  Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- A source repository for your project in CodeCommit.

  Open the CodeCommit console at https://console.aws.amazon.com/codecommit/.
- A pipeline for your project in CodePipeline.

  Open the CodePipeline console at https://console.aws.amazon.com/codepipeline/.
- An application and associated deployment groups in CodeDeploy.

  Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
- An application and associated environments in AWS Elastic Beanstalk.

  Open the Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.

- A function in AWS Lambda.

  Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
- One or more APIs in API Gateway.

  Open the API Gateway console at https://console.aws.amazon.com/apigateway/.
- One or more IAM policies or roles in IAM.

  Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
- An instance in Amazon EC2.

  Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
- One or more development environments in AWS Cloud9.

  To view, access, and manage development environments, open the AWS Cloud9 console at https://console.aws.amazon.com/cloud9/.

If your project uses resources outside of AWS (for example, a GitHub repository or issues in Atlassian JIRA), those resources are not deleted, even if the **Delete associated AWS resources along with CodeStar project** box is selected.

Team management failure: An IAM user could not be added to a team in an AWS CodeStar project

Problem: When you try to add a user to a project, you see an error message that says that the addition failed.

Possible fixes: The most common reason for this error is that the user has reached the limit of managed policies that can be applied to a user in IAM. You might also receive this error if you do not have the owner role in the AWS CodeStar project where you tried to add the user, or if the IAM user does not exist or was deleted.

Make sure you are signed in as an user who is an owner in that AWS CodeStar project. For more information, see Add Team Members to an AWS CodeStar Project .

To troubleshoot other issues, open the IAM console, choose the user you tried to add, and check how many managed policies are applied to that IAM user.

For more information, see Limitations on IAM Entities and Objects. For limits that can be changed, see AWS Service Limits.

Access failure: A federated user cannot access an AWS CodeStar project

Problem: A federated user is unable to see projects in the AWS CodeStar console.

Possible fixes: If you are signed in as a federated user, make sure you have the appropriate managed policy attached to the role you assume to sign in. For more information, see Attach Your Project's AWS CodeStar Viewer/Contributor/Owner Managed Policy to the Federated User's Role.

Add federated users to your AWS Cloud9 environment by manually attaching policies. See Attach an AWS Cloud9 Managed Policy to the Federated User's Role.

Access failure: A federated user cannot access or create an AWS Cloud9 environment

Problem: A federated user is unable to see or create an AWS Cloud9 environment in the AWS Cloud9 console.

Possible fixes: If you are signed in as a federated user, make sure you have the appropriate managed policy attached to the federated user's role.

You add federated users to your AWS Cloud9 environment by manually attaching policies to the federated user's role. See Attach an AWS Cloud9 Managed Policy to the Federated User's Role.

Access failure: A federated user can create an AWS CodeStar project, but cannot view project resources

Problem: A federated user was able to create a project, but cannot view project resources, such as the project pipeline.

Possible fixes: If you have attached the **`AWSCodeStarFullAccess`** managed policy, you have permissions to create a project in AWS CodeStar. However, to access all project resources, you must attach the owner managed policy.

After AWS CodeStar creates the project resources, project permissions to all project resources are available in the owner, contributer, and viewer managed policies. To access all of the resources, you must manually attach the owner policy to your role. See Step 3: Configure the User's IAM Permissions.

Service role issue: The service role could not be created

Problem: When you try to create a project in AWS CodeStar, you see a message that prompts you to create the service role. When you choose the option to create it, you see an error.

Possible fixes: The most common reason for this error is that you are signed in to AWS with an account that does not have sufficient permissions to create the service role. To create the AWS CodeStar service role (`aws-codestar-service-role`), you must be signed in as an administrative user or with a root account. Sign out of the console, and sign in with an IAM user that has the `AdministratorAccess` managed policy applied.

Service role issue: The service role is not valid or missing

Problem: When you open the AWS CodeStar console, you see a message that says the AWS CodeStar service role is missing or not valid.

Possible fixes: The most common reason for this error is that an administrative user edited or deleted the service role (`aws-codestar-service-role`). If the service role was deleted, you are prompted to create it. You must be signed in as an administrative user or with a root account to create the role. If the role was edited, it is no longer valid. Sign in to the IAM console as an administrative user, find the service role in the list of roles, and delete it. Switch to the AWS CodeStar console and follow the instructions to create the service role.

Project role issue: AWS Elastic Beanstalk health status checks fail for instances in an AWS CodeStar project

Problem: If you created an AWS CodeStar project that includes Elastic Beanstalk before September 22, 2017, Elastic Beanstalk health status checks might fail. If you have not changed the Elastic Beanstalk configuration since you created the project, the health status check fails and reports a gray state. Despite the health check failure, your application should still run as expected. If you changed the Elastic Beanstalk configuration since you created the project, the health status check fails, and your application might not run correctly.

Fix: One or more IAM roles are missing required IAM policy statements. Add the missing policies to the affected roles in your AWS account.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   (If you cannot do this, see your AWS account administrator for assistance.)
2. In the navigation pane, choose **Roles**.
3. In the list of roles, choose **CodeStarWorker-*Project-ID*-EB**, where *Project-ID* is the ID of one of the affected projects. (If you cannot easily find a role in the list, enter some or all of the role's name in the **Search** box.)
4. On the **Permissions** tab, choose **Attach Policy**.
5. In the list of policies, select **AWSElasticBeanstalkEnhancedHealth** and **AWSElasticBeanstalkService**. (If you cannot easily find a policy in the list, enter some or all of the policy's name in the search box.)
6. Choose **Attach Policy**.
7. Repeat steps 3 through 6 for each affected role that has a name following the pattern **CodeStarWorker-*Project-ID*-EB**.

Project role issue: A project role is not valid or missing

Problem: When you try to add a user to a project, you see an error message that says the addition failed because the policy for a project role is either missing or not valid.

Possible fixes: The most common reason for this error is that one or more project policies was edited in or deleted from IAM. Project policies are unique to AWS CodeStar projects and cannot be recreated. The project cannot be used. Create a project in AWS CodeStar, and then migrate data to the new project. Clone project code from the unusable project's repository, and push that code to the new project's repository. Copy team wiki information from the old project to the new project. Add users to the new project. When you are sure you have migrated all data and settings, delete the unusable project.

Project extensions: Can't connect to JIRA

Problem: When you use the Atlassian JIRA extension to try to connect an AWS CodeStar project to a JIRA instance, the following message is displayed: "The URL is not a valid JIRA URL. Verify that the URL is correct."

Possible fixes:

- Make sure the JIRA URL is correct, and then try connecting again.
- Your self-hosted JIRA instance might not be accessible from the public internet. Contact your network administrator to make sure your JIRA instance can be accessed from the public internet, and then try connecting again.

GitHub: Can't access a repository's commit history, issues, or code

Problem: In the dashboard for a project that stores its code in GitHub, the **Commit history** and **GitHub Issues** tiles display a connection error, or choosing **Open in GitHub** or **Create issue** in these tiles displays an error.

Possible causes:

- The AWS CodeStar project might no longer have access to the GitHub repository.
- The repository might have been deleted or renamed in GitHub.

AWS CloudFormation: Stack Creation Rolled Back for Missing Permissions

After you add a resource to the `template.yml` file, view the AWS CloudFormation stack update for any error messages. The stack update fails if certain criteria are not met (for example, when required resource permissions are missing).

> **Note**
>
> As of May 2, 2019, we have updated the AWS CloudFormation worker role policy for all existing projects. This update reduces the scope of access permissions granted to your project pipeline for improved security in your projects.

To troubleshoot, view the failure status in the AWS CodeStar dashboard view for your project's pipeline.

Next, choose the **CloudFormation** link in your pipeline's Deploy stage to troubleshoot the failure in the AWS CloudFormation console. To view stack creation details, expand the **Events** list for your project and view any failure messages. The message indicates which permission is missing. Correct the AWS CloudFormation worker role policy and then execute your pipeline again.

AWS CloudFormation is not authorized to perform iam:PassRole on Lambda execution role

If you have a project created before December 6, 2018 PDT that creates Lambda functions, you might see a AWS CloudFormation error like this:

```
User: arn:aws:sts::id:assumed-role/CodeStarWorker-project-id-CloudFormation/
AWSCloudFormation is not authorized to perform: iam:PassRole on resource:
 arn:aws:iam::id:role/CodeStarWorker-project-id-Lambda (Service: AWSLambdaInternal;
 Status Code: 403; Error Code: AccessDeniedException; Request ID: id)
```

This error occurs because your AWS CloudFormation worker role does not have permission to pass a role for provisioning your new Lambda function.

To fix this error, you will need to update your AWS CloudFormation worker role policy with the following snippet.

```
{
        "Action":[ "iam:PassRole" ],
        "Resource": [
           "arn:aws:iam::account-id:role/CodeStarWorker-project-id-Lambda",
```

```
            ],

        "Effect": "Allow"
}
```

After you update the policy, execute your pipeline again.

Alternatively, you can use a custom role for your Lambda function by adding a permissions boundary to your project, as described in Add an IAM Permissions Boundary to Existing Projects

Unable to create the connection for a GitHub repository

Problem:

Because a connection to a GitHub repository uses the AWS Connector for GitHub, you need organization owner permissions or admin permissions to the repository to create the connection.

Possible fixes: For information about permission levels for a GitHub repository, see https://docs.github.com/en/free-pro-team@latest/github/setting-up-and-managing-organizations-and-teams/permission-levels-for-an-organization.

AWS CodeStar User Guide Release Notes

The following table describes the important changes in each release of the AWS CodeStar User Guide. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Access policy updates	The AWS CodeStar access role policy has been updated. Outcome of the policy is the same, but cloudformation requires ListStacks in addition to DescribeStacks, which is already required. To reference the updated policy, refer to AWSCodeStarFullAccess Policy.	March 24, 2023
Service role policy updates	The AWS CodeStar service role policy has been updated. To reference the updated policy, refer to AWSCodeStarServiceRole Policy.	September 23, 2021
Use a connection resource for projects with a GitHub source repository	When you use the console to create a project in AWS CodeStar with a GitHub repository, a connection resource is used to manage your GitHub actions. Connections use GitHub Apps, while the previous GitHub authorization used OAuth. For a tutorial that shows you how to create a project that uses a connection to GitHub, see Tutorial: Create a Project with	April 27, 2021

	a GitHub Source Repository. The tutorial also shows you how to create, review, and merge a pull request for your project source repository.	
AWS CodeStar supports AWS Cloud9 in the US West (N. California) Region	AWS CodeStar now supports using AWS Cloud9 in the US West (N. California) Region. For more information, see Setting up Cloud9.	February 16, 2021
Update documentation to reflect new console experience	On August 12, 2020 the AWS CodeStar service moved to a new user experience in the AWS console. The user guide was updated to match the new console experience.	August 12, 2020
AWS CodeStar projects can be created with the AWS CodeStar CLI	AWS CodeStar projects can be created with the CLI command. AWS CodeStar creates your project and infrastructure using source code and a toolchain template you provide. See Create a Project in AWS CodeStar (AWS CLI).	October 24, 2018

All AWS CodeStar project templates now include AWS CloudFormation file for infrastructure updates	AWS CodeStar works with AWS CloudFormation to allow you to use code to create support services and servers or serverless platforms in the cloud. The AWS CloudFormation file is now available for all AWS CodeStar project template types (templates with the Lambda, EC2, or Elastic Beanstalk compute platform). The file is stored in `template.yml` in your source repository. You can view and modify the file to add resources to your project. See Project Templates.	August 3, 2018
AWS CodeStar User Guide update notifications now available through RSS	The HTML version of the AWS CodeStar User Guide now supports an RSS feed of updates that are documented in the Documentation Update Release Notes page. The RSS feed includes updates made after June 30, 2018 and later. Previously announced updates are still available in the Documentation Update Release Notes page. Use the RSS button in the top menu panel to subscribe to the feed.	June 30, 2018

The following table describes the important changes in each release of the AWS CodeStar User Guide before June 30, 2018.

Change	Description	Date Changed
The AWS CodeStar User Guide is now available on GitHub	This guide is now available on GitHub. You can also use GitHub to submit feedback and change requests for this guide's content. For more information, choose the **Edit on GitHub** icon in the guide's navigation bar, or see the awsdocs/aws-codestar-user-guide repository on the GitHub website.	February 22, 2018
AWS CodeStar is now available in Asia Pacific (Seoul)	AWS CodeStar is now available in the Asia Pacific (Seoul) region. For more information, see AWS CodeStar in the *Amazon Web Services General Reference*.	February 14, 2018
AWS CodeStar is now available in Asia Pacific (Tokyo) and Canada (Central)	AWS CodeStar is now available in the Asia Pacific (Tokyo) and Canada (Central) regions. For more information, see AWS CodeStar in the *Amazon Web Services General Reference*.	December 20, 2017
AWS CodeStar now supports AWS Cloud9	AWS CodeStar now supports using AWS Cloud9, a web browser-based online IDE, to work with project code. For more information, see Use AWS Cloud9 with AWS CodeStar. For a list of supported AWS Regions, see AWS Cloud9 in the *Amazon Web Services General Reference*.	November 30, 2017
AWS CodeStar now supports GitHub	AWS CodeStar now supports storing project code in GitHub. For more information, see Create a Project.	October 12, 2017
AWS CodeStar now available in US West (N. Californi a) and Europe (London)	AWS CodeStar is now available in the US West (N. California) and Europe (London) regions. For more information, see AWS CodeStar in the *Amazon Web Services General Reference*.	August 17, 2017

Change	Description	Date Changed
AWS CodeStar now available in Asia Pacific (Sydney), Asia Pacific (Singapore), and Europe (Frankfurt)	AWS CodeStar is now available in the Asia Pacific (Sydney), Asia Pacific (Singapore), and Europe (Frankfurt) regions. For more information, see AWS CodeStar in the *Amazon Web Services General Reference*.	July 25, 2017
AWS CloudTrail now supports AWS CodeStar	AWS CodeStar is now integrated with CloudTrail, a service that captures API calls made by or on behalf of AWS CodeStar in your AWS account and delivers the log files to an Amazon S3 bucket you specify. For more information, see Logging AWS CodeStar API Calls with AWS CloudTrail.	June 14, 2017
Initial release	This is the first release of the *AWS CodeStar User Guide*.	April 19, 2017

AWS Glossary

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