



User Guide

AWS Toolkit for JetBrains



AWS Toolkit for JetBrains: 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

AWS Toolkit for JetBrains	1
What the AWS Toolkit for JetBrains includes	1
Working with the AWS Toolkit for JetBrains	2
Related information	2
Related videos	2
Related webpages	3
Questions and help	3
Report a bug with the AWS Toolkit or make a feature request	3
Contribute to the AWS Toolkit	4
Amazon Q Developer and Amazon CodeWhisperer	5
Download the Toolkit	6
Downloading the Toolkit from the JetBrains Marketplace	6
Additional IDE Toolkits from AWS	6
Getting started	7
Installing the AWS Toolkit	7
Installing the AWS Toolkit from your JetBrains IDE	8
Installing custom builds and releases	8
Removing AWS Toolkit for JetBrains EAP and custom repository references	9
Navigation	9
Viewing the toolkit from JetBrains	10
The AWS Explorer	10
Connecting to AWS	12
Connecting to AWS	13
Prerequisites	14
Opening the Sign In panel	14
Connecting to AWS from the AWS Toolkit for JetBrains	15
Authentication for Amazon CodeCatalyst	17
AWS Regions	18
Viewing the current AWS Region	18
Changing AWS Regions	18
HTTP proxy setup	19
Authentication and access	20
AWS IAM Identity Center	20
IAM credentials	20

Prerequisites	14
Creating a shared credentials file from the AWS Toolkit for JetBrains	22
Configuring shared credentials	22
AWS Builder ID	23
Working with AWS services	24
Experimental features	24
AWS App Runner	25
Prerequisites	26
Pricing	28
Creating App Runner services	29
Managing App Runner services	31
Amazon CodeCatalyst	34
What is Amazon CodeCatalyst?	34
Getting Started with CodeCatalyst	35
Working with CodeCatalyst	37
AWS CloudFormation	41
Viewing event logs for a stack	42
Deleting a stack	44
Amazon CloudWatch Logs	45
Viewing CloudWatch log groups and log streams	45
Working with CloudWatch log events	48
Working with CloudWatch Logs Insights	51
Amazon DynamoDB	53
Working with Amazon DynamoDB	53
Working with DynamoDB tables	54
Amazon ECS	56
Amazon ECS Exec	56
Amazon EventBridge	59
Working with Amazon EventBridge schemas	59
AWS Lambda	62
Lambda Runtimes	63
Creating a function	64
Running (invoking) or debugging a local function	66
Running (invoking) a remote function	68
Changing (updating) function settings	69
Deleting a function	71

Amazon RDS	72
Prerequisites for accessing Amazon RDS databases	73
Connecting to an Amazon RDS database	75
Amazon Redshift	81
Prerequisites for accessing Amazon Redshift clusters	81
Connecting to an Amazon Redshift cluster	83
Amazon S3	88
Working with Amazon S3 buckets	88
Working with Amazon S3 objects	90
AWS Serverless	93
Creating an application	93
Syncing an application	98
Changing (updating) application settings	101
Deleting an application	103
Amazon SQS	104
Amazon SQS queues	104
Working with Lambda	106
Working with Amazon SNS	107
Resources	107
IAM permissions for accessing resources	108
Adding and interacting with existing resources	109
Creating and updating resources	110
User interface reference	113
AWS Explorer	113
Create Function dialog box	116
Deploy Serverless Application dialog box	118
New Project dialog box	120
New Project dialog box (IntelliJ IDEA, PyCharm, and WebStorm)	121
New Project dialog box (JetBrains Rider)	122
Run/Debug Configurations dialog box	124
Run/Debug Configurations (local)	124
Run/Debug Configurations (remote)	131
Edit configuration (Amazon ECS cluster)	134
Update Code dialog box	140
Update Configuration dialog box	141
Troubleshooting	144

Troubleshooting best practices	144
Unable to sign out of AWS IAM Identity Center	144
Amazon Q Developer is unavailable for JetBrains 2023.2.0 releases	145
Amazon Q Developer Code Transformation fails to start at sign in	145
Security	146
Data protection	146
Identity and Access Management	147
Audience	148
Authenticating with identities	148
Managing access using policies	152
How AWS services work with IAM	154
Troubleshooting AWS identity and access	154
Compliance Validation	156
Resilience	157
Infrastructure Security	158
Document history	159

AWS Toolkit for JetBrains

The AWS Toolkit for JetBrains is an open source plugin for the integrated development environments (IDEs) from JetBrains. The toolkit makes it easier for you to develop, debug, and deploy serverless applications with Amazon Web Services (AWS) by making your AWS resources available from your JetBrains IDE.

Topics

- [What the AWS Toolkit for JetBrains includes](#)
- [Working with the AWS Toolkit for JetBrains](#)
- [Related information](#)

What the AWS Toolkit for JetBrains includes

AWS Toolkit for JetBrains includes the following specific toolkits:

- AWS Toolkit for [CLion](#) (for C & C++ development)
- AWS Toolkit for [GoLand](#) (for Go development)
- AWS Toolkit for [IntelliJ](#) (for Java development)
- AWS Toolkit for [WebStorm](#) (for Node.js development)
- AWS Toolkit for [Rider](#) (for .NET development)
- AWS Toolkit for [PhpStorm](#) (for PHP development)
- AWS Toolkit for [PyCharm](#) (for Python development)
- AWS Toolkit for [RubyMine](#) (for Ruby development)
- AWS Toolkit for [DataGrip](#) (for database management)

Note

When there are meaningful differences in functionality between the AWS Toolkits for the supported JetBrains IDEs, we note them in this guide.

You can also use the AWS Toolkit for JetBrains to work with AWS Lambda functions, AWS CloudFormation stacks, and Amazon Elastic Container Service (Amazon ECS) clusters. The AWS

Toolkit for JetBrains includes features such as AWS credentials management and AWS Region management, which simplify writing applications for AWS.

Working with the AWS Toolkit for JetBrains

You can use the AWS Toolkit for JetBrains to do the following:

- Create, deploy, update, and delete AWS Serverless Application Model (AWS SAM) applications. For more information on working with AWS SAM through the AWS Toolkit for JetBrains, see the [AWS Serverless](#) topic located in this User Guide.
- Remotely and locally create, update, run, and debug AWS Lambda functions. To learn more about working with the AWS Lambda service through the AWS Toolkit for JetBrains, see the [AWS Lambda](#) topic located in this User Guide.
- View event logs for, and delete AWS CloudFormation stacks. For additional information on working with AWS CloudFormation and the AWS Toolkit for JetBrains, see the [AWS CloudFormation](#) topic in this User Guide.
- Debug code in AWS clusters using Amazon Elastic Container Service. For more information on working with Amazon ECS with the AWS Toolkit for JetBrains, see the [Amazon Elastic Container Service](#) topic in this User Guide.
- Work with Amazon EventBridge schemas, to learn more see the [Amazon EventBridge Scheduler](#) topic in this User Guide.

Related information

Related videos

- [Announcement | Introducing the AWS Toolkit for IntelliJ IDEA](#) (16 minutes, April 2019, YouTube website)
- [Getting Started with the AWS Toolkit for JetBrains](#) (covers the AWS Toolkit for PyCharm only, 2 minutes, November 2018, YouTube website)
- [Building Serverless Applications with the AWS Toolkit for JetBrains](#) (covers the AWS Toolkit for PyCharm only, 6 minutes, November 2018, YouTube website)

Related webpages

- [The AWS Toolkit for IntelliJ is Now Generally Available](#) (March 2019, blog post, AWS website)
- [AWS Toolkit for IntelliJ – Now generally available](#) (March 2019, blog post, AWS website)
- [New – AWS Toolkits for PyCharm, IntelliJ \(Preview\)](#) (November 2018, blog post, AWS website)
- [Introducing the AWS Toolkit for PyCharm](#) (November 2018, blog post, AWS website)
- [AWS Toolkit for IntelliJ](#) (part of the AWS Toolkit for JetBrains, AWS website)
- [AWS Toolkit for PyCharm](#) (part of the AWS Toolkit for JetBrains, AWS website)
- [AWS Toolkit](#) (JetBrains website)
- [Develop on AWS with JetBrains Tools](#) (JetBrains website)
- [All Developer Tools and Products by JetBrains](#) (JetBrains website)

Questions and help

To ask questions or seek help from the AWS developer community, see the following AWS Discussion Forums:

- [C & C++ Development](#)
- [Go Development](#)
- [Java Development](#)
- [JavaScript Development](#)
- [.NET Development](#)
- [PHP Development](#)
- [Python Development](#)
- [Ruby Development](#)

(When you enter these forums, AWS might require you to sign in.)

You can also [contact us](#) directly.

Report a bug with the AWS Toolkit or make a feature request

To report a bug with the AWS Toolkit for JetBrains or to make a feature request, go to the [Issues](#) tab in the [aws/aws-toolkit-jetbrains](#) repository on the GitHub website. Choose **New issue**, and then

follow the on-screen instructions to finish making your bug report or feature request. (When you enter this website, GitHub might require you to sign in.)

Contribute to the AWS Toolkit

We greatly value your contributions to the AWS Toolkit. To begin contributing, read the [Contributing Guidelines](#) in the [aws/aws-toolkit-jetbrains](#) repository on the GitHub website. (When you enter this website, GitHub might require you to sign in.)

Amazon Q Developer in the AWS Toolkit for JetBrains

As of April 30th 2024, Amazon CodeWhisperer is now part of Amazon Q Developer, this includes inline code suggestions and Amazon Q Developer security scans. Download the [Amazon Q Developer IDE plugin](#) from the JetBrains Marketplace to get started.

For detailed information about the Amazon Q Developer service, see the [Amazon Q Developer User Guide](#). For detailed information about plans and pricing for Amazon Q, see the [Amazon Q Pricing guide](#).

Downloading the AWS Toolkit for JetBrains

You can download, install, and set up the AWS Toolkit for JetBrains through the JetBrains Marketplace in your IDE. For detailed instructions, see the [Installing the AWS Toolkit from your JetBrains](#) section in the *Getting started* topic of this User Guide.

Downloading the Toolkit from the JetBrains Marketplace

Download the AWS Toolkit for JetBrains installation files by navigating to the [JetBrains Marketplace](#) from your web browser.

Additional IDE Toolkits from AWS

In addition to the AWS Toolkit for JetBrains, AWS also offers IDE Toolkits for VS Code and Visual Studio.

AWS Toolkit for Visual Studio Code links

- Follow this link to [Download the AWS Toolkit for Visual Studio Code](#) from the VS Code Marketplace.
- To learn more about the AWS Toolkit for Visual Studio Code, see the [AWS Toolkit for Visual Studio Code](#) User Guide.

Toolkit for Visual Studio links

- Follow this link to [Download the Toolkit for Visual Studio](#) from the Visual Studio Marketplace.
- To learn more about the Toolkit for Visual Studio, see the [Toolkit for Visual Studio](#) User Guide.

Getting started with the AWS Toolkit for JetBrains

The AWS Toolkit for JetBrains makes your AWS services and resources available directly from your JetBrains integrated development environment (IDE).

To get you started, the following topics walk you through the processes of installing, setting up, and configuring the AWS Toolkit for JetBrains.

Note

To run the AWS Toolkit for JetBrains version 3.0 or later, you must also install AWS Core from the JetBrains Marketplace.

Topics

- [Installing the AWS Toolkit for JetBrains](#)
- [Installing AWS Toolkit for JetBrains Early Access Program \(EAP\) and custom builds](#)
- [Navigating the AWS Toolkit for JetBrains](#)
- [Connecting the AWS Toolkit for JetBrains to your AWS account](#)
- [Setting an AWS Region for the AWS Toolkit for JetBrains](#)
- [Setting up HTTP proxy for the AWS Toolkit for JetBrains](#)

Installing the AWS Toolkit for JetBrains

You can download, install, and set up the AWS Toolkit for JetBrains from the JetBrains Marketplace in your IDE. Alternatively, you can download the latest AWS Toolkit for JetBrains installation files by navigating to the [AWS Toolkit for JetBrains Marketplace](#) listing from your web browser.

Note

To run the AWS Toolkit for JetBrains version 3.0 or later, you must also install AWS Core from the JetBrains Marketplace.

The following sections describe how to install and set up the AWS Toolkit for JetBrains directly from your JetBrains IDE.

Installing the AWS Toolkit from your JetBrains IDE

To download and install the AWS Toolkit for JetBrains directly from your preferred JetBrains IDE, complete the following procedure.

1. From the JetBrains main menu, open your **Preferences** menu (expand **File** choose **Settings**, for Windows users).
2. From the **Preferences/Settings** menu, choose **Plugins** to open the **Plugins** menu.
3. From the **Plugins** menu navigation, choose **Marketplace** to open the JetBrains Plugin **Marketplace**.
4. In the provided search field, enter **AWS Toolkit**.
5. From the **AWS Toolkit** plugin entry, choose the green **Install** button, next to the entry title.
6. Accept the **Third-party Plugins Privacy Notice** to continue with the installation process.
7. JetBrains prompts you to restart the IDE when installation is complete.

Installing AWS Toolkit for JetBrains Early Access Program (EAP) and custom builds

Early Access Program (EAP) builds of the AWS Toolkit for JetBrains contain previews of new and experimental features.

Note

To run the AWS Toolkit for JetBrains version 3.0 or later, you must also install AWS Core from the JetBrains Marketplace.

To configure your toolkit for EAP builds, complete the following procedure:

1. From the JetBrains main menu, open your **Preferences** menu (expand **File** choose **Settings**, for Windows users).
2. From the **Preferences/Settings** menu, choose **Plugins** to open the **Plugins** menu.
3. From the **Plugins** menu navigation, expand the **Settings (Manage Repositories, Configure Proxy or Install Plugin from Disk)** icon and choose **Manage Plugin Repositories**.

4. From the **Manage Plugin Repositories** menu choose the **+ (Add)** icon and enter **https://plugins.jetbrains.com/plugins/eap/aws.toolkit** into the **EAP repository for the AWS Toolkit** field.
5. Choose **OK** to start the EAP installation.
6. JetBrains prompts you to restart the IDE when the installation is complete.

Removing AWS Toolkit for JetBrains EAP and custom repository references

It may be necessary to remove an EAP or custom repository reference in order to use a specific version of the AWS Toolkit for JetBrains. To remove a repository reference, complete the following procedure.

Note

After completing this procedure it may still be necessary to uninstall your current version of the AWS Toolkit for JetBrains before updating or installing a different version.

To remove an EAP repository reference

1. From the JetBrains main menu, open your **Preferences** menu (expand **File** choose **Settings**, for Windows users).
2. From the **Preferences/Settings** menu, choose **Plugins** to open the **Plugins** menu.
3. From the **Plugins** menu navigation, expand the **Settings (Manage Repositories, Configure Proxy or Install Plugin from Disk)** icon and choose **Manage Plugin Repositories**.
4. From the **Manage Plugin Repositories** menu choose the **- (Remove)** icon and confirm the removal.

Navigating the AWS Toolkit for JetBrains

The following topics describe the basic locations and components of the AWS Toolkit for JetBrains.

Topics

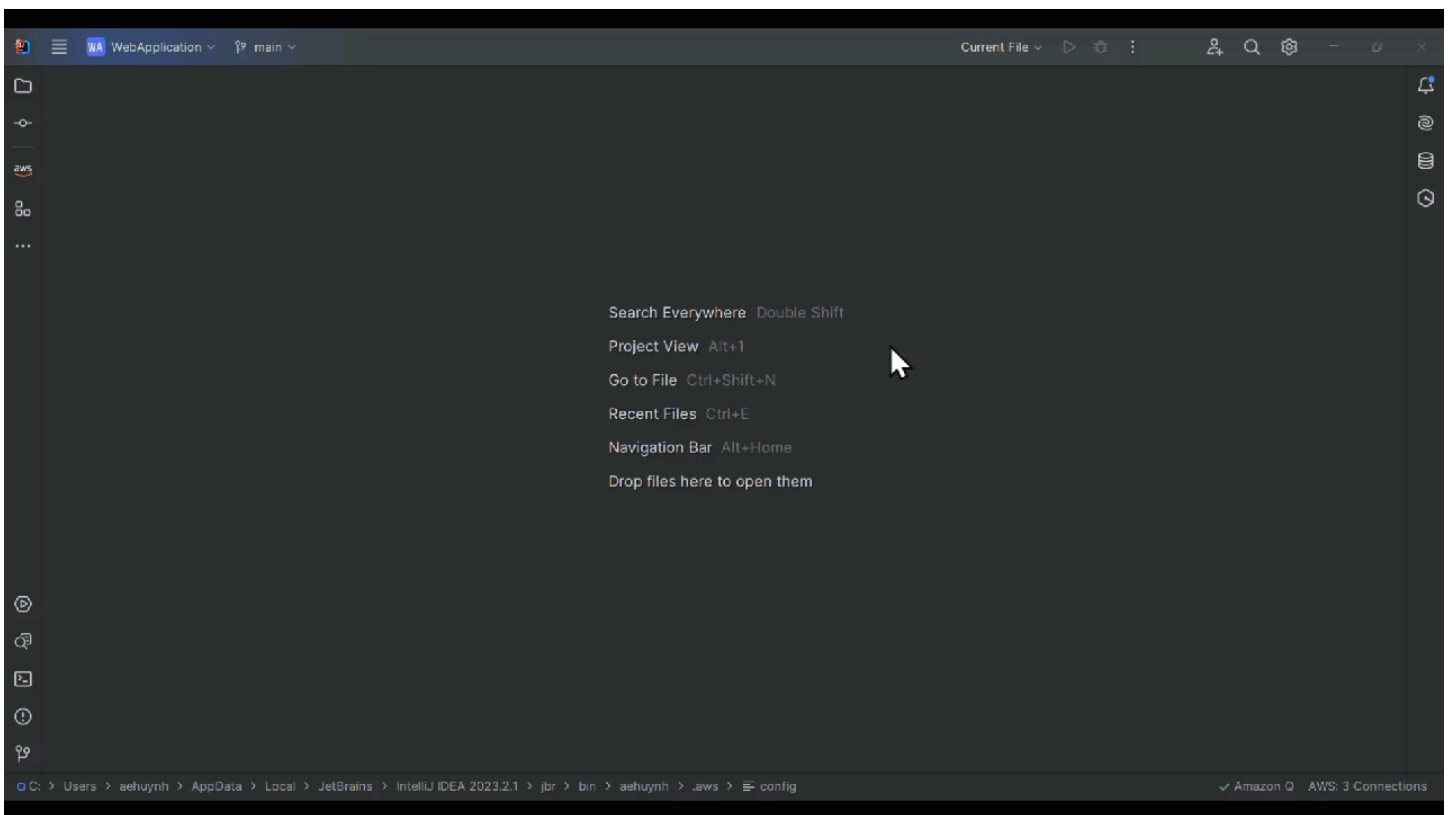
- [Viewing the toolkit from JetBrains](#)

- [The AWS Explorer](#)
- [Connecting to AWS](#)

Viewing the toolkit from JetBrains

To view the toolkit in your JetBrains IDE, complete the following steps:

1. From the JetBrains IDE, expand the **Active Toolbar** using the **Active Toolbar** icon located in the bottom left-hand corner of the JetBrains IDE.
2. From the **Active Toolbar** choose **AWS Toolkit**.
3. The AWS Toolkit for JetBrains is now open in the **Active Toolbar** window.



The AWS Explorer

Your AWS services and resources are available through the AWS Toolkit for JetBrains Explorer.

Note

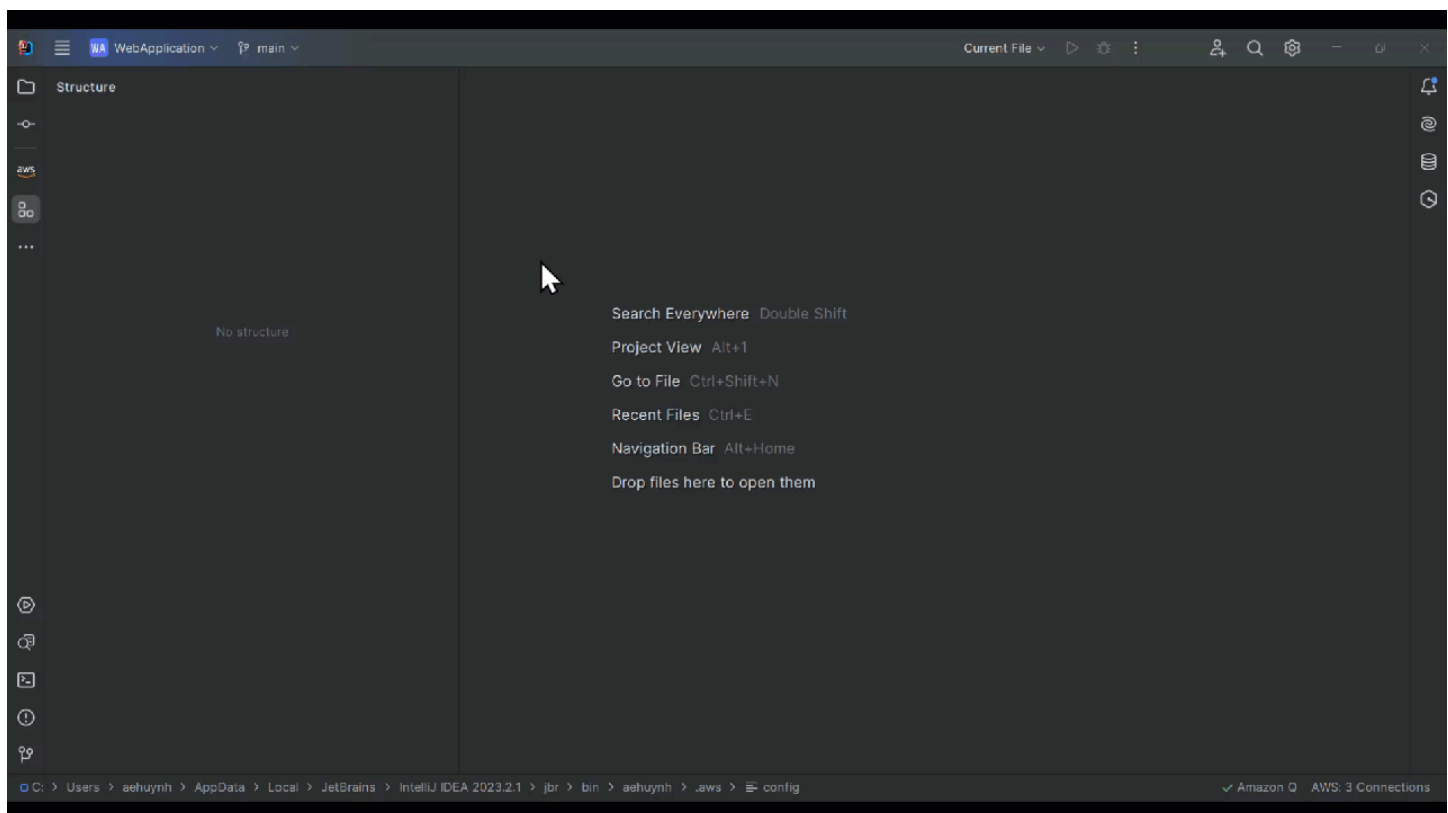
Your AWS services and resources are only visible from the AWS Explorer after you've set up authentication and connected to your AWS account.

For additional information about authentication and the AWS Toolkit for JetBrains, see the [Authentication and access](#) table of contents in this User Guide.

For additional information about connecting to your AWS account from the AWS Toolkit for JetBrains, see the [Connecting to AWS](#) topic in this User Guide.

To view your AWS services and resources from the AWS Toolkit for JetBrains Explorer:

1. From the AWS Toolkit for JetBrains choose the **Explorer** tab to view the AWS services associated with your account and region.
2. Select a service to expand a list of your resources.
3. Open the context menu for (right-click) a resource to see a list of features for modifying your resource.



Connecting to AWS

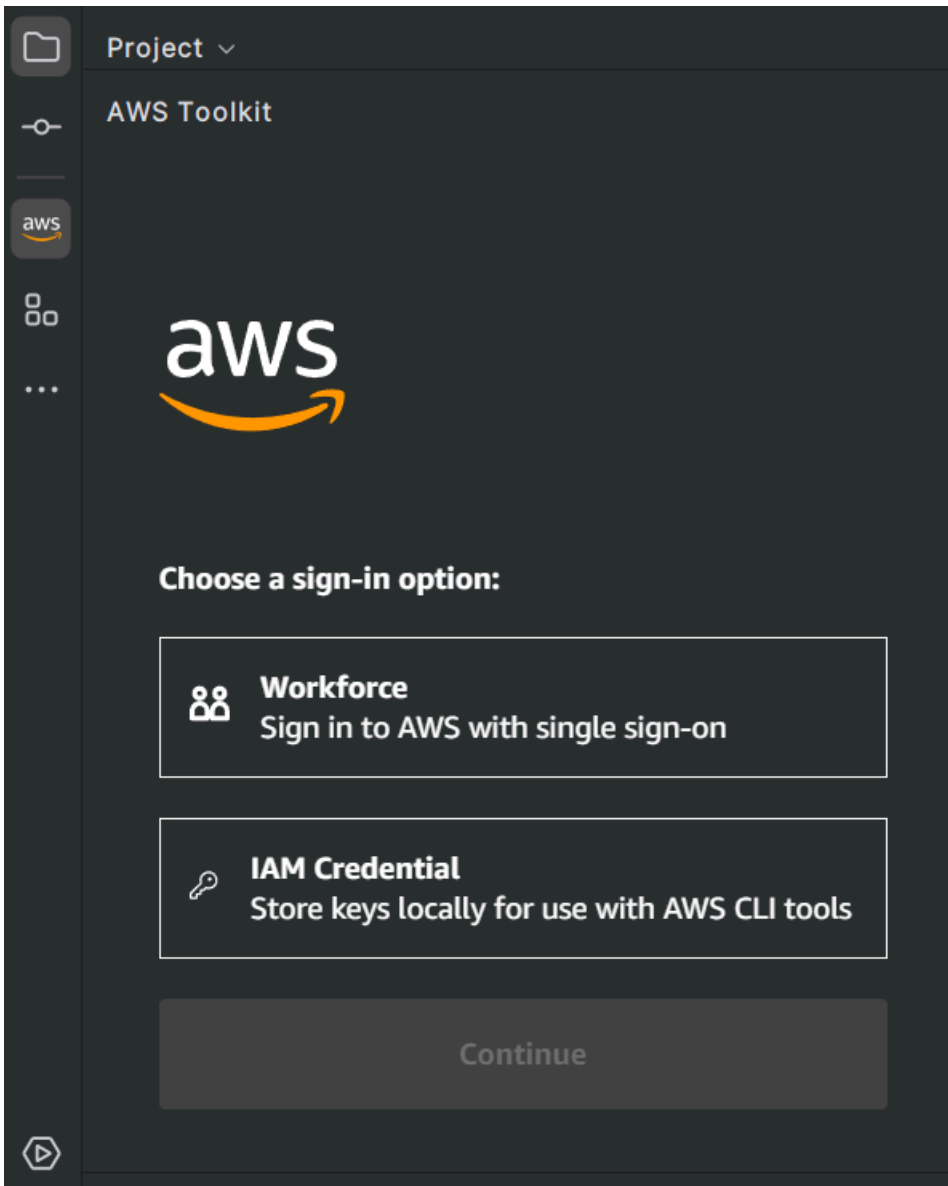
Your connection and authentication settings can be added or updated from the **AWS Toolkit Sign In** panel. The following procedure describes how to access the **AWS Toolkit Sign In** panel.

Note

If this is the first time that you're using the Toolkit or no credentials are detected on start up, then the **AWS Toolkit Sign In** pane automatically opens in the JetBrains editor. For detailed instructions on how to connect to your AWS account from the AWS Toolkit for JetBrains, see the [Connect to AWS](#) topic in this User Guide.

1. From the Toolkit, open **AWS Connection Settings** by choosing the **ellipses** icon in the connection pane.
2. From **AWS Connection Settings**, choose **Setup Authentication...** to open the **AWS Toolkit Sign In** pane.
3. From the **AWS Toolkit Sign In** panel, select your authentication method and follow the on-screen prompts.

The following is an image of the AWS Sign In panel.



Connecting the AWS Toolkit for JetBrains to your AWS account

Most Amazon Web Services (AWS) and resources are managed through an AWS account. An AWS account isn't required to use the AWS Toolkit for JetBrains, however Toolkit functions are limited without a connection.

If you've previously set up an AWS account and authentication through another AWS service (such as the AWS Command Line Interface), then the AWS Toolkit for JetBrains automatically detects your credentials and guides you through the connection process.

Prerequisites

If you're new to AWS or haven't created an account, then there are 3 main steps to connect the AWS Toolkit for JetBrains with your AWS account:

1. **Signing up for an AWS account:** You can sign up for an AWS account from the [AWS sign up](#) portal. For detailed information on setting up a new AWS account, see the [Overview](#) topic in the *AWS Setup User Guide*.
2. **Setting up authentication:** There are 3 primary methods to authenticate with your AWS account from the AWS Toolkit for JetBrains. To learn more about each of these methods, see the [Authentication and access](#) topic in this User Guide.
3. **Connecting with your AWS account from the AWS Toolkit for JetBrains:** After you've created an AWS account and set up authentication, you can connect the AWS Toolkit for JetBrains with your AWS account by completing the *Connecting to AWS from the AWS Toolkit for JetBrains* procedure, located in the following section.

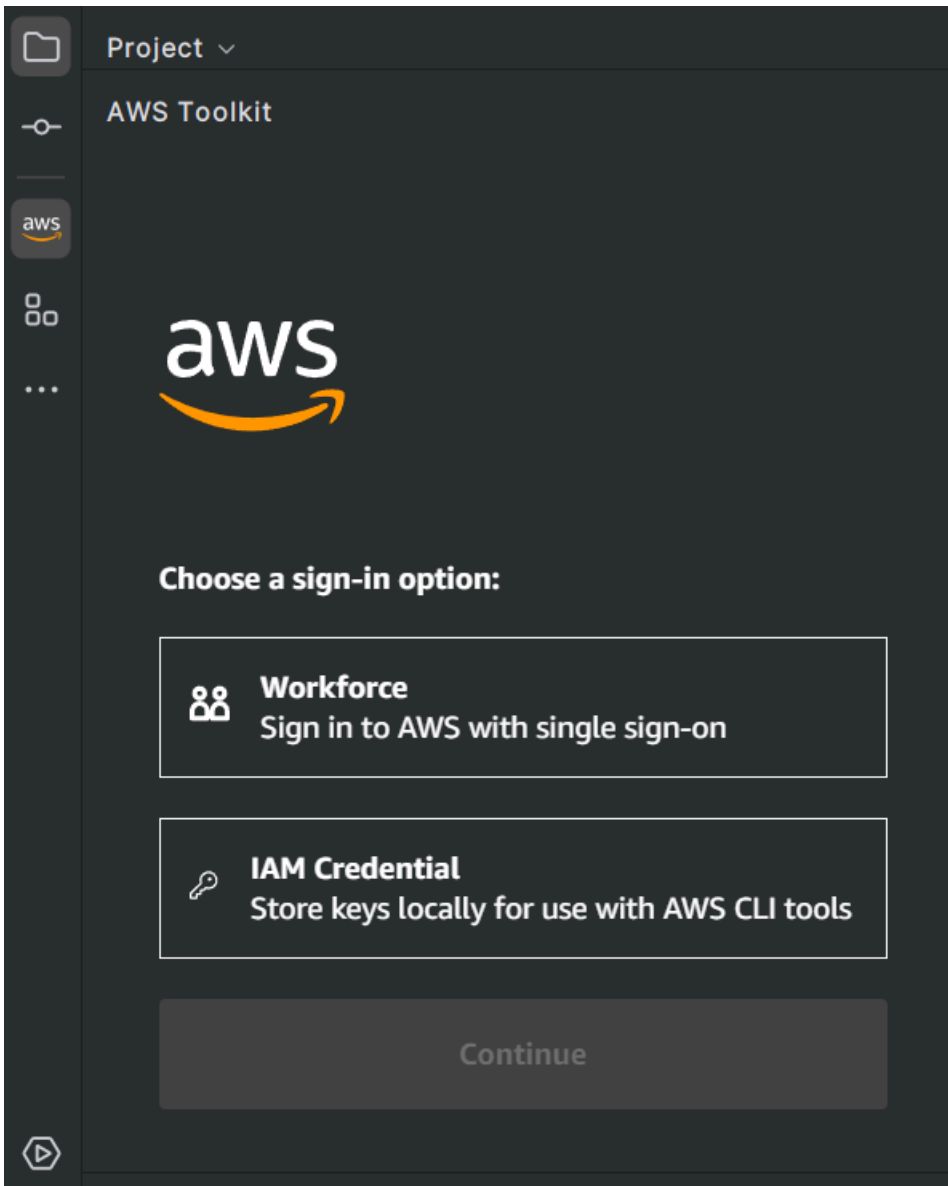
Opening the Sign In panel

If this is your first time using the AWS Toolkit for JetBrains, the **AWS Toolkit Sign In** panel automatically opens in JetBrains.

To access the AWS Sign In panel at anytime, complete the following procedure.

1. From the Toolkit, open **AWS Connection Settings** by choosing the **ellipses** icon in the connection panel.
2. From **AWS Connection Settings**, choose **Setup Authentication...** to open the **AWS Toolkit Sign In** pane.
3. From the **AWS Toolkit Sign In** panel, select your authentication method and follow the on-screen prompts.

The following is an image of the **AWS Toolkit Sign In** panel.



Connecting to AWS from the AWS Toolkit for JetBrains

The following sections describe how to authenticate and connect the AWS Toolkit for JetBrains with AWS to access the AWS Explorer and Amazon CodeCatalyst.

Note

Authentication with AWS Builder ID or IAM Identity Center launches the AWS authorization portal in your default web browser. Each time your credentials expire this process must be repeated to renew the connection between your AWS account and the AWS Toolkit for JetBrains.

Authentication for the AWS Explorer

To get started working with the AWS Explorer in the Toolkit, authenticate and connect with either your IAM credentials or IAM Identity Center credentials.

The following procedures describe how to authenticate and connect the Toolkit with your AWS account.

Authenticate and connect with IAM Identity Center

1. From the **AWS Toolkit Sign In** panel, select the **Workforce** button, then choose the **Continue** button to advance to the **Sign in with SSO** screen.
2. From the **IAM Identity Center** screen, enter your **Profile Name**, **Start URL**, and select your **Region** in the drop down, then choose the **Continue** button to open the **AWS SSO Login Required** dialog.

Note

Choose the **Region** that matches the region associated with your IAM Identity Center credentials.

3. From the **AWS SSO Login Required** dialog, choose the **Proceed To Browser** button to open the **AWS Authorization request** website in your default web browser.
4. Follow the prompts in your default web browser, you're notified when the authorization process is complete, it is safe to return to close your browser, and return to JetBrains.

Authenticate and connect with IAM Credentials

1. From the **AWS Toolkit Sign In** panel, select the **IAM Credentials** button, then choose the **Continue** button to advance to the **IAM Credentials** screen.
2. From the **IAM Credentials** screen, enter your **Profile Name**, **Access Key**, and **Secret Key**, then choose the **Continue** button to add the profile to your config file and connect the Toolkit with your AWS account.
3. The AWS Explorer updates to display your AWS services and resources when authentication is complete.

Authentication for Amazon CodeCatalyst

To get started working with CodeCatalyst from the Toolkit, authenticate and connect with your AWS Builder ID.

Complete the following steps to authenticate with your AWS account from the Toolkit, with your existing AWS Builder ID credentials.

Authenticate and connect with an AWS Builder ID

1. From the **AWS Toolkit Sign In** panel, select the **Workforce** button, then choose the **Continue** button to advance to the **Sign in with SSO** screen.
2. From the **Sign in with SSO** screen, choose the **Skip to sign-in** link to open the **AWS Authorization request** website in your default web browser.
3. Follow the prompts in your default web browser, you're notified when the authorization process is complete, it is safe to return to close your browser, and return to JetBrains.

Authenticate and connect with IAM Identity Center

1. From the **AWS Toolkit Sign In** panel, select the **Workforce** button, then choose the **Continue** button to advance to the **Sign in with SSO** screen.
2. From the **IAM Identity Center** screen, enter your **Profile Name**, **Start URL**, and select your **Region** in the drop down, then choose the **Continue** button to open the **AWS SSO Login Required** dialog.

Note

Choose the **Region** that matches the region associated with your IAM Identity Center credentials.

3. From the **AWS SSO Login Required** dialog, choose the **Proceed To Browser** button to open the **AWS Authorization request** website in your default web browser.
4. Follow the prompts in your default web browser, you're notified when the authorization process is complete, it is safe to return to close your browser, and return to JetBrains.

Setting an AWS Region for the AWS Toolkit for JetBrains

An AWS Region specifies where your AWS resources are managed. Your default AWS Region is detected when you connect to your AWS account from the AWS Toolkit for JetBrains and automatically displays in the AWS Explorer.

The following sections describe how to view and change your Region from the AWS Toolkit for JetBrains Explorer.

Viewing the current AWS Region

To check which AWS Region is currently selected, complete the following steps.

1. From the AWS Explorer, choose the **Settings** icon to open the **Show Options Menu**.
2. From the **Show Options Menu**, expand **AWS connection Settings** to display a list of AWS Regions that are available for your account.
3. Your current AWS Region displays a **check mark** icon, next to the region name.

Changing AWS Regions

To change your current AWS Region complete the following steps.

1. From the AWS Explorer, choose the **Settings** icon to open the **Show Options Menu**.
2. From the **Show Options Menu**, expand **AWS connection Settings** to display a list of AWS Regions.
3. Choose the AWS Region you want to connect to, from the list.

Note

If you don't see the region you want to connect to, choose **All Regions** to open a complete list of all AWS regions.

Setting up HTTP proxy for the AWS Toolkit for JetBrains

Setting an HTTP Proxy for the AWS Toolkit for JetBrains is handled through your JetBrains integrated development environment (IDE). To learn more about how to set an HTTP proxy, choose your JetBrains IDE from the following list.

- **CLion** – See [Configure HTTP proxy](#) on the CLion help website.
- **GoLand** – See [HTTP Proxy](#) on the GoLand help website.
- **IntelliJ IDEA** – See [HTTP Proxy](#) on the IntelliJ IDEA help website.
- **WebStorm** – See [HTTP Proxy](#) on the WebStorm help website.
- **JetBrains Rider** – See [Configure HTTP Proxy](#) on the JetBrains Rider help website.
- **PhpStorm** – See [HTTP Proxy](#) on the PhpStorm help website.
- **PyCharm** – See [HTTP Proxy](#) on the PyCharm help website.
- **RubyMine** – See [HTTP Proxy](#) on the RubyMine help website.

Authentication and access for the AWS Toolkit for JetBrains

You don't need to authenticate with AWS to start working with the AWS Toolkit for JetBrains. However, most AWS resources are managed through an AWS account. If you've already set up an AWS account and authentication method, see the [Connecting to AWS](#) topic in this User Guide to get started connecting to your AWS account.

The following topics contain additional details and set up instructions for each AWS credential type and authentication method that's compatible with the AWS Toolkit for JetBrains.

Topics

- [AWS IAM Identity Center](#)
- [AWS IAM credentials](#)
- [Authenticating with an AWS Builder ID](#)

AWS IAM Identity Center

AWS IAM Identity Center is the recommended best practice for managing your AWS account authentication.

For detailed instructions on how to set up IAM Identity Center for Software Development Kits (SDKs) and the AWS Toolkit for JetBrains, see the [IAM Identity Center authentication](#) section in the *AWS SDKs and Tools Reference Guide*.

For instructions on how to authenticate and connect your IAM Identity Center account with the AWS Toolkit for JetBrains, see the [Connecting to AWS](#) topic in this User Guide.

AWS IAM credentials

AWS Identity and Access Management (AWS IAM) Credentials authenticate with your AWS account through locally-stored access keys.

For instructions on how to authenticate with your existing IAM Credentials in the AWS Toolkit for JetBrains, see the [Connecting to AWS](#) topic in this User Guide.

The following sections describe how to set up new IAM Credentials.

Important

Before setting up IAM credentials to authenticate with your AWS account, note that:

- If you've already set IAM credentials through another AWS service (such as the AWS CLI), then the AWS Toolkit for JetBrains automatically detects those credentials and makes them available.
- AWS recommends using IAM Identity Center authentication. For additional information about AWS IAM best practices, see the [Security best practice in IAM](#) section of the *AWS Identity and Access Management User Guide*.
- To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

Prerequisites

Before you can configure the AWS Toolkit for JetBrains to authenticate with IAM user credentials, the following prerequisites must be met. If you've already set up IAM user credentials through another service (such as the AWS Command Line Interface), then you can skip the prerequisite steps and proceed to the following sections.

1. **Create an IAM user.** For detailed instructions on how to create an IAM user, see [Step 1: Create your IAM user](#) in the *AWS SDKs and Tools Reference Guide*.
2. **Get your IAM user access keys.** For detailed instructions on how to get your IAM user access keys, see [Step 2: Get your access keys](#) in the *AWS SDKs and Tools Reference Guide*.
3. **Optional: Update the shared credentials file.** For detailed instructions on how to update the shared credentials file, see [Step 3: Update the shared credentials file](#) in the *AWS SDKs and Tools Reference Guide*.

Note

If the optional prerequisite **Step 3: Update the shared credentials file** has been completed, the AWS Toolkit for JetBrains automatically detects your credentials during the **Creating a shared credentials file from the AWS Toolkit for JetBrains** procedure described in the following section.

Creating a shared credentials file from the AWS Toolkit for JetBrains

Your *shared config file* and *shared credentials file* store configuration and credential information for your AWS accounts. For more information about shared configuration and credentials, see the [Where are configuration settings stored?](#) section in the *AWS Command Line Interface User Guide*.

Creating a shared credentials file from the AWS Toolkit for JetBrains

1. From the AWS Toolkit for JetBrains open **AWS Connection Settings** by choosing the ... (ellipsis) icon.
2. From the **AWS Connection Settings** menu, choose **Set up authentication** to open the **AWS Toolkit for JetBrains** connection UI.
3. From the **AWS Explorer** section of the **Authenticate with AWS Toolkit** connection UI, choose the **Authenticate with IAM** link to open the **AWS Toolkit: Setup Authentication** dialog.
4. From the **IAM Credentials** tab, enter your **Profile Name**, **Access Key ID**, and **Secret Access Key**, then choose the **Connect** button to add the profile to your config file and connect the Toolkit with your AWS account.
5. The Toolkit **AWS Explorer** updates to display your AWS services and resources when authentication is complete and a connection has been established.

Configuring shared credentials

The following procedure describes how to configure your shared credentials files from the AWS Toolkit for JetBrains.

1. From the AWS Toolkit for JetBrains, choose **+ Add Connection to AWS** to open the **AWS Toolkit: Add Connection** dialog box.
2. From the **AWS Toolkit: Add Connection** dialog box, choose **Edit AWS Credential files(s)** to open your **Credential File**.
3. When your `credentials` file opens in the JetBrains, locate the section labeled `[default]`.
4. From the `[default]` section, locate the entry `#aws_access_key_id =`, remove the `#` and enter your AWS access key. The entry should look similar to the following:

```
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
```

5. From the [default] section, locate the entry `#aws_secret_access_key =`, remove the `#` and enter your AWS secret access key. The entry should look similar to the following:

```
aws_secret_access_key = wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

The final version of your updated credentials file resembles the following:

```
[default]
# The access key and secret key pair identify your account and grant access to AWS.
# Treat your secret key like a password. Never share your secret key with anyone.
Do
# not post it in online forums, or store it in a source control system. If your
secret
# key is ever disclosed, immediately use IAM to delete the access key and secret
key
# and create a new key pair. Then, update this file with the replacement key
details.
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

6. Save your changes to the file, the AWS Toolkit for JetBrains automatically detects your updated credentials and connects to your AWS account.

```
aws_secret_access_key = wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

Authenticating with an AWS Builder ID

This topic describes how to authenticate with your AWS Builder ID from the AWS Toolkit for JetBrains. For detailed information about the AWS Builder ID authentication method, see the [Sign in with AWS Builder ID](#) topic in the *AWS Sign-In* User Guide.

For instructions on how to authenticate and connect your AWS Builder ID account with the AWS Toolkit for JetBrains, see the [Connecting to AWS](#) topic in this User Guide.

Working with AWS services from the AWS Toolkit Explorer

Your AWS services and resources are available through the AWS Toolkit for JetBrains Explorer.

For more information on how to navigate the AWS Toolkit for JetBrains and the AWS Explorer, see the [Navigation](#) topic in this User Guide.

To learn more about working with a specific AWS service from the AWS Toolkit for JetBrains, choose from the following list of topics.

Topics

- [Working with experimental features](#)
- [Using AWS Toolkit for JetBrains with AWS App Runner](#)
- [Amazon CodeCatalyst for JetBrains](#)
- [Working with AWS CloudFormation by using the AWS Toolkit for JetBrains](#)
- [Working with CloudWatch Logs by using the AWS Toolkit for JetBrains](#)
- [Amazon DynamoDB in the AWS Toolkit for JetBrains](#)
- [Working with Amazon Elastic Container Service by Using the AWS Toolkit for JetBrains](#)
- [Working with Amazon EventBridge by using the AWS Toolkit for JetBrains](#)
- [Working with AWS Lambda from the AWS Toolkit for JetBrains](#)
- [Accessing Amazon RDS by using the AWS Toolkit for JetBrains](#)
- [Accessing Amazon Redshift by using the AWS Toolkit for JetBrains](#)
- [Working with Amazon S3 by using the AWS Toolkit for JetBrains](#)
- [Working with AWS serverless applications by using the AWS Toolkit for JetBrains](#)
- [Working with Amazon Simple Queue Service from the AWS Toolkit for JetBrains](#)
- [Working with resources](#)

Working with experimental features

Experimental features offer early access to features in the AWS Toolkit for JetBrains before they're officially released.

⚠ Warning

Because experimental features continue to be tested and updated, they may have usability issues. And experimental features may be removed from the AWS Toolkit for JetBrains without notice.

You can enable experimental features for specific AWS services in the **AWS** section of the **Settings** pane in your JetBrains IDE.

1. To edit AWS settings in JetBrains, choose **File, Settings** (or press **Ctrl+Alt+S**).
2. In the **Settings** pane, expand **Tools** and choose **AWS, Experimental Features**.
3. Select the checkboxes for the experimental features you want to access prior to release. If you want to switch off an experimental feature, clear the relevant checkbox.
4. After enabling experimental features, you can confirm by opening the **AWS Explorer** and choosing **Options** (the gear icon), **Experimental Features**. A checkmark beside the name of the feature indicates that it's available for use.

Using AWS Toolkit for JetBrains with AWS App Runner

[AWS App Runner](#) provides a fast, simple, and cost-effective way to deploy from source code or a container image directly to a scalable and secure web application in the AWS Cloud. Using it, you don't need to learn new technologies, decide which compute service to use, or know how to provision and configure AWS resources.

You can use AWS App Runner to create and manage services based on a *source image* or *source code*. If you use a source image, you can choose a public or private container image that's stored in an image repository. App Runner supports the following image repository providers:

- Amazon Elastic Container Registry (Amazon ECR): Stores private images in your AWS account.
- Amazon Elastic Container Registry Public (Amazon ECR Public): Stores publicly readable images.

If you choose the source code option, you can deploy from a source code repository that's maintained by a supported repository provider. Currently, App Runner supports [GitHub](#) as a source code repository provider:

Prerequisites

This section assumes you already have an AWS account and the latest version of AWS Toolkit for JetBrains that features AWS App Runner. In addition to those core requirements, make sure that all relevant IAM users have permissions to interact with the App Runner service. Also you need to obtain specific information about your service source such as the container image URI or the connection to the GitHub repository. You need this information when creating your App Runner service.

Configuring IAM permissions for App Runner

The easiest way to grant the permissions that are required for App Runner is to attach an existing AWS managed policy to the relevant IAM entity, specifically a user or group. App Runner provides two managed policies that you can attach to your IAM users:

- `AWSAppRunnerFullAccess`: Allows users to perform all App Runner actions.
- `AWSAppRunnerReadOnlyAccess`: Allow users to list and view details about App Runner resources.

In addition, if you choose a private repository from the Amazon Elastic Container Registry (Amazon ECR) as the service source, you must create the following access role for your App Runner service:

- `AWSAppRunnerServicePolicyForECRAccess`: Allows App Runner to access Amazon Elastic Container Registry (Amazon ECR) images in your account.

You can use the **Create App Runner Service** dialog box to create this IAM role.

Note

The `AWSServiceRoleForAppRunner` service-linked role allows AWS App Runner to complete the following tasks:

- Push logs to Amazon CloudWatch Logs log groups.
- Create Amazon CloudWatch Events rules to subscribe to Amazon Elastic Container Registry (Amazon ECR) image push.

You don't need to manually create the service-linked role. When you create an AWS App Runner in the AWS Management Console or by using API operations that are called by AWS Toolkit for JetBrains, AWS App Runner creates this service-linked role for you.

For more information, see [Identity and access management for App Runner](#) in the *AWS App Runner Developer Guide*.

Obtaining service sources for App Runner

You can use AWS App Runner to deploy services from a source image or source code.

Source image

If you're deploying from a source image, you can obtain a link to the repository for that image from a private or public AWS image registry.

- Amazon ECR private registry: Copy the URI for a private repository that uses the Amazon ECR console at <https://console.aws.amazon.com/ecr/repositories>.
- Amazon ECR public registry: Copy the URI for a public repository that uses the Amazon ECR Public Gallery at <https://gallery.ecr.aws/>.

You specify the URI for the image repository when entering details for your source in the **Create App Runner Service** dialog box.

For more information, see [App Runner service based on a source image](#) in the *AWS App Runner Developer Guide*.

Source code

For your source code to be deployed to an AWS App Runner service, that code must be stored in a Git repository that's maintained by a supported repository provider. App Runner supports one source code repository provider: [GitHub](#).

For information about setting up a GitHub repository, see the [Getting started documentation](#) on GitHub.

To deploy your source code to an App Runner service from a GitHub repository, App Runner establishes a connection to GitHub. If your repository is private (that is, it isn't publicly accessible on GitHub), you must provide App Runner with connection details.

Important

To create GitHub connections, you must use the App Runner console (<https://console.aws.amazon.com/apprunner>) to create a connection that links GitHub to AWS. You can select the connections that are available on the **GitHub connections** page when using the **Create App Runner Service** dialog box to specify details about your source code repository.

For more information, see [Managing App Runner connections](#) in the *AWS App Runner Developer Guide*.

The App Runner service instance provides a managed runtime that allows your code to build and run. AWS App Runner currently supports the following runtimes:

- Python managed runtime
- Node.js managed runtime

Using the **Create App Runner Service** dialog box that's available in AWS Toolkit for JetBrains, you provide information about how the App Runner service builds and starts your service. You can enter the information directly in the interface or specify a YAML-formatted [App Runner configuration file](#). Values in this file instruct App Runner how to build and start your service, and provide runtime context. This includes relevant network settings and environment variables. The configuration file is named `apprunner.yaml`. It's automatically added to root directory of your application's repository.

Pricing

You're charged for the compute and memory resources that your application uses. In addition, if you automate your deployments, you also pay a set monthly fee for each application that covers all automated deployments for that month. If you opt to deploy from source code, you additionally pay a build fee for the amount of time that it takes App Runner to build a container from your source code.

For more information, see [AWS App Runner Pricing](#).

Topics

- [Creating App Runner services](#)
- [Managing App Runner services](#)

Creating App Runner services

You can create an App Runner service in AWS Toolkit for JetBrains by using the **Create App Runner Service** dialog box. You can use its interface to select a source repository and configure the service instance where your application runs.

Before creating an App Runner service, make sure that you completed all the [prerequisites](#). Included is providing the relevant IAM permissions and taking note of the specific information about the source repository that you want to deploy.

To create an App Runner service

1. Open AWS Explorer, if it isn't already open.
2. Right-click the **App Runner** node and choose **Create Service**.

The **Create App Runner Service** dialog box is displayed.

3. Enter your unique **Service name**.
4. Choose your source type (**ECR**, **ECR public** or **Source code repository**) and configure the relevant settings:

ECR/ECR public

If you're using a private registry, choose the **Deployment type**:

- **Manual**: Use manual deployment if you want to explicitly initiate each deployment to your service.
- **Automatic**: Use automatic deployment if you want implement continuous integration and deployment (CI/CD) behavior for your service. If you choose this option, it means that whenever you push a new image version to your image repository or a new commit to your code repository, App Runner automatically deploys it to your service without further action required from you.

For **Container image URI**, enter the URI of the image repository that you copied from your Amazon ECR private registry or Amazon ECR Public Gallery.

For **Start Command**, enter the command to start the service process.

For **Port**, enter the IP port that's used by the service.

If you're using an Amazon ECR private registry, select the required **ECR access role** and choose **Create**.

- The **Create IAM Role** dialog box displays the **Name**, **Managed Policies**, and **Trust Relationships** for the IAM role. Choose **Create**.

Source code repository

Choose the **Deployment type**:

- **Manual**: Use manual deployment if you want to explicitly initiate each deployment to your service.
- **Automatic**: Use automatic deployment if you want implement continuous integration and deployment (CI/CD) behavior for your service. If you choose this option, it means that whenever you push a new image version to your image repository, or a new commit to your code repository, App Runner automatically deploys it to your service without further action required from you.

For **Connections**, select a connection that's available from the list on **GitHub connections** page.

For **Repository URL**, enter the link to remote repository that's hosted on GitHub.

For **Branch**, indicate which Git branch of your source code that you want to deploy.

For **Configuration**, specify how you want to specify your runtime configuration:

- **Configure all settings here**: Choose this option if you want to specify the following settings for the runtime environment of your application:
 - **Runtime**: Choose **Python 3** or **Nodejs 12**.

- **Port:** Enter the IP port that your service uses.
 - **Build command:** Enter the command to build your application in the runtime environment of your service instance.
 - **Start command:** Enter the command to start your application in the runtime environment of your service instance.
 - **Provide a configuration file settings here:** Choose this option to use the settings that are defined by the `apprunner.yaml` configuration file. This file is in the root directory of your application's repository.
5. Specify values to define the runtime configuration of the App Runner service instance:
- **CPU:** The number of CPU units that are reserved for each instance of your App Runner service (Default: 1 vCPU).
 - **Memory:** The amount of memory that's reserved for each instance of your App Runner service (Default: 2 GB)
 - **Environmental variables:** Optional environment variables that you use to customize behavior in your service instance. Create environmental variables by defining a key and a value.
6. Choose **Create**
- When your service is being created, its status changes from **Operation in progress** to **Running**.
7. After your service starts running, right-click it and choose **Copy Service URL**.
8. To access your deployed application, paste the copied URL into the address bar of your web browser.

Managing App Runner services

After creating an App Runner service, you can manage it by using the AWS Explorer pane to carry out the following activities:

- [Pausing and resuming App Runner services](#)
- [Deploying App Runner services](#)
- [Viewing logs streams for App Runner](#)
- [Deleting App Runner services](#)

Pausing and resuming App Runner services

If you need to disable your web application temporarily and stop the code from running, you can pause your AWS App Runner service. App Runner reduces the compute capacity for the service to zero. When you're ready to run your application again, resume your App Runner service. App Runner provisions new compute capacity, deploys your application to it, and runs the application.

Important

You're billed for App Runner only when it's running. Therefore, you can pause and resume your application as needed to manage costs. This is particularly helpful in development and testing scenarios.

To pause your App Runner service

1. Open AWS Explorer, if it isn't already open.
2. Expand **App Runner** to view the list of services.
3. Right-click your service and choose **Pause**.
4. In the dialog box that displays, choose **Pause**.

While the service is pausing, the service status changes from **Running** to **Operation in progress** and then to **Paused**.

To resume your App Runner service

1. Open AWS Explorer, if it isn't already open.
2. Expand **App Runner** to view the list of services.
3. Right-click your service and choose **Resume**.
4. In the dialog box that displays, choose **Resume**.

While the service is resuming, the service status changes from **Paused** to **Operation in progress** and then to **Running**.

Deploying App Runner services

If you choose the manual deployment option for your service, you need to explicitly initiate each deployment to your service.

1. Open AWS Explorer, if it isn't already open.
2. Expand **App Runner** to view the list of services.
3. Right-click your service and choose **Deploy**.
4. While your application is being deployed, the service status changes from **Operation in progress** to **Running**.
5. To confirm that your application is successfully deployed, right-click the same service and choose **Copy Service URL**.
6. To access your deployed web application, paste the copied URL into the address bar of your web browser.

Viewing logs streams for App Runner

Use CloudWatch Logs to monitor, store, and access your log files for services such as App Runner. CloudWatch Logs records two distinct types of log files: log events and log streams. Log events are records of activity that was recorded by the application or resource that you monitor with CloudWatch Logs. A log stream is a sequence of log events that share the same source.

You can access the two following types of log streams for App Runner:

- **Service log streams:** Contains the logging output that's generated by App Runner. For this type of log stream, the log events are records of how App Runner manages your service and acts on it.
- **Application log streams:** Contains the output of your running application code.

1. Expand **App Runner** to view the list of services
2. Right-click a service and choose one of the following options:
 - **View Service Log Streams**
 - **View Application Log Streams**

The **Log Streams** pane displays the log events that make up the log stream.

3. To view more information about a specific event, right-click it and choose **Export Log Stream**, **Open in Editor** or **Export Log Stream, Save to a File**.

Deleting App Runner services

Important

If you delete your App Runner service, it's permanently removed and your stored data is deleted. If you need to recreate the service, App Runner needs to fetch your source again and build it if it's a code repository. Your web application gets a new App Runner domain.

1. Open AWS Explorer, if it isn't already open.
2. Expand **App Runner** to view the list of services.
3. Right-click a service and choose **Delete Service**.
4. In the confirmation dialog box, enter *delete me* and then choose **OK**.

The deleted service displays the **Operation in progress** status, and then the service disappears from the list.

Amazon CodeCatalyst for JetBrains

What is Amazon CodeCatalyst?

Amazon CodeCatalyst is a cloud-based collaboration space for software development teams. Using the AWS Toolkit for JetBrains Gateway, you can view and manage your CodeCatalyst resources directly from JetBrains Gateway. You can also use the Toolkit to launch, manage, and edit Dev Environments virtual computing environments. For more information about CodeCatalyst, see the [Amazon CodeCatalyst User Guide](#).

The following topics describe how to connect the AWS Toolkit for JetBrains Gateway with CodeCatalyst and how to work with CodeCatalyst through JetBrains Gateway.

Topics

- [Getting Started with CodeCatalyst and the AWS Toolkit for JetBrains](#)
- [Working with Amazon CodeCatalyst in JetBrains Gateway](#)

Getting Started with CodeCatalyst and the AWS Toolkit for JetBrains

To get started working with CodeCatalyst from the JetBrains Gateway, complete the following.

Downloading JetBrains Gateway

Before you integrate the AWS Toolkit with your CodeCatalyst accounts, make sure that you're using the current version of JetBrains Gateway. To download the latest version of JetBrains Gateway, choose the JetBrains Gateway distribution you want from the following links:

- [JetBrains Gateway for Linux](#)
- [JetBrains Gateway for Windows](#)
- [JetBrains Gateway for macOS](#)
- [JetBrains Gateway for macOS Apple Silicon](#)

Installing the AWS Toolkit for JetBrains Gateway

To connect JetBrains with your CodeCatalyst account, you must install the latest version of the toolkit extension. You can find the latest version and complete the installation of the toolkit directly from the JetBrains **Plugins Marketplace**.

To install the AWS Toolkit plugin from the JetBrains **Plugins Marketplace**, complete the following steps:


1. From the JetBrains Gateway main screen, choose the **Settings/Preferences** icon, located in the bottom-left corner of the application.
2. choose **Settings/Preferences** to open the **Settings/Preferences** view.
3. In the **Settings/Preferences** view, choose **Plugins** to open the **Plugins** view.

Note

The **Plugins** view can open in either the **Marketplace** view or the **Installed** view.

- If this is your first time installing the AWS Toolkit for JetBrains Gateway, select the **Plugins Marketplace** view to continue.
- If you have a previous version of the AWS Toolkit for JetBrains Gateway, update it from the **Installed** view.

4. From the **Marketplace** view, enter the text **AWS Toolkit** and choose the **AWS Toolkit** plugin entry when it appears.
5. Choose **Install** to download and install the **AWS Toolkit for JetBrains Gateway**.

 **Note**


JetBrains Gateway displays the status of your download and installation progress. After the toolkit is successfully installed, the JetBrains Gateway **Connections** explorer updates with the **Amazon CodeCatalyst** plugin icon.

Creating a CodeCatalyst account and setting up authentication

To work with CodeCatalyst from the AWS Toolkit for JetBrains, you must have an active CodeCatalyst account and set up either an AWS Builder ID or IAM Identity Center authentication to connect with JetBrains Gateway. If you don't have an active CodeCatalyst account, AWS Builder ID, or IAM Identity Center authentication set up, see the [Setting up with CodeCatalyst](#) section in the *CodeCatalyst* User Guide.

Authenticate and connect JetBrains Gateway with CodeCatalyst

To authenticate with AWS and connect JetBrains Gateway with your CodeCatalyst account, complete the following steps.

 **Note**

To authenticate with AWS using IAM Identity Center, you must be running AWS Toolkit for JetBrains version 2.6 or later.

1. From the JetBrains Gateway **Connections** explorer, choose the **Amazon CodeCatalyst** plugin to open the **Amazon CodeCatalyst** plugin view.
2. From the **CodeCatalyst** plugin view, choose **Sign in** to open the **AWS Toolkit: Setup Authentication** dialog.
3. From the **AWS Toolkit: Setup Authentication** dialog, choose the tab for your preferred authentication method: **IAM Identity Center** or **AWS Builder ID**.

4. From the tab of your preferred authentication method, complete any required fields, then choose the **Connect** button to open the **AWS authentication portal** in your default web browser.
5. From your web browser, confirm the security code matches the code presented in your IDE to proceed.
6. When prompted, choose **Allow** to confirm the connection between JetBrains and your AWS account. When the connection process is complete, your web browser displays a confirmation indicating that it's safe to close the window and return to your IDE.
7. The JetBrains Gateway **CodeCatalyst** plugin view is updated to the **Dev Environments** view.

Working with Amazon CodeCatalyst in JetBrains Gateway

You can launch a virtual computing environment, known as a Dev Environment, from JetBrains. Dev Environments are customizable cloud-development environments that you can copy and share among different team members in your Space. For more information about Dev Environments and how you can access them from CodeCatalyst, see the [Dev Environments](#) section in the *Amazon CodeCatalyst User Guide*.

The following sections describe how to create, open, and work with Dev Environments from JetBrains Gateway.

Opening a Dev Environment

To open an existing Dev Environment from JetBrains Gateway, complete the following steps.

1. From the **Connections** explorer, choose the **Amazon CodeCatalyst** plugin.
2. From the **Remote Development** wizard body, navigate to the parent Space and project for the Dev Environment that you want to open.
3. Choose the Dev Environment that you want to open.
4. Confirm the opening process for your Dev Environment to continue.

Note

JetBrains displays the progress in a new status window, when the opening process is complete, your Dev Environment opens in a new window.

Creating a Dev Environment

To create a new Dev Environment:

1. From the **Connections** explorer, choose the **CodeCatalyst** plugin.
2. From the **Remote Development** wizard header section, choose the **Create a Dev Environment** link to open the **New CodeCatalyst Dev Environment** view.
3. From the **New CodeCatalyst Dev Environment** view, use the following fields to configure your Dev Environment preferences.
 - **IDE:** Select your preferred JetBrains IDE to launch in your Dev Environment.
 - **CodeCatalyst Project:** Choose a CodeCatalyst Space and project for your Dev Environment.
 - **Dev Environment Alias:** Enter an alternate name for your Dev Environment.
 - **Compute:** Choose the virtual hardware configuration for your Dev Environment.
 - **Persistent storage:** Choose the amount of persistent storage for your Dev Environment.
 - **Inactivity timeout:** Choose the system idle time that passes before your Dev Environment enters into standby.
4. To create your Dev Environment, choose **Create Dev Environment**.

Note

When you choose **Create Dev Environment**, the **New Dev Environment** view closes and the process to create your Dev Environment starts. The process can take several minutes and you can't use other JetBrains Gateway features until your Dev Environment is created.

JetBrains displays the progress in a new status window, and, when the process is complete, your Dev Environment opens in a new window.

Creating a Dev Environment from a third-party repository

You can create Dev Environments from a third-party repository by linking to the repository as a source.

Linking to a third-party repository as a source is handled at the project level in CodeCatalyst. For instructions and additional details on how to connect a third-party repository to your Dev Environment, see the [Linking a source repository](#) topic in the *Amazon CodeCatalyst User Guide*.

Configuring Dev Environment settings

To change the settings for an existing Dev Environment from JetBrains Gateway, complete the following steps.

Note

You can't modify the amount of storage assigned to your Dev Environment after it has been created.

1. From the **Connections** explorer, choose the **Amazon CodeCatalyst** plugin.
2. From the **Remote Development** wizard body, navigate to the parent Space and project for the Dev Environment that you want to configure.
3. Choose the **Settings** icon, next to the Dev Environment that you want to configure, to open the **Configure Dev Environment:** settings.
4. From **Configure Dev Environment:** settings menu, configure your Dev Environment by changing the following options:
 - **Dev Environment Alias:** Optional field to specify and alternate name for your Dev Environment.
 - **IDE:** choose the JetBrains IDE you want to launch inside of your Dev Environment.
 - **Compute:** Choose the virtual hardware configuration for your Dev Environment.
 - **Inactivity timeout:** Choose the system idle time that passes before your Dev Environment enters into standby.

Pausing a Dev Environment

The activity in your Dev Environment is stored persistently. This means that you can pause and resume your Dev Environment without losing your work.

To pause your Dev Environment, complete the following steps.

1. From the **Connections** explorer, choose the **Amazon CodeCatalyst** plugin.
2. From the **Remote Development** wizard body, navigate to the parent Space and project for the Dev Environment that you want to pause.

3. Choose the **Pause** icon next to your active Dev Environment to open the **Confirm Pause** dialog.
4. Choose **Yes** to close the **Confirm Pause** dialog and initialize the pause process.

Note

JetBrains displays the progress of the pause process in a new status window. When the Dev Environment has stopped, the **Pause** icon is removed from the user interface.

Resuming a Dev Environment

The activity in your Dev Environment is stored persistently. This means that you can resume a paused Dev Environment without losing your previous work.

To resume a paused Dev Environment, complete the following steps.

1. From the **Connections** explorer, choose the **Amazon CodeCatalyst** plugin.
2. From the **Remote Development** wizard body, navigate to the parent Space and project for the Dev Environment that you want to resume.
3. Choose the Dev Environment you want to resume.

Note

JetBrains displays the progress of the resume process in a new status window. When the Dev Environment has resumed, a **Pause** icon is added next to the Dev Environment **Settings** icon.

Deleting a Dev Environment

To delete your Dev Environment, complete the following steps:

1. From the **Connections** explorer, choose the **Amazon CodeCatalyst** plugin.
2. From the **Remote Development** wizard body, navigate to the parent Space and project for the Dev Environment that you want to delete.
3. Choose the **X** icon button next to your Dev Environment to open the **Confirm Deletion** dialog.
4. Choose **Yes** to close the dialog and delete your Dev Environment.

⚠ Important

After you choose **Yes**, your Dev Environment is deleted and can't be retrieved. Before deleting a Dev Environment, make sure to commit and push your code changes to the original source repository. Otherwise, your unsaved changes will be permanently lost. After a Dev Environment is deleted, the **Remote Development** wizard updates and the Dev Environment is no longer listed in your resources.

Configuring Dev Environments defaults

You can configure your Dev Environment default settings in the `devfile` of your Dev Environment. The `devfile` specification is an open standard, which you can update in a YAML document.

For more information about how to define and configure your `devfile`, see devfile.io.

To open and edit your `devfile` from your JetBrains Gateway Dev Environment instance, complete the following steps.

1. From the **Navigation bar** in your active JetBrains Dev Environment, expand the **Amazon CodeCatalyst Dev Environment** node to open the **Backend Status Details** menu.
2. Choose the **Configure Dev Environment** tab, then choose **Open Devfile** to open your `devfile` in the JetBrains **Editor**.
3. From the **Editor**, make changes to your `devfile` and save your work.
4. Upon saving your changes, the **Amazon CodeCatalyst Dev Environment** node displays an alert indicating that your Dev Environment requires a rebuild.
5. Expand the **Amazon CodeCatalyst Dev Environment** node and choose the **Rebuild Dev Environment** node from the **Configure Dev Environment** tab.

Working with AWS CloudFormation by using the AWS Toolkit for JetBrains

The following topics describe how to use the AWS Toolkit for JetBrains to work with AWS CloudFormation stacks in an AWS account.

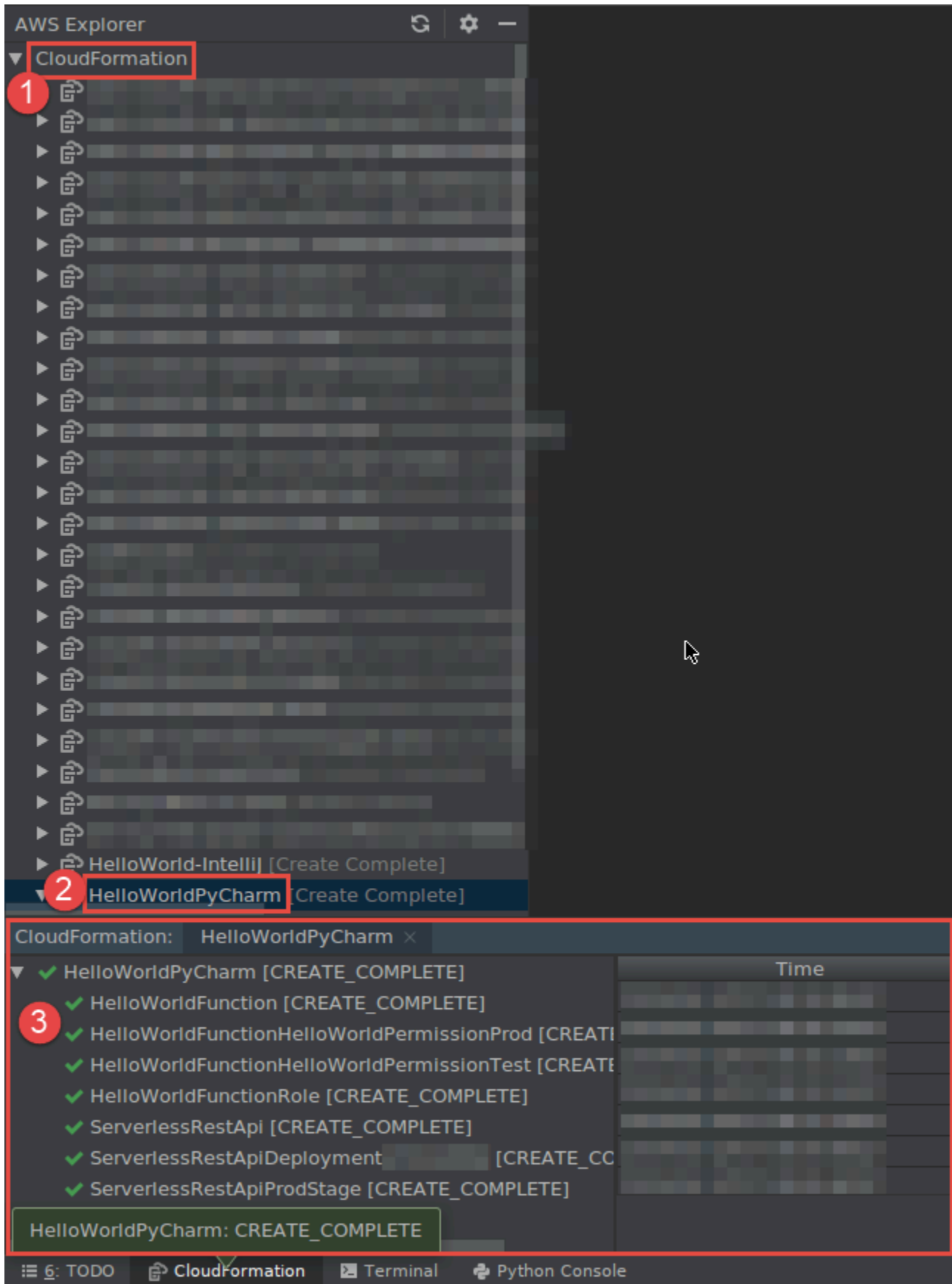
Topics

- [Viewing event logs for an AWS CloudFormation stack by using the AWS Toolkit for JetBrains](#)
- [Deleting an AWS CloudFormation stack by using the AWS Toolkit for JetBrains](#)

Viewing event logs for an AWS CloudFormation stack by using the AWS Toolkit for JetBrains

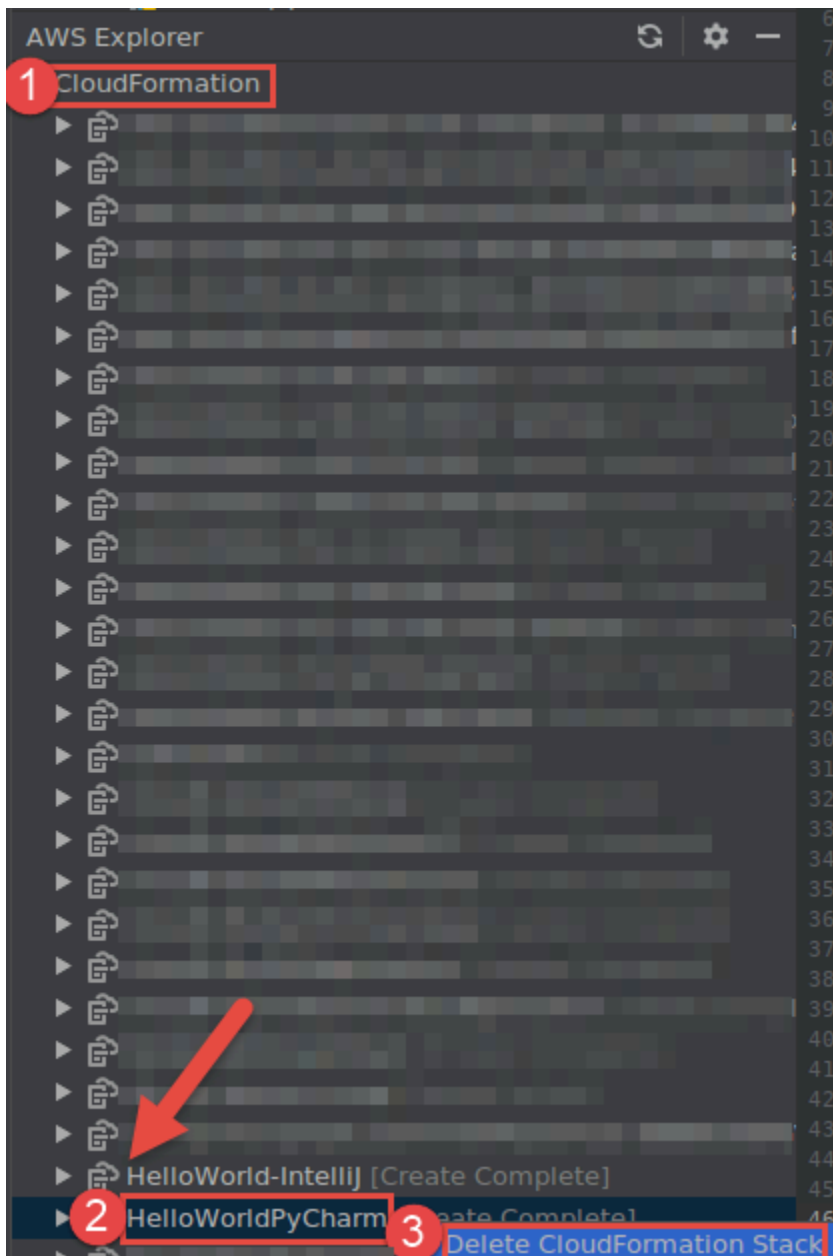
1. Open AWS Explorer, if it isn't already open. If the stack is in an AWS Region that's different from the current one, switch to a different AWS Region that contains it.
2. Expand **CloudFormation**.
3. To view event logs for the stack, right-click the stack's name. The AWS Toolkit for JetBrains displays the event logs in the **CloudFormation** tool window.

To hide or show the **CloudFormation** tool window, on the main menu, choose **View, Tool Windows, CloudFormation**.



Deleting an AWS CloudFormation stack by using the AWS Toolkit for JetBrains

1. Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region that contains the stack, do that now.
2. Expand **CloudFormation**.
3. Right-click the name of the stack to delete, and then choose **Delete CloudFormation Stack**.



4. Enter the stack's name to confirm it's deleted, and then choose **OK**. If the stack deletion succeeds, the AWS Toolkit for JetBrains removes the stack name from the **CloudFormation** list

in **AWS Explorer**. If the stack deletion fails, you can troubleshoot by viewing the event logs for the stack.

Working with CloudWatch Logs by using the AWS Toolkit for JetBrains

Amazon CloudWatch Logs enables you to centralize the logs from all of your systems, applications, and AWS services that you use, in a single, highly scalable service. You can then easily view them, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis. For more information, see [What Is Amazon CloudWatch Logs?](#) in the *Amazon CloudWatch User Guide*.

The following topics describe how to use the AWS Toolkit for JetBrains to work with CloudWatch Logs in an AWS account.

Topics

- [Viewing CloudWatch log groups and log streams by using the AWS Toolkit for JetBrains](#)
- [Working with CloudWatch log events in log streams by using the AWS Toolkit for JetBrains](#)
- [Working with CloudWatch Logs Insights by using the AWS Toolkit for JetBrains](#)

Viewing CloudWatch log groups and log streams by using the AWS Toolkit for JetBrains

A *log stream* is a sequence of log events that share the same source. Each separate source of logs into CloudWatch Logs makes up a separate log stream.

A *log group* is a group of log streams that share the same retention, monitoring, and access control settings. You can define log groups and specify which streams to put into each group. There is no limit on the number of log streams that can belong to one log group.

For more information, see [Working with Log Groups and Log Streams](#) in the *Amazon CloudWatch User Guide*.

Topics

- [Viewing log groups and log streams with the CloudWatch Logs node](#)

- [Viewing log streams with the Lambda node](#)
- [Viewing log streams with the Amazon ECS node](#)

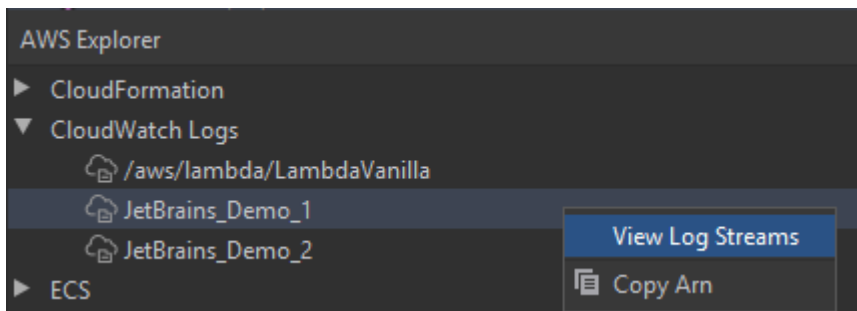
Viewing log groups and log streams with the CloudWatch Logs node

1. Open AWS Explorer, if it isn't already open.
2. Click the **CloudWatch Logs** node to expand the list of log groups.

The log groups for the [current AWS Region](#) are displayed under the **CloudWatch Logs** node.

3. To view the log streams in a log group, do one of the following:
 - Double-click the name of the log group.
 - Right-click the name of the log group, and then choose **View Log Streams**.

The log group's contents are displayed in the **Log Streams** pane. For information about interacting with the log events in each stream, see [Working with CloudWatch log events](#).



Viewing log streams with the Lambda node

You can view CloudWatch Logs for AWS Lambda functions by using the **Lambda** node in AWS Explorer.

Note

You can also view log streams for all AWS services, including Lambda functions, using the **CloudWatch Logs** node in AWS Explorer. We recommend using the **Lambda** node, however, for an overview of log data specific to Lambda functions.

1. Open AWS Explorer, if it isn't already open.

2. Click the **Lambda** node to expand the list of Lambda functions.

The Lambda functions for the [current AWS Region](#) are displayed beneath the **Lambda** node.

3. Right-click a Lambda function, and then choose **View Log Streams**.

The log streams for the function are displayed in the **Log Streams** pane. For information about interacting with the log events in each stream, see [Working with CloudWatch log events](#).

Viewing log streams with the Amazon ECS node

You can view CloudWatch Logs for clusters and containers that are run and maintained in Amazon Elastic Container Service by using the **Amazon ECS** node in AWS Explorer

Note

You can also view log groups for all AWS services, including Amazon ECS, using the **CloudWatch Logs** node in AWS Explorer. We recommend using the **Amazon ECS** node, however, for an overview of log data specific to Amazon ECS clusters and containers.

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon ECS** node to expand the list of Amazon ECS clusters.

The Amazon ECS clusters for the [current AWS Region](#) are displayed beneath the **Amazon ECS** node.

3. Right-click a cluster, and then choose **View Log Streams**.

The log streams for the cluster are displayed in the **Log Streams** pane.

4. To view log streams for a specific container, click a cluster to expand its list of registered containers.

The containers registered for the cluster are displayed beneath.

5. Right-click a container, and then choose **View Container Log Stream**.

The log streams for the container are displayed in the **Log Streams** pane. For information about interacting with the log events for clusters and containers, see [Working with CloudWatch log events](#).

Working with CloudWatch log events in log streams by using the AWS Toolkit for JetBrains

After you've opened the **Log Streams** pane, you can access the log events in each stream. Log events are records of activity recorded by the application or resource being monitored.

Topics

- [Viewing and filtering log events in a stream](#)
- [Working with log actions](#)
- [Exporting CloudWatch log events to a file or an editor](#)

Viewing and filtering log events in a stream

When you open a log stream, the **Log Events** pane displays that stream's sequence of log events.

1. To find a log stream to view, open the **Log Streams** pane (see [Viewing CloudWatch log groups and log streams](#)).

Note

You can use pattern matching to locate a stream in a list. Click the **Log Streams** pane and start entering text. The first log stream name with text that matches yours is highlighted. You can also reorder the list by clicking the top of the **Last Event Time** column.

2. Double-click a log stream to view its sequence of log events.

The **Log Events** pane displays the log events that make up the log stream.

3. To filter the log events according to content, enter text in the **Filter logstream** field and press **Return**.

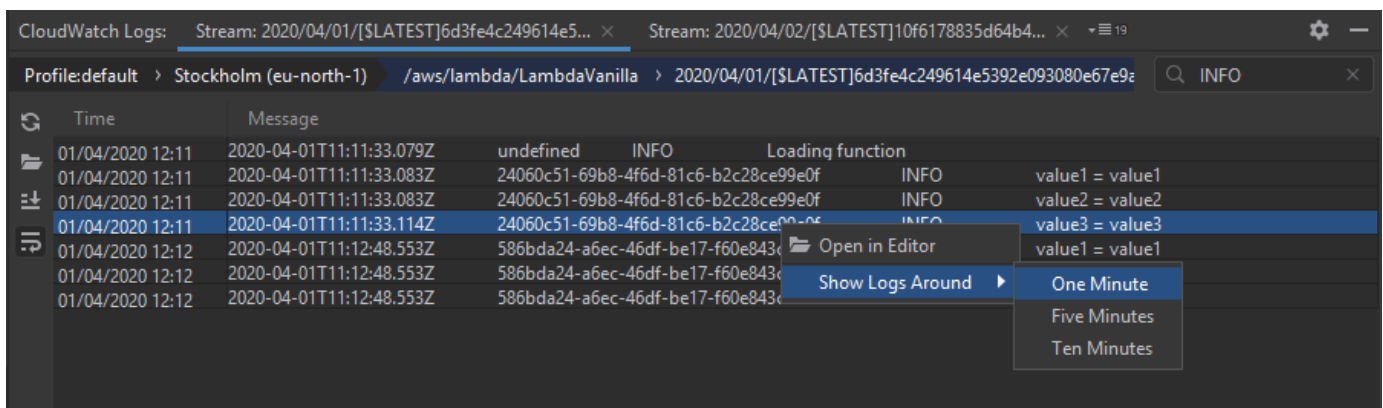
The results are log events containing text that's a case-sensitive match with your filter text. The filter searches the complete log stream, including events not displayed on the screen.

Note

You can also use pattern matching to locate a log event in the pane. Click the **Log Events** pane and start entering text. The first log event with text that matches yours is highlighted. Unlike with **Filter logstream** search, only on-screen events are checked.

4. To filter log events according to time, right-click a log event, and then choose **Show Logs Around**.

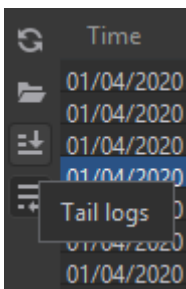
You can select **One Minute**, **Five Minutes**, or **Ten Minutes**. For example, if you select **Five Minutes**, the filtered list shows only log events that occurred five minutes before and after the selected entry.



On the left of the **Log Events** pane, the [log actions](#) offer more ways to interact with log events.

Working with log actions

On the left of the **Log Events** pane, four log actions allow you to refresh, edit, tail, and wrap CloudWatch log events.



1. To find log events to interact with, [open the Log Streams](#) pane.

2. Choose one of the following log actions:

- **Refresh** – Updates the list with log events that occurred after the **Log Events** pane was opened.
- **Open in Editor** – Opens the on-screen log events in the IDE's default editor.

Note

This action exports only on-screen log events to the IDE editor. To view all the stream's events in the editor, choose the [Export Log Stream](#) option.

- **Tail logs** – Streams new logs events to the **Log Events** pane. This is a useful feature for continuous updates on longer-running services such as Amazon EC2 instances and AWS CodeBuild builds.
- **Wrap logs** – Displays log event text on multiple lines if the size of the pane hides longer entries.

Exporting CloudWatch log events to a file or an editor

Exporting a CloudWatch log stream enables you to open its log events in the IDE's default editor or download them to a local folder.

1. To find a log stream to access, [open the Log Streams](#) pane.
2. Right-click a log stream, and then choose **Export Log Stream**, **Open in Editor** or **Export Log Stream, Save to a File**.
 - **Open in Editor** –Opens the log events that make up the selected stream in the IDE's default editor.

Note

This option exports all events in the log stream to the IDE editor.

- **Save to a File** – Opens the **Download Log Stream** dialog box. This enables you to select a download folder and rename the file containing the log events.

Working with CloudWatch Logs Insights by using the AWS Toolkit for JetBrains

You can use the AWS Toolkit for JetBrains to work with CloudWatch Logs Insights. CloudWatch Logs Insights enables you to interactively search and analyze your log data in Amazon CloudWatch Logs. For more information, see [Analyzing Log Data with CloudWatch Logs Insights](#) in the *Amazon CloudWatch Logs User Guide*.

IAM permissions for CloudWatch Logs Insights

You need the following permissions to run and view CloudWatch Logs Insights query results:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:StartQuery",
        "logs:GetQueryResults",
        "logs:GetLogRecord",
        "logs:describeLogGroups",
        "logs:describeLogStreams"
      ],
      "Resource" : "*"
    }
  ]
}
```

The following permission is not required but will allow the AWS Toolkit for JetBrains to automatically stop any currently running queries when you close the associated results pane or IDE.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:StopQuery"
      ]
    }
  ]
}
```

```
    ],  
    "Resource" : "*"    
  }  
]  
}
```

Working with CloudWatch Logs Insights

To open the CloudWatch Logs Insights query editor

1. Open AWS Explorer.
2. Double-click the **CloudWatch Logs** node to expand the list of log groups.
3. Right-click on the log group you want to open, and then choose **Open Query Editor**.

To start a CloudWatch Logs Insights query

1. In the **Query Log Groups** window, change the query parameters as desired.

You can choose a time range by date or relative time.

The **Query Log Groups** field accepts the CloudWatch Logs Insights Query Syntax. For more information, see [CloudWatch Logs Insights Query Syntax](#) in the *Amazon CloudWatch Logs User Guide*.

2. Choose **Execute** to begin the query.

To save a CloudWatch Logs Insights query

1. Type a query name.
2. Choose **Save Query**.

The selected log groups and query are saved to your AWS account. Time ranges are not saved.

You can retrieve and reuse saved queries from the CloudWatch Logs Insights AWS Management Console page.

To retrieve a saved CloudWatch Logs Insights query

1. In the **Query Log Groups** window, choose **Retrieve Saved Queries**.

2. Choose the desired query and choose **OK**.

The selected log groups and query replace anything in the existing dialog.

To navigate through query results

- In the CloudWatch Logs Insights **Query Results** window, in the top right corner, choose **Open Query Editor**.

To view an individual log record

- In the query results pane, double-click a row to open a new tab with details about that log record.

You can also navigate to the log record's associated log stream by choosing **View Log Stream** in the top right corner.

Amazon DynamoDB in the AWS Toolkit for JetBrains

Amazon DynamoDB is a fully managed NoSQL database service that provides predictable performance with seamless scalability. For detailed information about the DynamoDB service, see the [Amazon DynamoDB User Guide](#).

The following topics describe how to work with the DynamoDB service from the AWS Toolkit for JetBrains.

Topics

- [Working with Amazon DynamoDB from the AWS Toolkit for JetBrains](#)
- [Working with Amazon DynamoDB tables the AWS Toolkit for JetBrains](#)

Working with Amazon DynamoDB from the AWS Toolkit for JetBrains

The AWS Toolkit for JetBrains allows you to view, copy Amazon Resource Names (ARNs), and delete your Amazon DynamoDB resources, directly from your JetBrains IDE.

The following sections describe how to work with these service features from the AWS Toolkit for JetBrains.

Viewing DynamoDB resources

At this time, DynamoDB resources can't be created directly from the toolkit, but your resources are visible. To view your DynamoDB resources, complete the following steps:

1. Navigate to the **Explorer** tab in the AWS Toolkit for JetBrains.
2. Expand the **DynamoDB** node.
3. Your DynamoDB resources are displayed under the DynamoDB node.

Copying DynamoDB resources ARNs

An Amazon Resource Name (ARN) is a unique ID that gets assigned to every AWS resource, which include DynamoDB tables. To copy the ARN ID for a DynamoDB resource, complete the following steps:

1. Navigate to the **Explorer** tab in the AWS Toolkit for JetBrains.
2. Expand the **DynamoDB** node.
3. Open the context menu for (right-click) the DynamoDB resource you want to copy the ARN ID for.
4. Choose **Copy ARN** to copy the resource ARN ID to your OS clipboard.

Deleting DynamoDB resources

To delete a DynamoDB resource, complete the following steps:

1. Navigate to the **Explorer** tab in the AWS Toolkit for JetBrains.
2. Expand the **DynamoDB** node.
3. Open the context menu for (right-click) the DynamoDB resource you want to delete.
4. Choose **Delete table...** to open the **Delete table...** confirmation dialog.
5. Complete the confirmation instructions to delete your DynamoDB table.

Working with Amazon DynamoDB tables the AWS Toolkit for JetBrains

The primary resource of Amazon DynamoDB is a data-base table. The following sections describe how to work with DynamoDB tables from the AWS Toolkit for JetBrains.

Viewing a DynamoDB table

To view a DynamoDB table, complete the following steps:

1. Navigate to the **Explorer** tab in the AWS Toolkit for JetBrains.
2. Expand the **DynamoDB** node.
3. From your list of DynamoDB resources, double-click a table to view it in the **Editor** window.

Note

The first time you view table data, an initial scan with a max-result limit of 50 items is retrieved.

Setting the max-result limit

To change the default limit of retrieved table entries, complete the following steps:

1. From the AWS Explorer, double-click a table to view it in the JetBrains **Editor** window.
2. From the table view, choose the **Settings icon**, located in the upper-right hand corner of your **Editor** window.
3. Hover over the **Max Results** option to view a list of available max-result values.

Scanning a DynamoDB table

To scan a DynamoDB table, complete the following steps:

Note

This scan generates a **PartiQL query** and requires that you have the correct AWS Identity and Access Management (AWS IAM) policies in place. To learn more about the PartiQL security policy requirements, see the [IAM security policies with PartiQL for DynamoDB](#) topic in the *Amazon DynamoDB Developer Guide*.

1. From the AWS Explorer, double-click a table to view it in the JetBrains **Editor** window.
2. From the table view, expand the **Scan** header.

3. Select the **Table/ Index** you want to scan from the drop-down menu.
4. Choose **Run** to proceed with the scan, the scan is complete when the table data is returned in the **Editor** window.

Working with Amazon Elastic Container Service by Using the AWS Toolkit for JetBrains

The following topics describe how to use the AWS Toolkit for JetBrains to work with Amazon ECS resources in an AWS account.

Topics

- [Amazon Elastic Container Service \(Amazon ECS\) Exec in AWS Toolkit](#)

Amazon Elastic Container Service (Amazon ECS) Exec in AWS Toolkit

You can use the Amazon ECS Exec feature to issue single commands or run a shell in an Amazon Elastic Container Service (Amazon ECS) container, directly from the AWS Toolkit.

Important

Enabling and Disabling Amazon ECS Exec changes the state of resources in your AWS account. This includes stopping and restarting the service. Altering the state of resources while the Amazon ECS Exec is enabled can lead to unpredictable results. For more information about Amazon ECS Exec, see the developer guide [Using Amazon ECS Exec for Debugging](#).

Amazon ECS Exec prerequisites

Before you can use the Amazon ECS Exec feature, there are prerequisite conditions that need to be met.

Important

In order to enable Amazon ECS Exec for a particular service, Amazon ECS Cloud Debugging must be disabled for that service.

Amazon ECS requirements

Depending on whether your tasks are hosted on Amazon EC2 or AWS Fargate (Fargate), Amazon ECS Exec has different version requirements.

- If you're using Amazon EC2, you must use an Amazon ECS optimized AMI that was released after January 20th, 2021, with an agent version of 1.50.2 or greater. Additional information is available for you in the developer guide [Amazon ECS optimized AMIs](#).
- If you're using AWS Fargate, you must use platform version 1.4.0 or higher. Additional information about Fargate requirements is available to you in the developer guide [AWS Fargate platform versions](#).

AWS account configuration and IAM permissions

To use the Amazon ECS Exec feature, you need to have an existing Amazon ECS cluster associated with your AWS account. Amazon ECS Exec uses Systems Manager to establish a connection with the containers on your cluster and requires specific Task IAM Role Permissions to communicate with the SSM service.

You can find IAM role and policy information, specific to Amazon ECS Exec, in the [IAM permissions required for ECS Exec](#) developer guide.

Working with the Amazon ECS Exec

You can enable or disable Amazon ECS Exec directly from the AWS Explorer in the AWS Toolkit for JetBrains. When Amazon ECS Exec is enabled, you can choose containers from the Amazon ECS menu and then run commands against them.

Enabling Amazon ECS Exec

1. From the AWS Explorer, expand the Amazon ECS menu.
2. Expand the **Clusters** section, and choose the cluster you want to modify.
3. Open the context menu for (right-click) the service you want to modify and choose **Enable Command Execution**.

Note

If Amazon ECS Cloud Debugging is enabled for this service, the **Enable Command Execution** option will not be available. Disabling Cloud Debugging will restore the option, but it will stop and restart your service.

Important

This will start a new deployment of your Service and may take a few minutes. For more information, see the note at the beginning of this section.)

Disabling Amazon ECS Exec

1. From the AWS Explorer, expand the Amazon ECS menu.
2. Expand the **Clusters** section, and choose the cluster you want to modify.
3. Open the context menu for (right-click) the service you want to modify and choose **Disable Command Execution**.

Important

This will start a new deployment of your Service and may take a few minutes. For more information, see the note at the beginning of this section.

Running commands against a Container

To run commands against a container using the AWS Explorer, Amazon ECS Exec must be enabled. If it's not enabled, see the **Enabling Amazon ECS Exec** procedure in this section.

1. From the AWS Explorer, expand the Amazon ECS menu.
2. Expand the **Clusters** section, and choose the cluster you want to modify.
3. Expand a service to list its containers.
4. Open the context menu for (right-click) the container you want to modify and choose **Run Command in Container**.

5. In the **Run Command in Container** dialog box, choose the **Task ARN** that you want.
6. You can type the command you want to run or select it from a list of commands that were run during the same session.
7. Choose **Execute**

Running commands from within a shell

To run commands against a container from within a shell, using the AWS Explorer, Amazon ECS Exec must be enabled. If it's not enabled, see the **Enabling Amazon ECS Exec** procedure in this section.

1. From the AWS Explorer, expand the Amazon ECS menu.
2. Expand the **Clusters** section, and choose the cluster you want to modify.
3. Expand the service to list its containers.
4. Open the context menu for (right-click) the container you want to modify and choose **Open Interactive Shell**.
5. In the **Interactive Shell** dialog box, choose the **Task ARN** that you want.
6. Choose a shell from the corresponding drop down, or enter the name of the shell you want to interact with.
7. When you are satisfied with your settings, choose **Execute**.
8. When the shell opens in a terminal, you can enter commands to interact with the container.

Working with Amazon EventBridge by using the AWS Toolkit for JetBrains

The following topic describes how to use the AWS Toolkit for JetBrains to work with Amazon EventBridge schemas in an AWS account.

Topics

- [Working with Amazon EventBridge schemas](#)

Working with Amazon EventBridge schemas

You can use the AWS Toolkit for JetBrains to work with Amazon EventBridge Schemas as follows.

Note

Working with EventBridge Schemas is currently supported only by the AWS Toolkit for IntelliJ and the AWS Toolkit for PyCharm.

The following information assumes you have already [set up the AWS Toolkit for JetBrains](#).

Contents

- [View an available schema](#)
- [Find an available schema](#)
- [Generate code for an available schema](#)
- [Create an AWS Serverless Application Model application that uses an available schema](#)

View an available schema

1. With the [AWS Explorer](#) tool window displayed, expand **Schemas**.
2. Expand the name of the registry that contains the schema you want to view. For example, many of the schemas that AWS supplies are in the **aws.events** registry.
3. To view the schema in the editor, right-click the title of the schema, and on the context menu, choose **View Schema**.

Find an available schema

With the [AWS Explorer](#) tool window displayed, do one of the following:

- Begin typing the title of the schema you want to find. The **AWS Explorer** highlights the titles of schemas that contain a match.
- Right-click **Schemas**, and on the context menu, choose **Search Schemas**. In the **Search EventBridge Schemas** dialog box, begin typing the title of the schema you want to find. The dialog box displays the titles of schemas that contain a match.
- Expand **Schemas**. Right-click the name of the registry that contains the schema you want to find, and then choose **Search Schemas in Registry**. In the **Search EventBridge Schemas** dialog box, begin typing the title of the schema you want to find. The dialog box displays the titles of schemas that contain a match.

To view a schema in the list of matches, do one of the following:

- To display the schema in the editor, in **AWS Explorer**, right-click the title of the schema, and then choose **View Schema**.
- In the **Search EventBridge Schemas** dialog box, choose the title of the schema to display the schema.

Generate code for an available schema

1. With the [AWS Explorer](#) tool window displayed, expand **Schemas**.
2. Expand the name of the registry that contains the schema you want to generate code for.
3. Right-click the title of the schema, and then choose **Download code bindings**.
4. In the **Download code bindings** dialog box, choose the following:
 - The **Version** of the schema to generate code for.
 - The supported programming **Language** and language version to generate code for.
 - The **File location** where you want to store the generated code on the local development machine.
5. Choose **Download**.

Create an AWS Serverless Application Model application that uses an available schema

1. On the **File** menu, choose **New, Project**.
2. In the **New Project** dialog box, choose **AWS**.
3. Choose **AWS Serverless Application**, and then choose **Next**.
4. Specify the following:
 - A **Project name** for the project.
 - A **Project location** on your local development machine for the project.
 - A supported AWS Lambda **Runtime** for the project.
 - An AWS Serverless Application Model (AWS SAM) **SAM Template** for the project. The choices currently include the following:

- **AWS SAM EventBridge Hello World (EC2 Instance State Change)** – When deployed, creates an AWS Lambda function and an associated Amazon API Gateway endpoint in your AWS account. By default, this function and endpoint respond only to an Amazon EC2 instance status change.
- **AWS SAM EventBridge App from Scratch (for any Event trigger from a Schema Registry)** – When deployed, creates an AWS Lambda function and an associated Amazon API Gateway endpoint in your AWS account. This function and endpoint can respond to events that are available in the schema you specify.

If you choose this template, you must also specify the following:

- The named profile, **Credentials**, to use.
- The AWS **Region** to use.
- The EventBridge **Event Schema** to use.
- The version of the SDK to use for the project (**Project SDK**).

After you create an AWS serverless application project, you can do the following:

- [Deploy the application](#)
- [Change \(update\) the application's settings](#)
- [Delete the deployed application](#)

You can also do the following with Lambda functions that are part of the application:

- [Run \(invoke\) or debug the local version of a function](#)
- [Run \(invoke\) the remote version of a function](#)
- [Change a function's settings](#)
- [Delete a function](#)

Working with AWS Lambda from the AWS Toolkit for JetBrains

The following topics describe how to work with AWS Lambda functions from the AWS Toolkit for JetBrains.

Topics

- [AWS Lambda runtimes and support in the AWS Toolkit for JetBrains](#)
- [Creating an AWS Lambda function by using the AWS Toolkit for JetBrains](#)
- [Running \(invoking\) or debugging the local version of an AWS Lambda function by using the AWS Toolkit for JetBrains](#)
- [Running \(invoking\) the remote version of an AWS Lambda function by using the AWS Toolkit for JetBrains](#)
- [Changing \(updating\) AWS Lambda function settings by using the AWS Toolkit for JetBrains](#)
- [Deleting an AWS Lambda function by using the AWS Toolkit for JetBrains](#)

AWS Lambda runtimes and support in the AWS Toolkit for JetBrains

AWS Lambda supports multiple languages through the use of runtimes. A runtime provides a language-specific environment that relays invocation events, context information, and responses between Lambda and the function. For detailed information about the Lambda service and supported runtimes, see the [Lambda runtimes](#) topic in the *AWS Lambda User Guide*.

The following describes runtime environments currently supported for use with the AWS Toolkit for JetBrains.

Name	Identifier	Operating System	Architecture
Node.js 18	nodejs18.x	Amazon Linux 2	x86_64, arm64
Node.js 16	nodejs16.x	Amazon Linux 2	x86_64, arm64
Node.js 14	nodejs14.x	Amazon Linux 2	x86_64, arm64
Python 3.11	python3.11	Amazon Linux 2	x86_64, arm64
Python 3.10	python3.10	Amazon Linux 2	x86_64, arm64
Python 3.9	python3.9	Amazon Linux 2	x86_64, arm64
Python 3.8	python3.8	Amazon Linux 2	x86_64, arm64
Python 3.7	python3.7	Amazon Linux 2	x86_64

Name	Identifier	Operating System	Architecture
Java 17	java17	Amazon Linux 2	x86_64, arm64
Java 11	java11	Amazon Linux 2	x86_64, arm64
Java 8	java8.al2	Amazon Linux 2	x86_64, arm64
Java 8	java8	Amazon Linux 2	x86_64
.NET 6	dotnet6	Amazon Linux 2	x86_64, arm64
Go 1.x	go1.x	Amazon Linux 2	x86_64

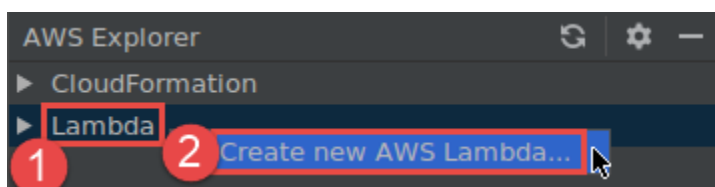
Creating an AWS Lambda function by using the AWS Toolkit for JetBrains

You can use the AWS Toolkit for JetBrains to create an AWS Lambda function that is part of an AWS serverless application. Or you can create a standalone Lambda function.

To create a Lambda function that is part of an AWS serverless application, skip the rest of this topic and see [Creating an application](#) instead.

To create a standalone Lambda function, you must first install the AWS Toolkit for JetBrains and, if you haven't yet, connect to an AWS account for the first time. Then, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

- Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region to create the function in, do that now. Then right-click **Lambda**, and choose **Create new AWS Lambda**.

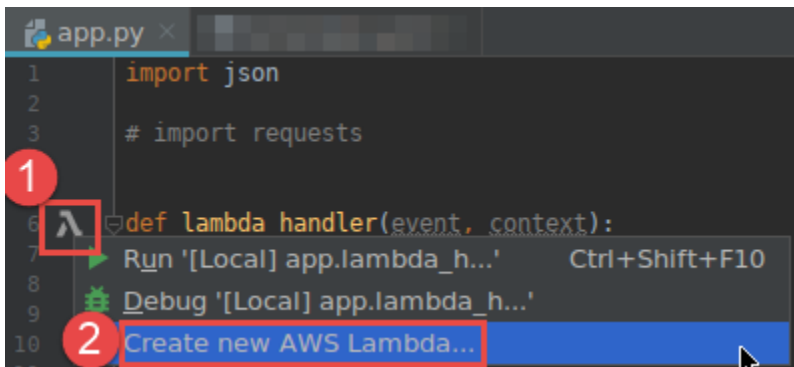


Complete the [Create Function](#) dialog box, and then choose **Create Function**. The AWS Toolkit for JetBrains creates a corresponding AWS CloudFormation stack for the deployment, and adds

the function name to the **Lambda** list in **AWS Explorer**. If the deployment fails, you can try to determine why by viewing event logs for the stack.

- Create a code file that implements a function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#).

If you need to switch to a different AWS Region to create the remote function to be run (invoked), do that now. Then in the code file, choose the **Lambda** icon in the gutter next to the function handler, and then choose **Create new AWS Lambda**. Complete the [Create Function](#) dialog box, and then choose **Create Function**.

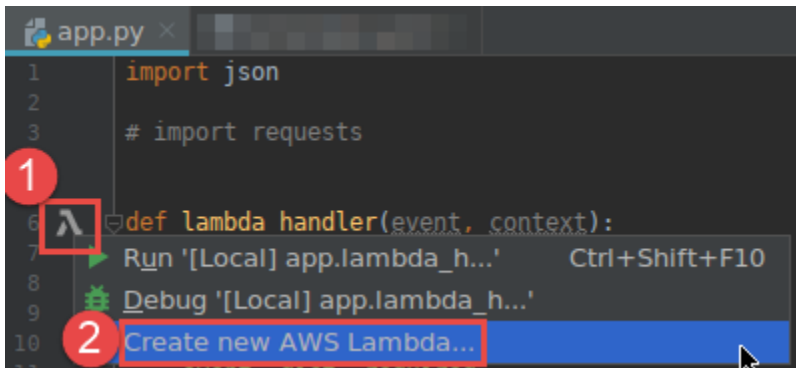


Note

If the **Lambda** icon isn't displayed in the gutter next to the function handler, try displaying it for the current project by selecting the following box in **Settings/Preferences: Tools, AWS, Project settings, Show gutter icons for all potential AWS Lambda handlers**. Also, if the function handler is already defined in the corresponding AWS SAM template, the **Create new AWS Lambda** command won't appear.

After you choose **Create Function**, the AWS Toolkit for JetBrains creates a corresponding function in the Lambda service for the connected AWS account. If the operation succeeds, after you refresh **AWS Explorer**, the **Lambda** list displays the name of the new function.

- If you already have a project that contains an AWS Lambda function, and if you need to first switch to a different AWS Region to create the function in, do that now. Then in the code file that contains the function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#), choose the **Lambda** icon in the gutter next to the function handler. Choose **Create new AWS Lambda**, complete the [Create Function](#) dialog box, and then choose **Create Function**.



Note

If the **Lambda** icon isn't displayed in the gutter next to the function handler, try displaying it for the current project by selecting the following box in **Settings/Preferences: Tools, AWS, Project settings, Show gutter icons for all potential AWS Lambda handlers**. Also, the **Create new AWS Lambda** command won't be displayed if the function handler is already defined in the corresponding AWS SAM template.

After you choose **Create Function**, the AWS Toolkit for JetBrains creates a corresponding function in the Lambda service for the connected AWS account. If the operation succeeds, after you refresh **AWS Explorer**, the new function's name appears in the **Lambda** list.

After you create the function, you can run (invoke) or debug the local version of the function or run (invoke) the remote version.

Running (invoking) or debugging the local version of an AWS Lambda function by using the AWS Toolkit for JetBrains

To complete this procedure, you must create the AWS Lambda function that you want to run (invoke) or debug, if you have not created it already.

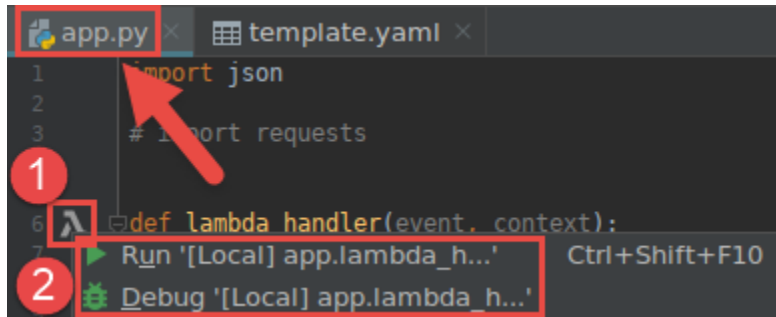
Note

To run (invoke) or debug the local version of a Lambda function, and run (invoke) or debug that function locally with any nondefault or optional properties, you must first set those properties in the function's corresponding AWS SAM template file (for example, in a file named `template.yaml` within the project). For a list of available properties,

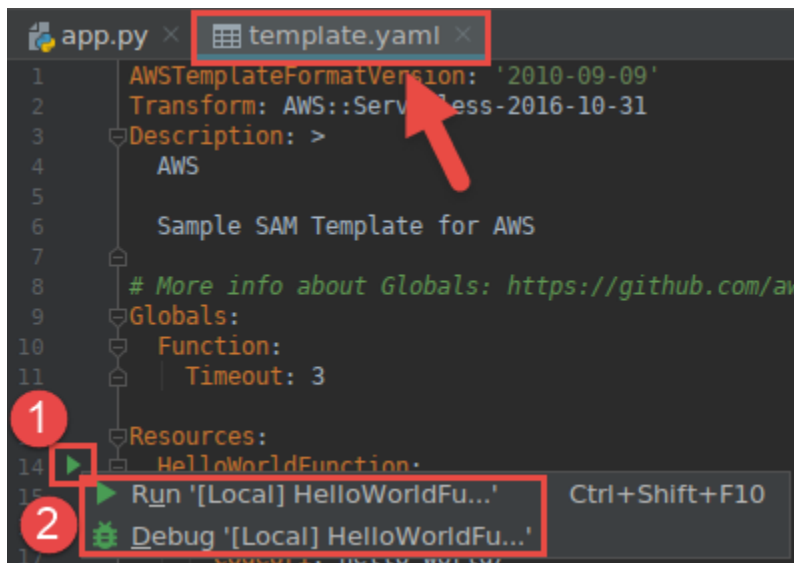
see [AWS::Serverless::Function](#) in the [awslabs/serverless-application-model](#) repository on GitHub.

1. Do one of the following:

- In the code file that contains the function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#), choose the Lambda icon in the gutter next to the function handler. Choose **Run '[Local]'** or **Debug '[Local]'**.



- With the **Project** tool window already open and displaying the project that contains the function, open the project's `template.yaml` file. Choose the **Run** icon in the gutter next to the function's resource definition, and then choose **Run '[Local]'** or **Debug '[Local]'**.



- ## 2. Complete the [Edit configuration \(local function settings\)](#) dialog box if it's displayed, and then choose **Run** or **Debug**. Results are displayed in the **Run** or **Debug** tool window.
- If the **Edit configuration** dialog box doesn't appear and you want to change the existing configuration, first change its configuration, and then repeat this procedure from the beginning.

- If the configuration details are missing, expand **Templates**, **AWS Lambda**, and then choose **Local**. Choose **OK**, and then repeat this procedure from the beginning.

Running (invoking) the remote version of an AWS Lambda function by using the AWS Toolkit for JetBrains

A *remote* version of an AWS Lambda function is a function whose source code already exists inside of the Lambda service for an AWS account.

To complete this procedure, you must first install the AWS Toolkit for JetBrains and, if you haven't yet, connect to an AWS account for the first time. Then with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider running, do the following.

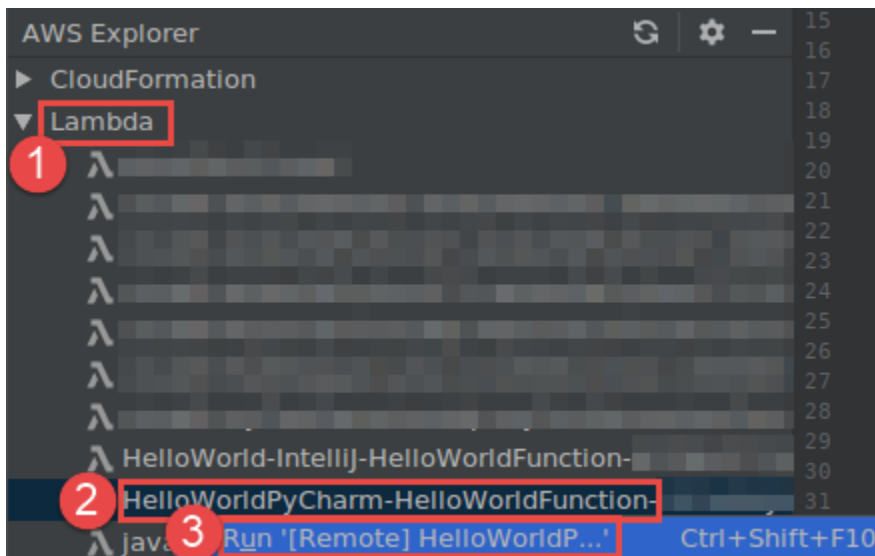
1. Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region that contains the function, do that now.
2. Expand **Lambda**, and confirm that the name of the function is listed. If it is, skip ahead to step 3 in this procedure.

If the name of the function isn't listed, create the Lambda function that you want to run (invoke).

If you created the function as part of an AWS serverless application, you must also deploy that application.

If you created the function by creating a code file that implements a function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#), then in the code file, choose the Lambda icon next to the function handler. Then choose **Create new AWS Lambda**. Complete the [Create Function](#) dialog box, and then choose **Create Function**.

3. With **Lambda** open in **AWS Explorer**, right-click the name of the function, and then choose **Run '[Remote]'**.

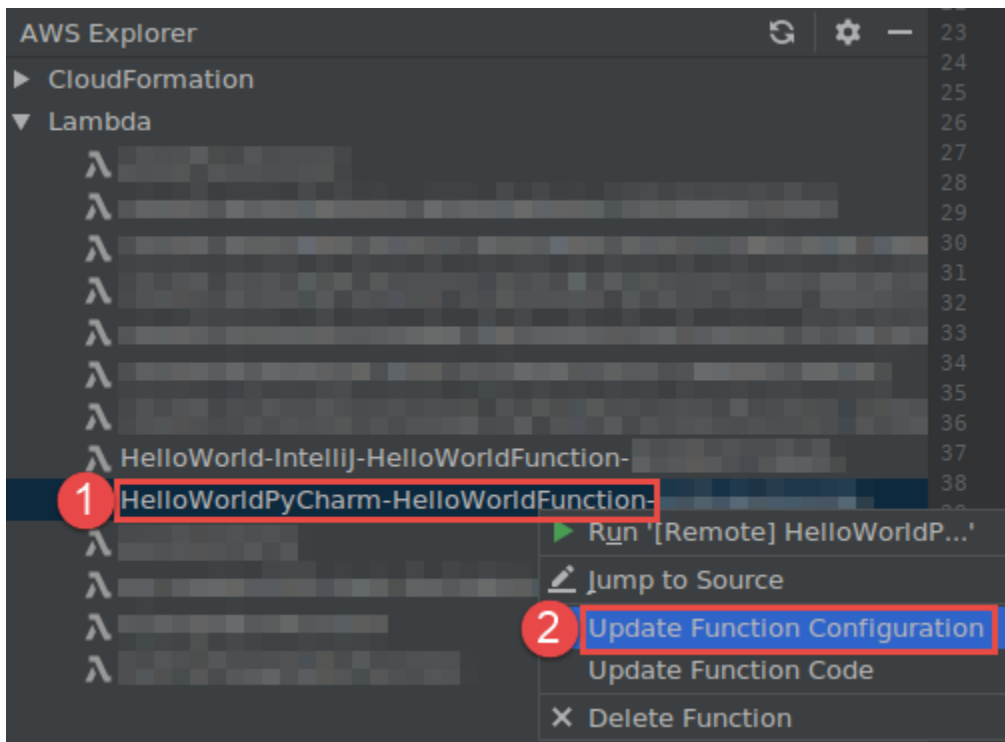


4. Complete the [Edit configuration \(remote function settings\)](#) dialog box if it's displayed, and then choose **Run** or **Debug**. Results are displayed in the **Run** or **Debug** tool window.
 - If the **Edit configuration** dialog box doesn't appear and you want to change the existing configuration, first change its configuration, and then repeat this procedure from the beginning.
 - If the configuration details are missing, expand **Templates**, **AWS Lambda**, and then choose **Local**. Choose **OK**, and then repeat this procedure from the beginning.

Changing (updating) AWS Lambda function settings by using the AWS Toolkit for JetBrains

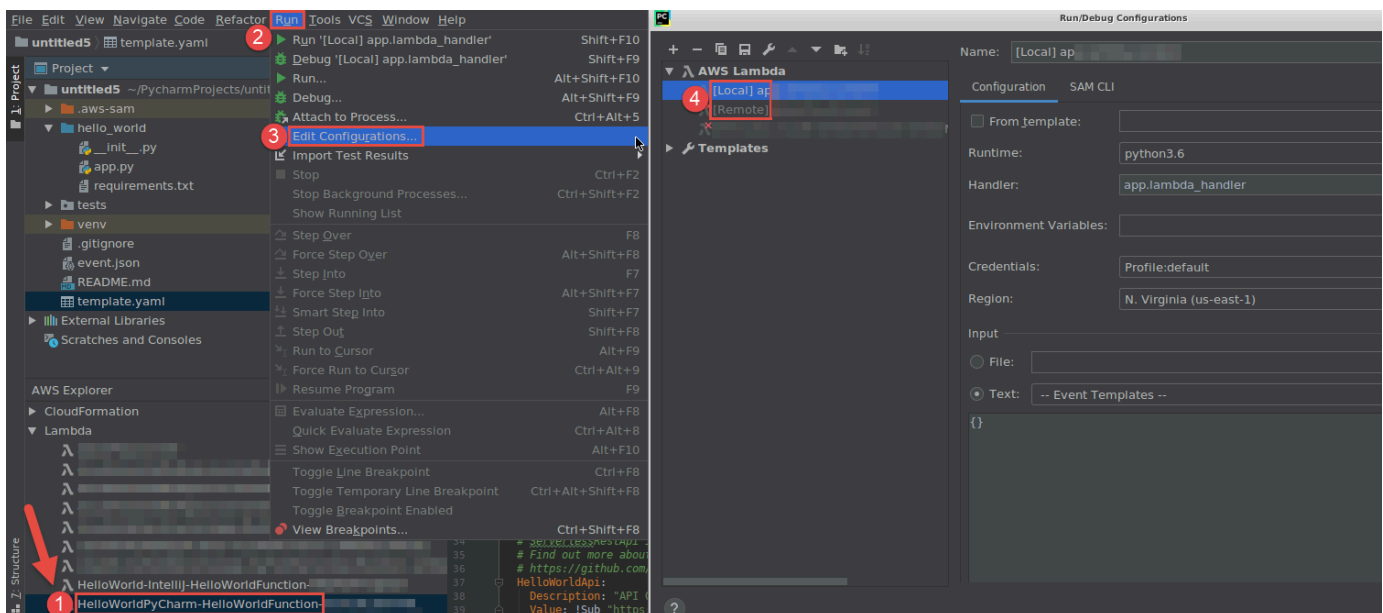
To use the AWS Toolkit for JetBrains to change (update) the settings for an AWS Lambda function, do one of the following.

- With the code file open that contains the function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#), on the main menu, choose **Run, Edit Configurations**. Complete the [Run/Debug Configurations](#) dialog box, and then choose **OK**.
- Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region that contains the function, do that now. Expand **Lambda**, choose the name of the function to change the configuration for, and then do one of the following:
 - **Change settings such as the timeout, memory, environment variables, and execution role** – Right-click the name of the function, and then choose **Update Function Configuration**.



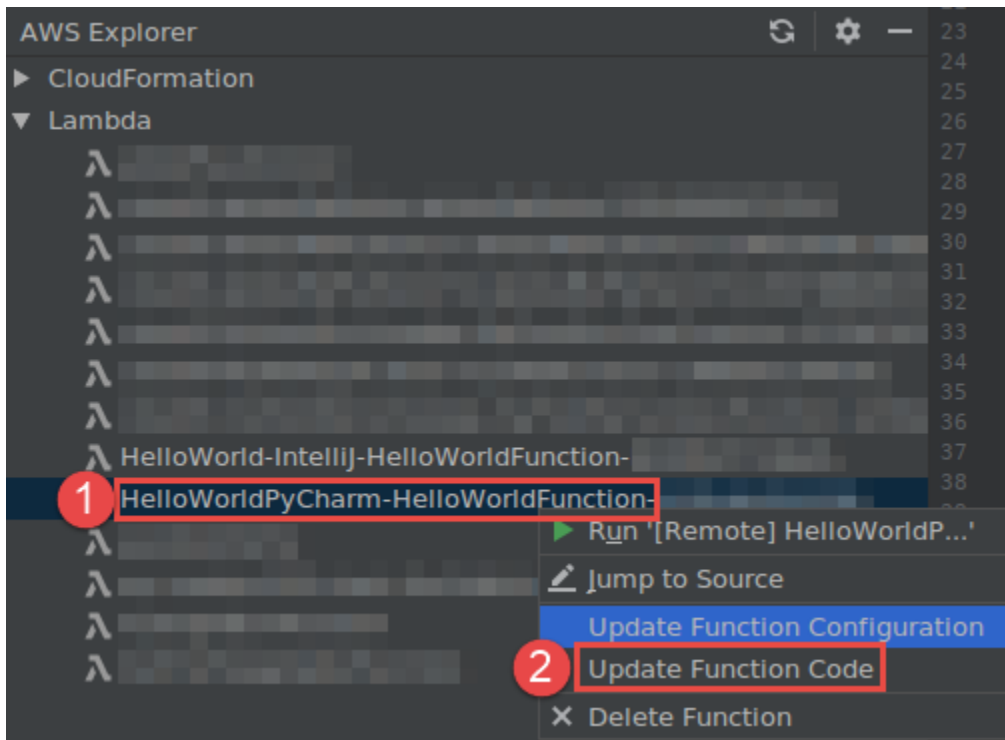
Complete the [Update Configuration](#) dialog box, and then choose **Update**.

- **Change settings such as the input payload** – On the main menu, choose **Run, Edit Configurations**. Complete the [Run/Debug Configurations](#) dialog box, and then choose **OK**.



If the configuration details are missing, first expand **Templates, AWS Lambda**, and then choose **Local** (for the local version of the function) or **Remote** (for the remote version of that same function). Choose **OK**, and then repeat this procedure from the beginning.)

- **Change settings such as the function handler name or Amazon Simple Storage Service (Amazon S3) source bucket** – Right-click the function name, and then choose **Update Function Code**.



Complete the [Update Code](#) dialog box, and then choose **Update**.

- **Change other available property settings that aren't listed in the preceding bullets** – Change those settings in the function's corresponding AWS SAM template file (for example, in a file named `template.yaml` within the project).

For a list of available property settings, see [AWS::Serverless::Function](#) in the [awslabs/serverless-application-model](#) repository on GitHub.

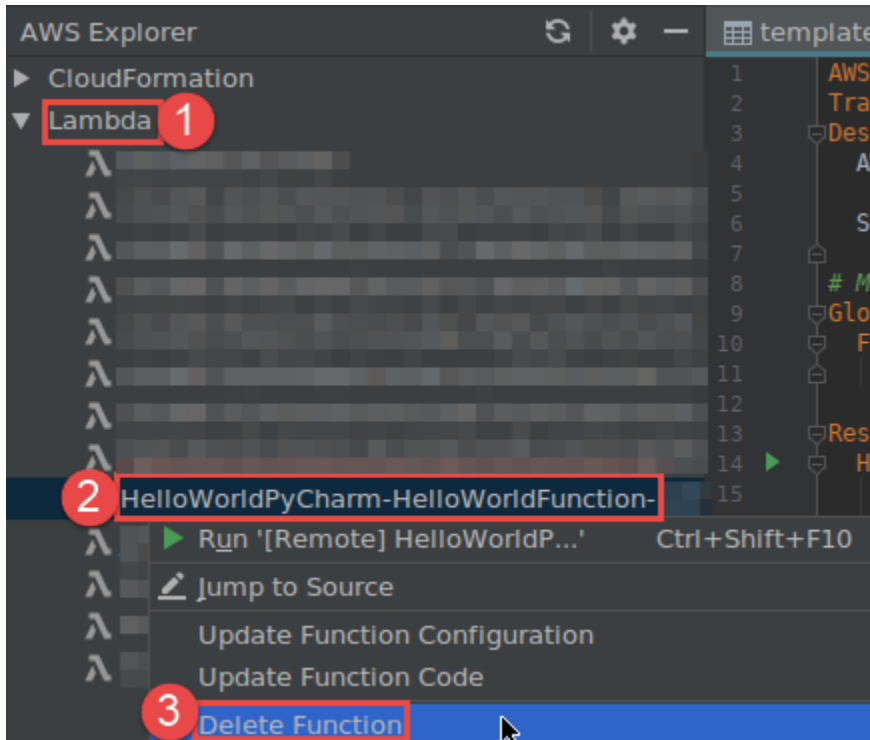
Deleting an AWS Lambda function by using the AWS Toolkit for JetBrains

You can use the AWS Toolkit to delete an AWS Lambda function that is part of an AWS serverless application, or you can delete a standalone Lambda function.

To delete a Lambda function that is part of an AWS serverless application, skip the rest of this topic and see [Deleting an application](#) instead.

To delete a standalone Lambda function, do the following.

1. Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region that contains the function, do that now.
2. Expand **Lambda**.
3. Right-click the name of the function to delete, and then choose **Delete Function**.



4. Enter the function's name to confirm the deletion, and then choose **OK**. If the function deletion succeeds, the AWS Toolkit for JetBrains removes the function name from the **Lambda** list.

Accessing Amazon RDS by using the AWS Toolkit for JetBrains

Using Amazon Relational Database Service (Amazon RDS), you can provision and manage SQL relational database systems in the cloud. Using AWS Toolkit for JetBrains, you can connect to and interact with the following Amazon RDS database engines:

- Aurora – A MySQL and PostgreSQL-compatible relational database built for the cloud. For more information, see the [Amazon Aurora User Guide](#).
- MySQL – Amazon RDS supports several major versions of the open-source relational database. For more information, see [MySQL on Amazon RDS](#) in the *Amazon RDS User Guide*.

- PostgreSQL – Amazon RDS supports several major version of the open-source object-relational database. For more information, see [PostgreSQL on Amazon RDS](#) in the *Amazon RDS User Guide*.

The following topics describe the prerequisites for accessing RDS databases and how to use AWS Toolkit for JetBrains to connect to a database instance.

Topics

- [Prerequisites for accessing Amazon RDS databases](#)
- [Connecting to an Amazon RDS database](#)

Prerequisites for accessing Amazon RDS databases

Before you can connect to an Amazon RDS database using AWS Toolkit for JetBrains, you need to complete the following tasks:

- [Create a DB instance and set up its authentication method](#)
- [Download and install DataGrip](#)

Creating an Amazon RDS DB instance and configuring an authentication method

AWS Toolkit for JetBrains enables you to connect to an Amazon RDS DB instance that's already been created and configured in AWS. A DB instance is an isolated database environment running in the cloud that can contain multiple user-created databases. For information about creating DB instances for the supported database engines, see [Getting started with Amazon RDS resources](#) in the *Amazon RDS User Guide*.

When connecting to a database using AWS Toolkit for JetBrains, users can choose to authenticate using IAM credentials or Secrets Manager. The following table describes key features and information resources for both options:

Authentication methods	How it works	More information
Connect with IAM credentials	With IAM database authentication, you don't need to store user credentials in the database because authentic	<ul style="list-style-type: none"> • Identity and access management in Amazon RDS in the <i>Amazon RDS User Guide</i>.

Authentication methods	How it works	More information
	<p>ation is managed externally using AWS Identity and Access Management (IAM) credentials.</p> <p>By default, IAM database authentication is disabled on DB instances. You can enable IAM database authentication (or disable it again) using the AWS Management Console, AWS CLI, or the API.</p>	<ul style="list-style-type: none"> • AWS Knowledge Center article: How do I allow users to authenticate to an Amazon RDS MySQL DB instance using their IAM credentials?
Connect with AWS Secrets Manager	<p>A database administrator can store credentials for a database as a secret in Secrets Manager. Secrets Manager encrypts and stores the credentials within the secret as the <i>protected secret text</i>.</p> <p>When an application with permissions accesses the database, Secrets Manager decrypts the protected secret text and returns it over a secured channel. The client parses the returned credentials, connection string, and any other required information and then uses that information to access the database.</p>	<ul style="list-style-type: none"> • What is AWS Secrets Manager? in the <i>AWS Secrets Manager User Guide</i>. • Tutorial: Rotating a secret for an AWS database in the <i>AWS Secrets Manager User Guide</i>. • AWS Security Blog: Rotate Amazon RDS database credentials automatically with Secrets Manager.

Working with Amazon RDS databases using DataGrip

After you've connected to an Amazon RDS data source, you can start interacting with it. By using DataGrip from JetBrains, you can carry out database tasks such as writing SQL, running queries, and importing/exporting data. Features provided by DataGrip are also available in the database plugin for a range of JetBrains IDEs. For information about DataGrip, see <https://www.jetbrains.com/datagrip/>.

Connecting to an Amazon RDS database

With **AWS Explorer**, you can select an Amazon RDS database, choose an authentication method, and then configure the connection settings. After you've successfully tested the connection, you can start interacting with the data source using JetBrains DataGrip.

Important

Ensure that you've completed the [prerequisites](#) to enable users to access and interact with Amazon RDS databases.

Select a tab for instructions on connecting to a database instance using your preferred authentication method.

Connect with IAM credentials

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon RDS** node to expand the list of supported database engines.
3. Click a supported database engine (Aurora, MySQL, or PostgreSQL) node to expand the list of available database instances.

Note

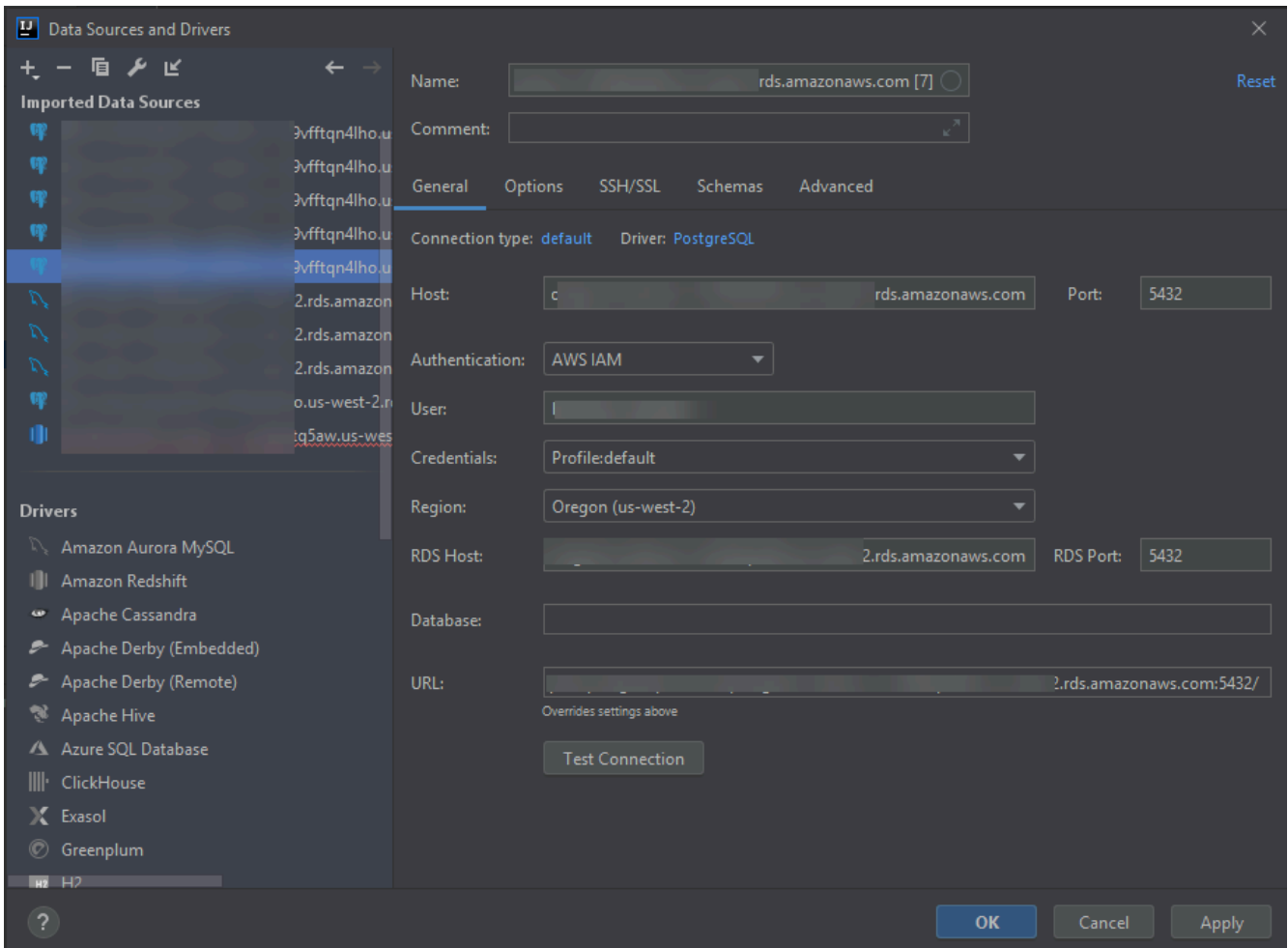
If you select Aurora, you can choose between expanding a MySQL cluster and a PostgreSQL cluster.

4. Right-click a database and choose **Connect with IAM credentials**.

Note

You can also choose **Copy Arn** to add the database's Amazon Resource Name (ARN) to your clipboard.

5. In the **Data Sources and Drivers** dialog box, do the following to ensure a database connection can be opened:
 - In the **Imported Data Sources** pane, confirm that the correct the correct data source is selected.
 - If a message indicates that you need to **Download missing driver files**, choose **Go to Driver** (the wrench icon) to download the required files.
6. In the **General** tab of the **Settings** pane, confirm that the following fields display the correct values:
 - **Host/Port** – The endpoint and port used for connections to the database. For Amazon RDS databases hosted in the AWS Cloud, endpoints always end with `rds.amazonaws.com`. If you're connecting to a DB instance through a proxy, use these fields to specify the proxy's connection details.
 - **Authentication – AWS IAM** (authentication using IAM credentials).
 - **User** – The name of your database user account.
 - **Credentials** – The credentials used to access your AWS account.
 - **Region** – The AWS Region where the database is hosted.
 - **RDS Host/Port** – The endpoint and port for the database as listed in the AWS Management Console. If you're using a different endpoint to connect to a DB instance, specify the proxy's connection details in the **Host/Port** fields (described previously).
 - **Database** – The name of the database.
 - **URL** – The URL that the JetBrains IDE will use to connect to the database.



Note

For a full description of the connection settings that you can configure using the **Data sources and drivers** dialog box, see the [documentation for the JetBrains IDE](#) that you're using.

7. To verify the connection settings are correct, choose **Test Connection**.

A green check mark indicates a successful test.

8. Choose **Apply** to apply your settings, and then choose **OK** to start working with the data source.

The **Database** tool window opens. This displays the available data sources as a tree with nodes representing database elements such as schemas, tables, and keys.

⚠ Important

To use the **Database** tool window, you must first download and install DataGrip from JetBrains. For more information, see <https://www.jetbrains.com/datagrip/>.

Connect with Secrets Manager

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon RDS** node to expand the list of supported database engines.
3. Click a supported database engine (Aurora, MySQL, or PostgreSQL) node to expand the list of available database instances.

📘 Note

If you select Aurora, you can choose between expanding a MySQL cluster and a PostgreSQL cluster.


4. Right-click a database and choose **Connect with Secrets Manager**.

📘 Note

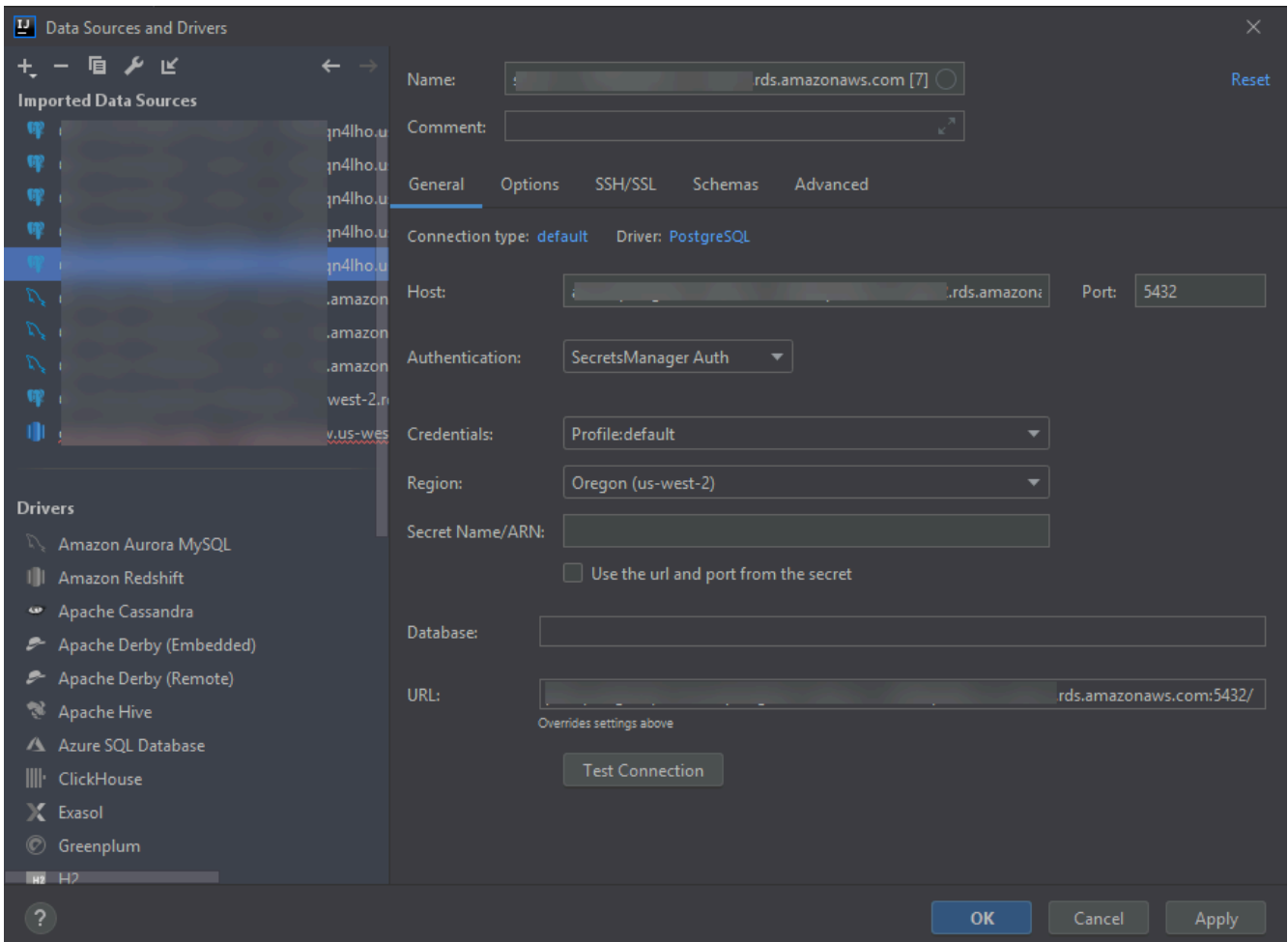
You can also choose **Copy Arn** to add the database's Amazon Resource Name (ARN) to your clipboard.

5. In the **Select a Database Secret** dialog box, use the drop-down field to pick credentials for the database, and then choose **Create**.
6. In the **Data Sources and Drivers** dialog box, do the following to ensure a database connection can be opened:
 - In the **Imported Data Sources** pane, confirm that the correct the correct data source is selected.
 - If a message indicates that you need to **Download missing driver files**, choose **Go to Driver** (the wrench icon) to download the required files.
7. In the **General** tab of the **Settings** pane, confirm that the following fields display the correct values:

- **Host/Port** – The endpoint and port used for connections to the database. For Amazon RDS databases hosted in the AWS Cloud, endpoints always end with `rds.amazonaws.com`. If you're connecting to a database through a proxy database, use these fields to specify the proxy's connection details.
- **Authentication – SecretsManager Auth** (authentication using AWS Secrets Manager).
- **Credentials** – The credentials used to access your AWS account.
- **Region** – The AWS Region where the database is hosted.
- **Secret Name/ARN** – The name and ARN of the secret containing authentication credentials. To override the connection settings in the **Host/Port** fields, select the **Use the url and port from the secret** check box.
- **Database** – The name of the database instance you selected in **AWS Explorer**.
- **URL** – The URL that the JetBrains IDE will use to connect to the database.

 **Note**

If you're using Secrets Manager for authentication, there are no fields for a user name and password for the database. This information is contained in the encrypted secret data portion of a secret.



Note

For a full description of the connection settings that you can configure using the **Data sources and drivers** dialog box, see the [documentation for the JetBrains IDE](#) that you're using.

- To verify the connection settings are correct, choose **Test Connection**.

A green check mark indicates a successful test.

- Choose **Apply** to apply your settings, and then choose **OK** to start working with the data source.

The **Database** tool window opens. This displays the available data sources as a tree with nodes representing database elements such as schemas, tables, and keys.

⚠ Important

To use the **Database** tool window, you must first download and install DataGrip from JetBrains. For more information, see <https://www.jetbrains.com/datagrip/>.

Accessing Amazon Redshift by using the AWS Toolkit for JetBrains

An Amazon Redshift data warehouse is an enterprise-class relational database query and management system. With AWS Toolkit for JetBrains, you can connect to and interact with Amazon Redshift clusters. An Amazon Redshift cluster consists of a collection of nodes that enables clients to query databases hosted on that cluster.

The following topics describe the prerequisites for accessing Amazon Redshift clusters and how to use AWS Toolkit for JetBrains to connect to a database in a cluster.

Topics

- [Prerequisites for accessing Amazon Redshift clusters](#)
- [Connecting to an Amazon Redshift cluster](#)

Prerequisites for accessing Amazon Redshift clusters

Before you start can interacting with an Amazon Redshift cluster using AWS Toolkit for JetBrains, you need to complete the following tasks:

- [Create an Amazon Redshift cluster and set up its authentication method](#)
- [Download and install DataGrip](#)

Creating an Amazon Redshift cluster and configuring an authentication method

AWS Toolkit for JetBrains enables you to connect to an Amazon Redshift cluster that's already created and configured in AWS. Each cluster contains one or more databases. For information about creating and configuring Amazon Redshift clusters, see [Getting started with Amazon Redshift](#) in the *Amazon Redshift Getting Started Guide*.

When connecting to a cluster using AWS Toolkit for JetBrains, users can choose to authenticate using IAM credentials or AWS Secrets Manager. The following table describes key features and information resources for both options:

Authentication methods	How it works	More information
Connect with IAM credentials	<p>With IAM database authentication, you don't need to store user credentials in the database because authentication is managed externally using AWS Identity and Access Management (IAM) credentials.</p> <p>By default, IAM database authentication is disabled on database instances. You can enable IAM database authentication (or disable it again) using the AWS Management Console, AWS CLI, or the API.</p>	<ul style="list-style-type: none"> • Identity and access management in Amazon Redshift in the <i>Amazon Redshift Management Guide</i>.
Connect with AWS Secrets Manager;	<p>A database administrator can store credentials for a database as a secret in Secrets Manager. Secrets Manager encrypts and stores the credentials within the secret as the <i>protected secret text</i>.</p> <p>When an application with permissions accesses the database, Secrets Manager decrypts the protected secret text and returns it over a</p>	<ul style="list-style-type: none"> • What is AWS Secrets Manager? in the <i>AWS Secrets Manager User Guide</i>. • Rotating secrets for Amazon Redshift in the <i>AWS Secrets Manager User Guide</i>. • AWS Security Blog: How to rotate Amazon DocumentDB and Amazon Redshift credentials in Secrets Manager.

Authentication methods	How it works	More information
	secured channel. The client parses the returned credentials, connection string, and any other required information and then uses that information to access the database.	

Working with Amazon RDS databases using DataGrip

After you've connected to a database in Amazon Redshift cluster, you can start interacting with it. Using DataGrip from JetBrains, you can carry out database tasks such as writing SQL, running queries, and importing/exporting data. Features provided by DataGrip are also available in the database plugin for a range of JetBrains IDEs. For information about DataGrip, see <https://www.jetbrains.com/datagrip/>.

Connecting to an Amazon Redshift cluster

With **AWS Explorer**, you can select an Amazon Redshift cluster, choose an authentication method, and then configure the connection settings. After you've successfully tested the connection, you can start interacting with the data source using JetBrains DataGrip.

Important

Ensure that you've completed the [prerequisites](#) to enable users to access and interact with Amazon Redshift clusters and databases.

Select a tab for instructions on connecting to a cluster using your preferred authentication method.

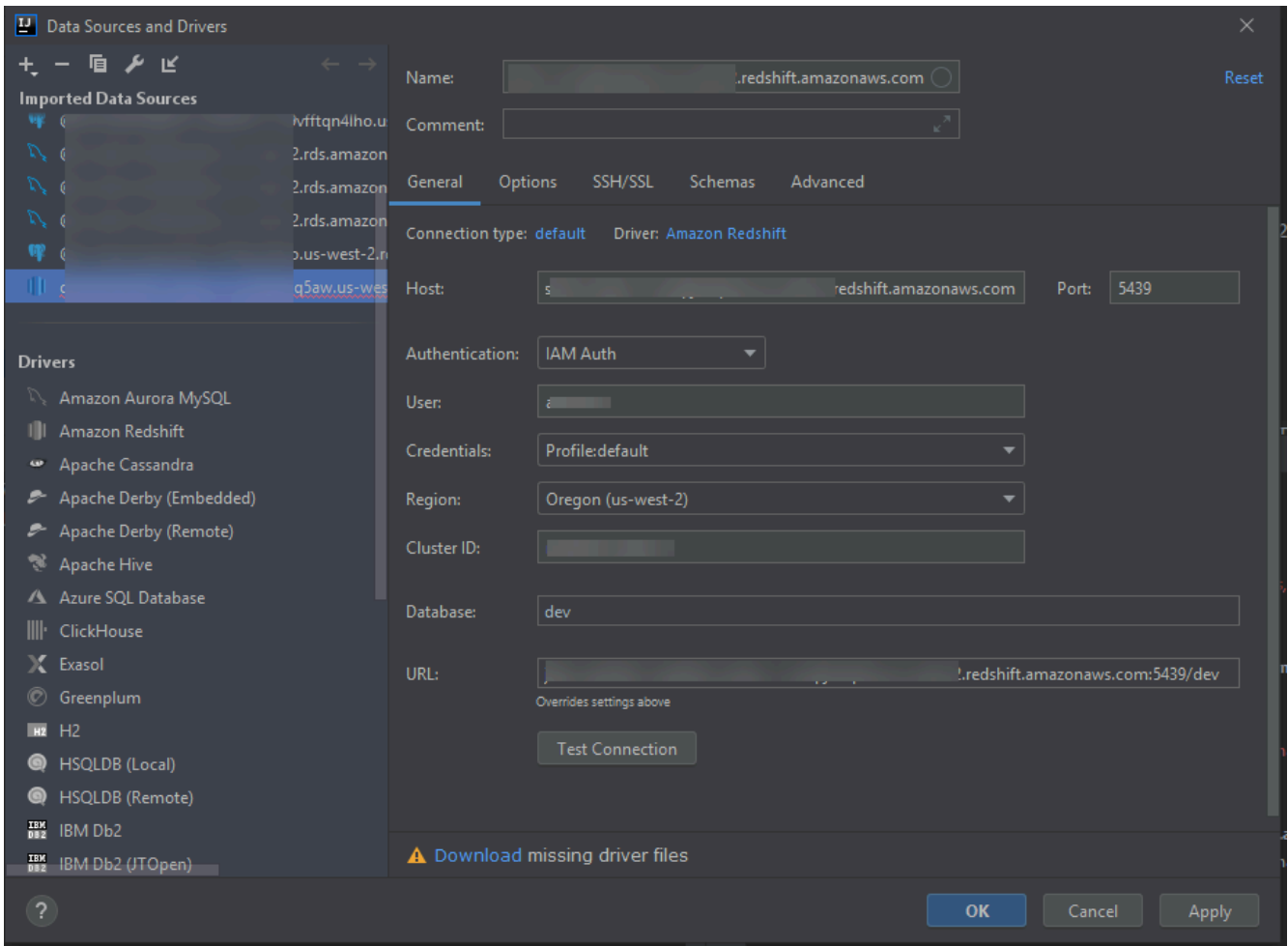
Connect with IAM credentials

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon Redshift** node to expand the list of available clusters.
3. Right-click a cluster and choose **Connect with IAM credentials**.

Note

You can also choose **Copy Arn** to add the cluster's Amazon Resource Name (ARN) to your clipboard.

4. In the **Data Sources and Drivers** dialog box, do the following to ensure a database connection can be opened:
 - In the **Imported Data Sources** pane, confirm that the correct data source is selected.
 - If a message indicates that you need to **Download missing driver files**, choose **Go to Driver** (the wrench icon) to download the required files.
5. On the **General** tab of the **Settings** pane, confirm that the following fields display the correct values:
 - **Host/Port** – The endpoint and port used for connections to the cluster. For Amazon Redshift clusters hosted in the AWS Cloud, endpoints always end with `redshift.amazonaws.com`.
 - **Authentication – AWS IAM** (authentication using IAM credentials).
 - **User** – The name of your database user account.
 - **Credentials** – The credentials used to access your AWS account.
 - **Region** – The AWS Region where the database is hosted.
 - **Cluster ID** – The ID of the cluster you selected in **AWS Explorer**.
 - **Database** – The name of the database in the cluster you'll connect to.
 - **URL** – The URL that the JetBrains IDE will use to connect to the cluster's database.



Note

For a full description of the connection settings that you can configure using the **Data sources and drivers** dialog box, see the [documentation for the JetBrains IDE](#) that you're using.

- To verify the connection settings are correct, choose **Test Connection**.

A green check mark indicates a successful test.

- Choose **Apply** to apply your settings, and then choose **OK** to start working with the data source.

The **Database** tool window opens. This displays the available data sources as a tree with nodes representing database elements such as schemas, tables, and keys.

⚠ Important

To use the **Database** tool window, you must first download and install DataGrip from JetBrains. For more information, see <https://www.jetbrains.com/datagrip/>.

Connect with Secrets Manager

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon Redshift** node to expand the list of available clusters.
3. Right-click a cluster and choose **Connect with Secrets Manager**.

ℹ Note

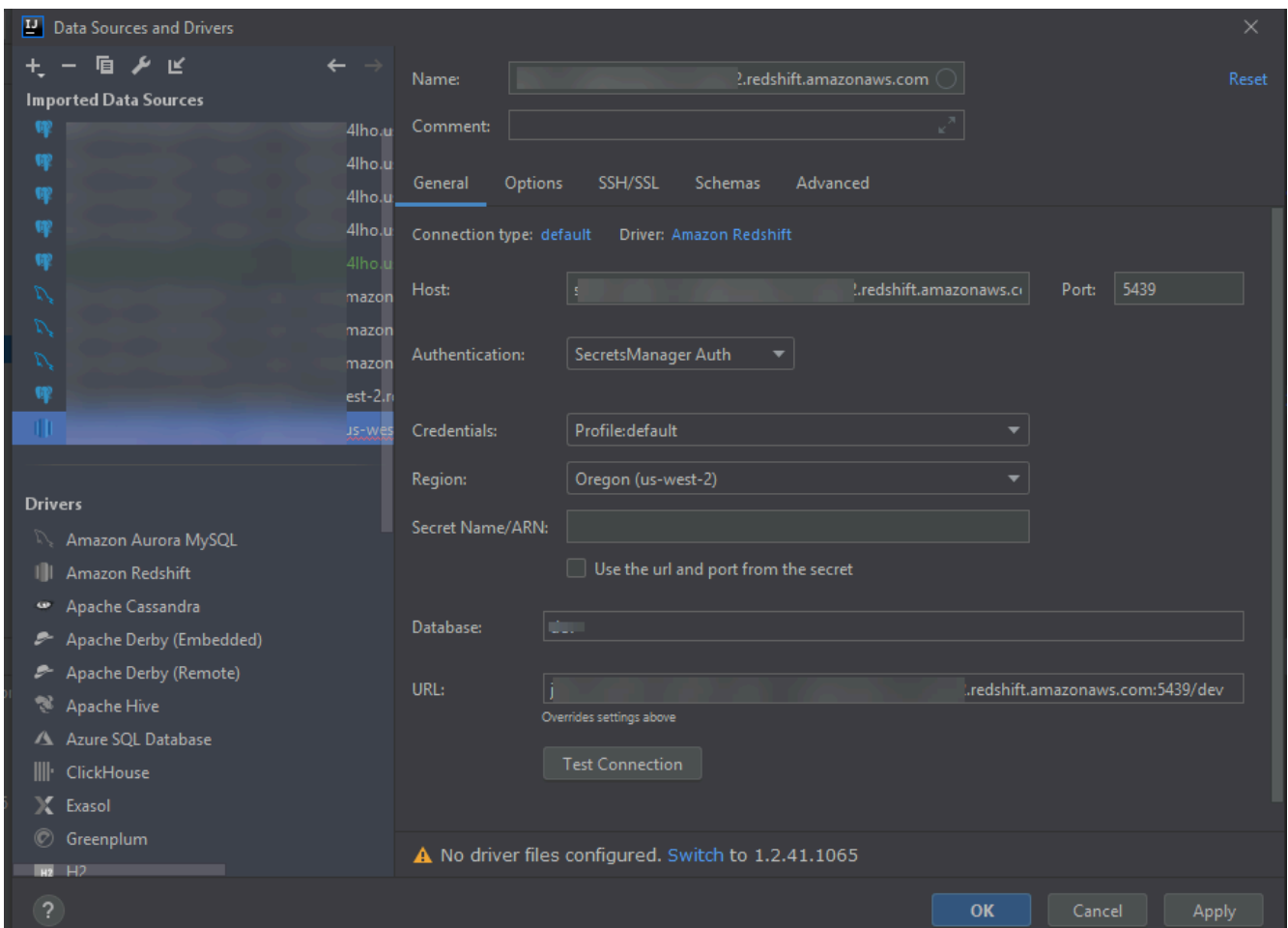
You can also choose **Copy Arn** to add the cluster's Amazon Resource Name (ARN) to your clipboard.

4. In the **Select a Database Secret** dialog box, use the drop-down field to pick credentials for the database, and then choose **Create**.
5. In the **Data Sources and Drivers** dialog box, do the following to ensure a database connection can be opened:
 - In the **Imported Data Sources**, confirm that the correct the correct data source is selected.
 - If a message appears in the dialog box to **Download missing driver files**, choose **Go to Driver** (the wrench icon) to download the required files.
6. On the **General** tab of the **Settings** pane, confirm that the following fields display the correct values:
 - **Host/Port** – The endpoint and port used for connections to the cluster. For Amazon Redshift clusters hosted in the AWS Cloud, endpoints always end with `redshift.amazonaws.com`.
 - **Authentication** – **SecretsManager Auth** (authentication using AWS Secrets Manager).
 - **Credentials** – The credentials used to connect to the AWS account.
 - **Region** – The AWS Region where the cluster is hosted.

- **Secret Name/ARN** – The name and ARN of the secret containing authentication credentials. If you want to override the connection settings in the **Host/Port** fields, select the **Use the url and port from the secret** check box.
- **Database** – The name of the database in the cluster you'll connect to.
- **URL** – The URL that the JetBrains IDE will use to connect to the database.

Note

If you're using AWS Secrets Manager for authentication, there are no fields for specifying a user name and password for the cluster. This information is contained in the encrypted secret data portion of a secret.



Note

For a full description of the connection settings that you can configure using the **Data sources and drivers** dialog box, see the [documentation for the JetBrains IDE](#) that you're using.

7. To verify the connection settings are correct, choose **Test Connection**.

A green check mark indicates a successful test.

8. Choose **Apply** to apply your settings, and then choose **OK** to start working with the data source.

The **Database** tool window opens. This displays the available data sources as a tree with nodes representing database elements such as schemas, tables, and keys.

Important

To use the **Database** tool window, you must first download and install DataGrip from JetBrains. For more information, see <https://www.jetbrains.com/datagrip/>.

Working with Amazon S3 by using the AWS Toolkit for JetBrains

The following topics describe how to use the AWS Toolkit for JetBrains to work with Amazon S3 buckets and objects in an AWS account.

Topics

- [Working with Amazon S3 buckets by using the AWS Toolkit for JetBrains](#)
- [Working with Amazon S3 objects by using the AWS Toolkit for JetBrains](#)

Working with Amazon S3 buckets by using the AWS Toolkit for JetBrains

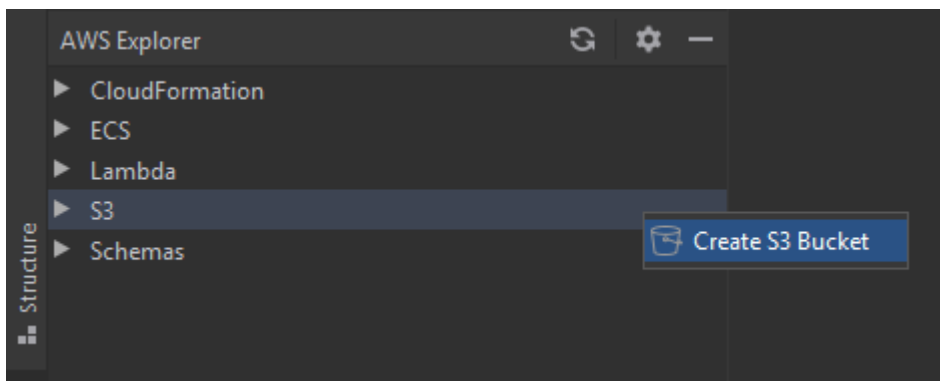
Every object you store in Amazon S3 resides in a bucket. You can use buckets to group related objects in the same way that you use a directory to group files in a file system.

Topics

- [Creating an Amazon S3 bucket](#)
- [Viewing Amazon S3 buckets](#)
- [Deleting an Amazon S3 bucket](#)

Creating an Amazon S3 bucket

1. Open AWS Explorer, if it isn't already open.
2. Right-click the **Amazon S3** node and choose **Create S3 Bucket**.



3. In the **Create S3 Bucket** dialog box, enter a name for the bucket.

Note

Because Amazon S3 allows your bucket to be used as a URL that can be accessed publicly, the bucket name that you choose must be globally unique. If some other account has already created a bucket with the name that you chose, you must use another name. For more information, see [Bucket Restrictions and Limitations](#) in the *Amazon Simple Storage Service User Guide*.

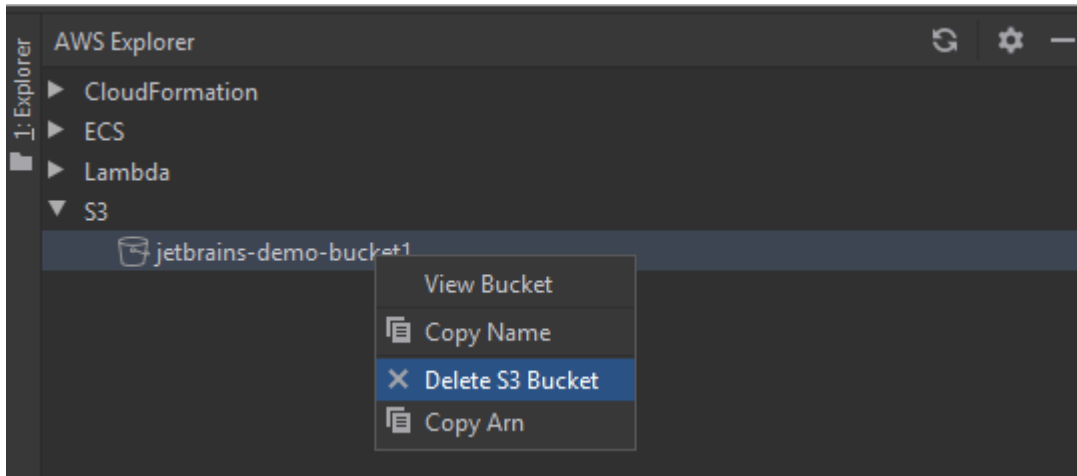
4. Choose **Create**.

Viewing Amazon S3 buckets

1. Open AWS Explorer, if it isn't already open.
2. Click the **Amazon S3** node to expand the list of buckets.
 - The S3 buckets for the [current AWS Region](#) are displayed beneath the **Amazon S3** node.

Deleting an Amazon S3 bucket

1. Open AWS Explorer, if it isn't already.
2. Click the **Amazon S3** node to expand the list of buckets.
3. Right-click the bucket to delete, and then choose **Delete S3 Bucket**.



4. Enter the bucket's name to confirm the deletion, and then choose **OK**.
 - If the bucket contains objects, the bucket is emptied before deletion. A notification is displayed after the deletion is complete.

Working with Amazon S3 objects by using the AWS Toolkit for JetBrains

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata.

Topics

- [Viewing an object in an Amazon S3 bucket](#)
- [Opening an object in the IDE](#)
- [Uploading an object](#)
- [Downloading an object](#)
- [Deleting an object](#)

Viewing an object in an Amazon S3 bucket

This procedure opens the **S3 Bucket Viewer**. You can use it to view, upload, download, and delete objects grouped by folders in an Amazon S3 bucket.

1. Open AWS Explorer, if it isn't already open.
2. To view a bucket's objects, do one of the following:
 - Double-click the name of the bucket.
 - Right-click the name of the bucket, and then choose **View Bucket**.

The **S3 Bucket Viewer** displays information about the bucket's name, [Amazon Resource Name \(ARN\)](#), and creation date. The objects and folders in the bucket are available in the pane beneath.

Opening an object in the IDE

If the object in an Amazon S3 bucket is a file type recognized by the IDE, you can download a read-only copy and open it in the IDE.

1. To find an object to download, open the **S3 Bucket Viewer** (see [Viewing an object in an Amazon S3 bucket](#)).
2. Double-click the name of the object.

The file opens in the default IDE window for that file type.

Uploading an object

1. To find the folder you want to upload objects to, open the **S3 Bucket Viewer** (see [Viewing an object in an Amazon S3 bucket](#)).
2. Right-click the folder, and then choose **Upload**.
3. In the dialog box, select the files to upload.

Note

You can upload multiple files at once. You can't upload directories.

4. Choose **OK**.

Downloading an object

1. To find a folder to download objects from, open the **S3 Bucket Viewer** (see [Viewing an object in an Amazon S3 bucket](#)).
2. Choose a folder to display its objects.
3. Right-click an object, and then choose **Download**.
4. In the dialog box, select the download location.

Note

If you're downloading multiple files, ensure you select the path name instead of the folder. You can't download directories.

5. Choose **OK**.

Note

If a file already exists in the download location, you can overwrite it or leave it in place by skipping the download.

Deleting an object

1. To find the object to delete, open the **S3 Bucket Viewer** (see [Viewing an object in an Amazon S3 bucket](#)).
2. After you select the object, delete it by doing one of the following:
 - Press **Delete**.
 - Right-click, and then choose **Delete**.

Note

You can select and delete multiple objects at once.

3. To confirm the deletion, choose **Delete**.

Working with AWS serverless applications by using the AWS Toolkit for JetBrains

The following topics describe how to use the AWS Toolkit for JetBrains to work with AWS serverless applications in an AWS account.

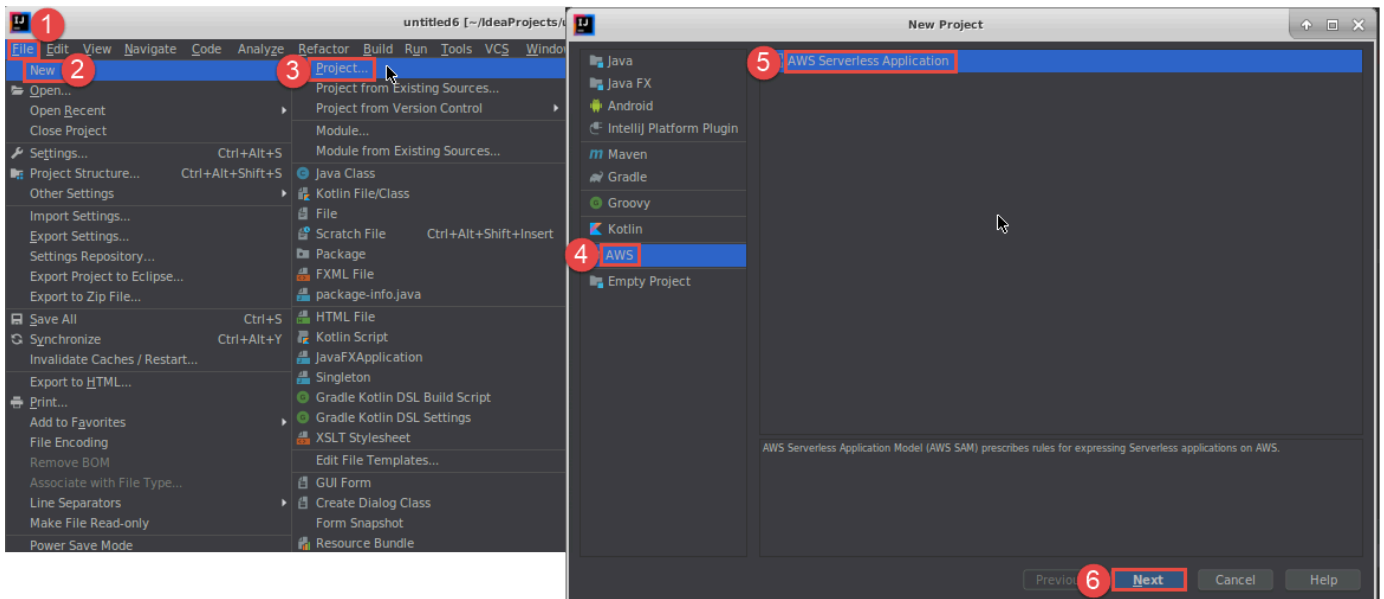
Topics

- [Creating an AWS serverless application by using the AWS Toolkit for JetBrains](#)
- [Syncing AWS SAM applications from the AWS Toolkit for JetBrains](#)
- [Changing \(updating\) AWS Serverless application settings by using the AWS Toolkit for JetBrains](#)
- [Deleting an AWS serverless application by using the AWS Toolkit for JetBrains](#)

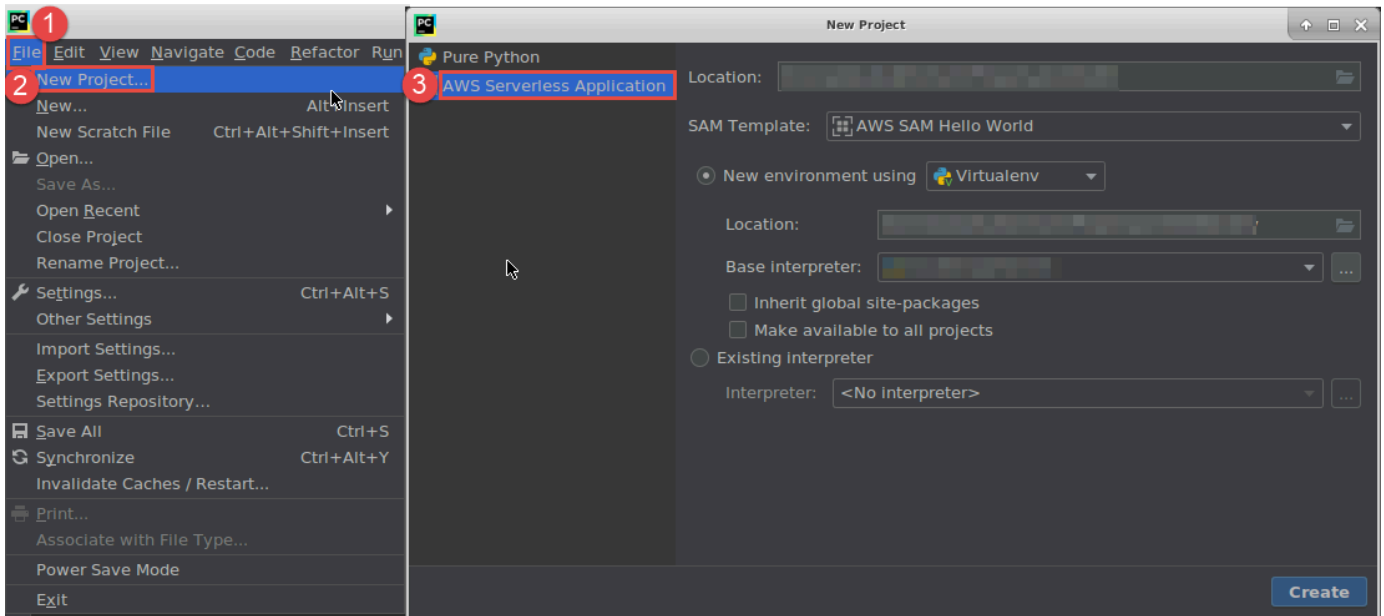
Creating an AWS serverless application by using the AWS Toolkit for JetBrains

To complete this procedure, you must first install the AWS Toolkit and, if you haven't yet, connect to an AWS account for the first time. Then with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following."?>

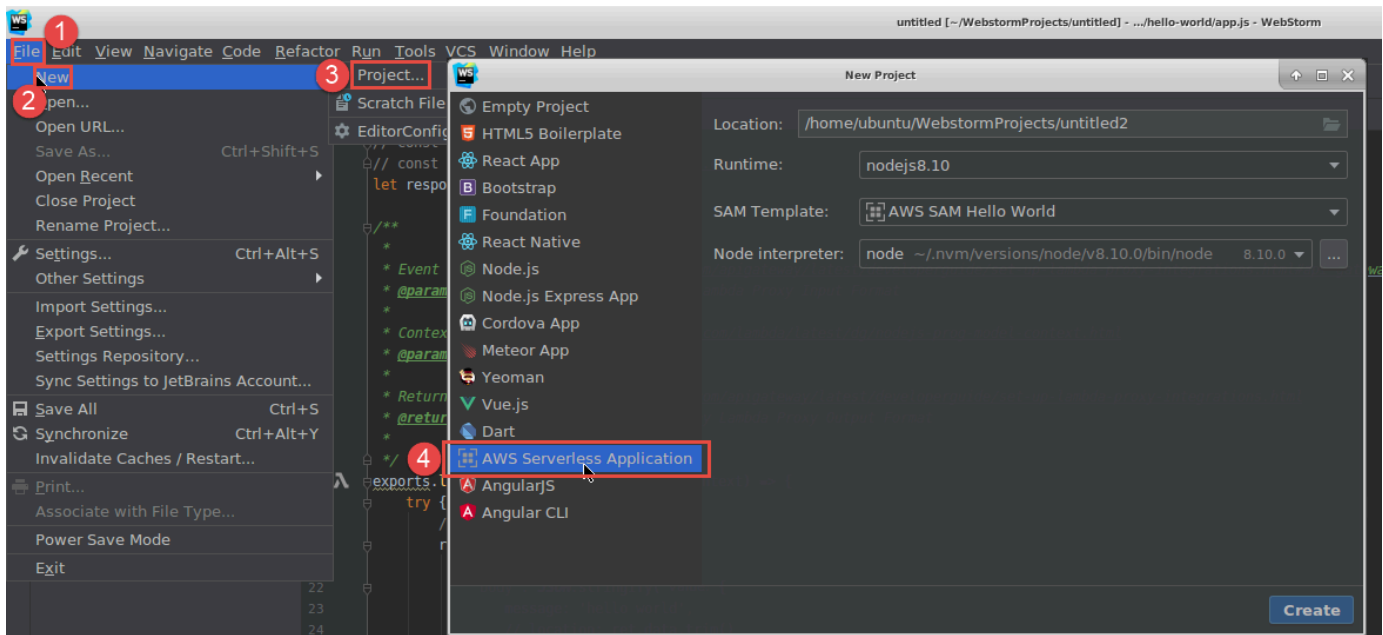
1. With IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:
 - For IntelliJ IDEA or WebStorm, choose **File, New, Project**.
 - For PyCharm, choose **File, New Project**.
 - For JetBrains Rider, choose **File, New** for a new solution. Or right-click an existing solution in the **Explorer** tool window, and then choose **Add, New Project**.
2. For IntelliJ IDEA, choose **AWS, AWS Serverless Application**, and then choose **Next**.



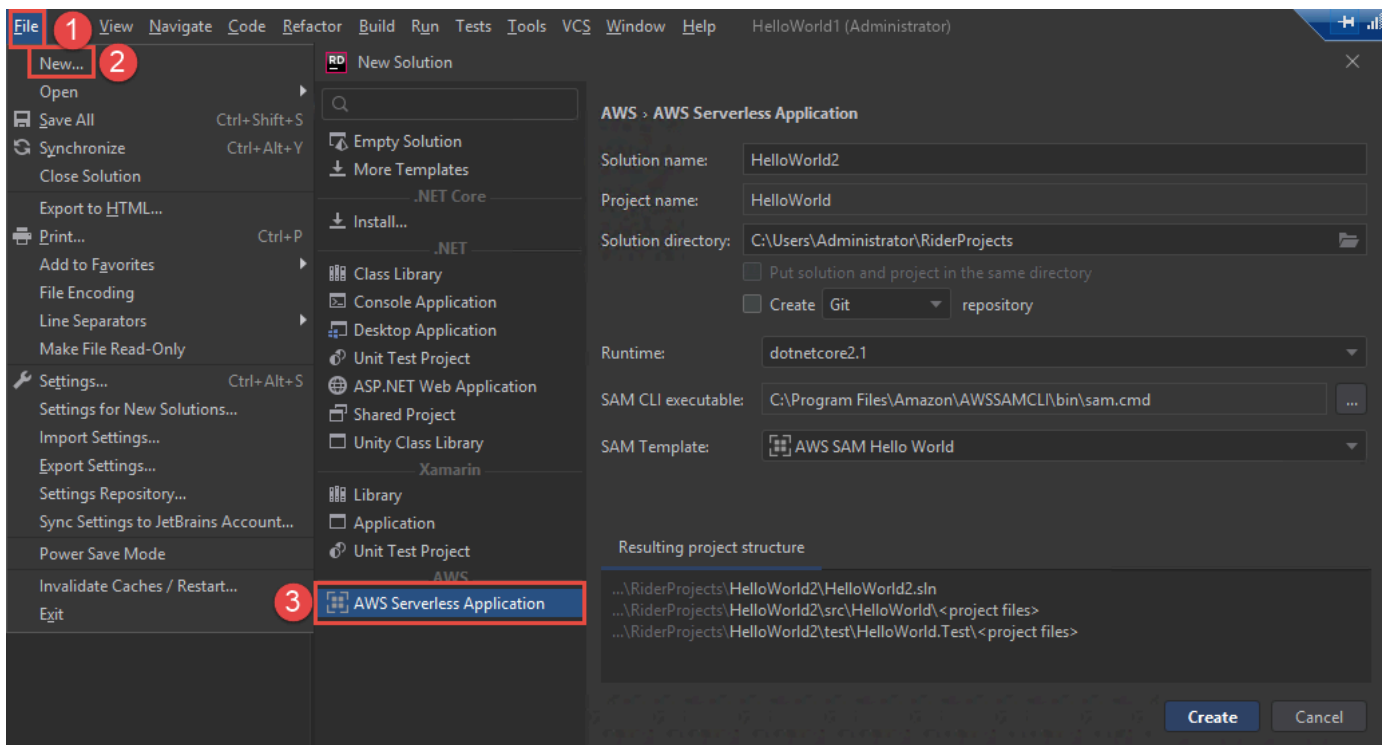
For PyCharm, choose **AWS Serverless Application**.



For WebStorm, choose **AWS Serverless Application**.

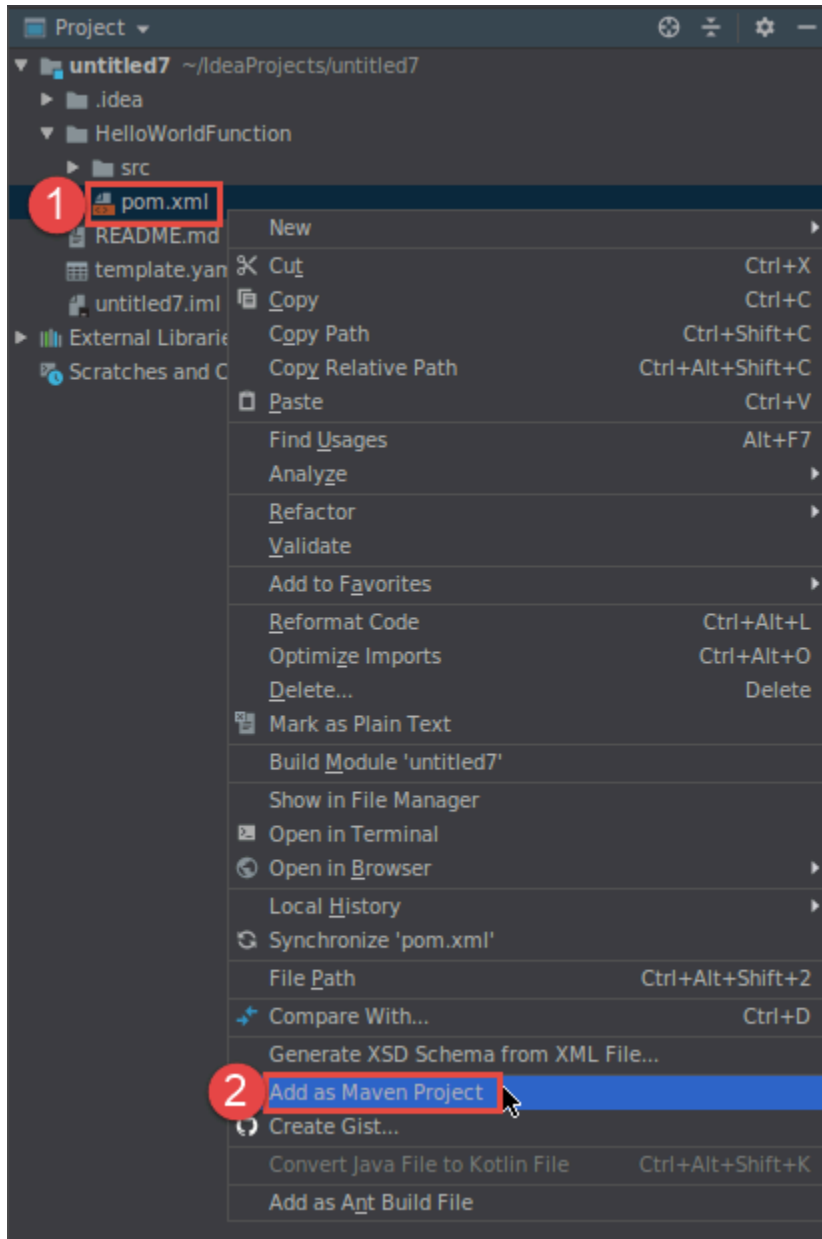


For JetBrains Rider, choose **AWS Serverless Application**.

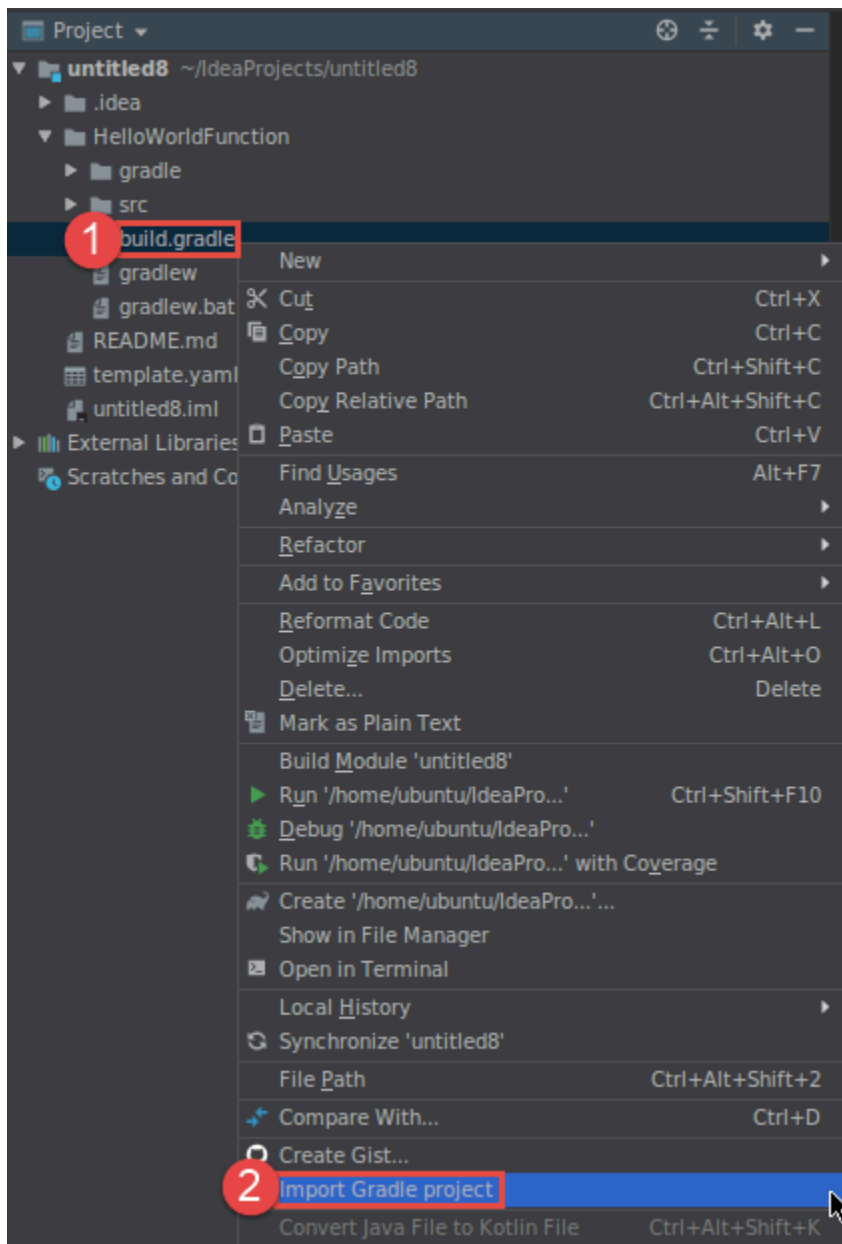


- Complete the [New Project dialog box](#) (or the [New Solution dialog box](#) for JetBrains Rider), and then choose **Finish** (for IntelliJ IDEA) or **Create** (for PyCharm, WebStorm, or JetBrains Rider). The AWS Toolkit for JetBrains creates the project and adds the serverless application's code files to the new project.

4. If you're using IntelliJ IDEA, with the **Project** tool window already open and displaying the project that contains the serverless application's files, do one of the following:
 - For Maven-based projects, right-click the project's `pom.xml` file, and then choose **Add as Maven Project**.



- For Gradle-based projects, right-click the project's `build.gradle` file, and then choose **Import Gradle project**.



Complete the **Import Module from Gradle** dialog box, and then choose **OK**.

After you create the serverless application, you can run (invoke) or debug the local version of an AWS Lambda function that is contained in that application.

You can also deploy the serverless application. After you deploy it, you can (invoke) the remote version of a Lambda function that is part of that deployed application.

Syncing AWS SAM applications from the AWS Toolkit for JetBrains

AWS Serverless Application Model (AWS SAM) `sam sync` is an AWS SAM-CLI-command deployment process that automatically identifies changes made to your serverless applications, then chooses the best way to build and deploy those changes to the AWS Cloud. If you've only made changes to your application code without changing the infrastructure, AWS SAM Sync updates your application without redeploying your AWS CloudFormation stack.

For additional information about `sam sync` and AWS SAM CLI commands, see the [AWS SAM CLI command reference](#) topic in the *AWS Serverless Application Model User Guide*.

The following sections describe how to get started working with AWS SAM Sync.

Prerequisites

Prior to working with AWS SAM Sync, the following prerequisites must be met:

- You have a working AWS SAM application. For more information on creating a AWS SAM application, see the [Working with AWS SAM](#) topic in this User Guide.
- You've installed version 1.78.0. (or later) of the AWS SAM CLI. For more information on installing the AWS SAM CLI, see the [Installing the AWS SAM CLI](#) topic in the *AWS Serverless Application Model User Guide*.
- Your application is running in a development environment.

Note

To sync and deploy a serverless application that contains an AWS Lambda function with any non-default properties, the optional properties must be set in the AWS SAM template file associated with the AWS Lambda function, prior to deployment.

To learn more about AWS Lambda properties, see the [AWS::Serverless::Function](#) section in the *AWS Serverless Application Model User Guide* on GitHub.

Getting Started

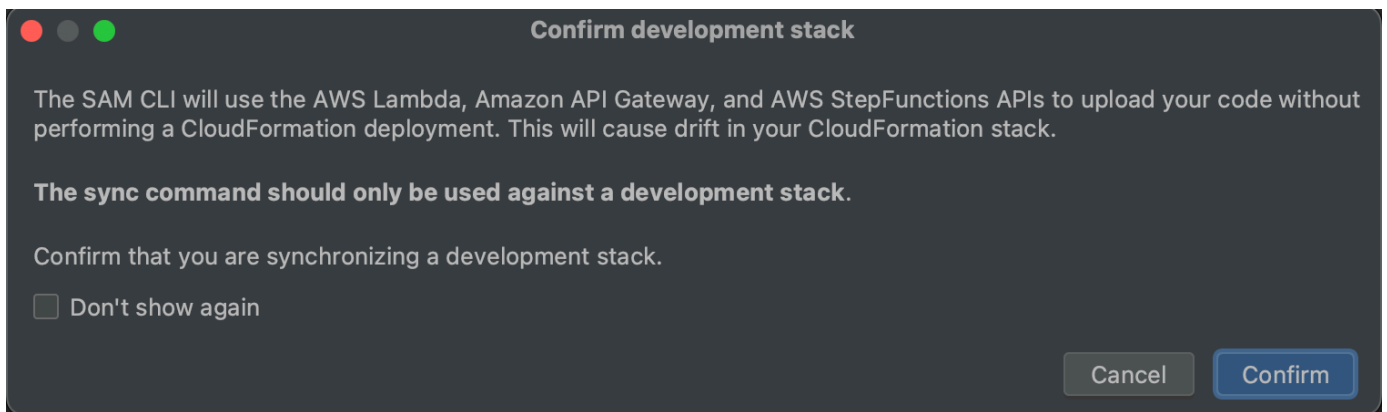
To get started working with AWS SAM Sync, complete the following procedure.

Note

Make sure that your AWS Region is set to the location associated with your serverless application.

To learn more about changing your AWS region from the AWS Toolkit for JetBrains, see the [Switch between AWS Regions](#) topic in this User Guide.

1. From your serverless application project in the **Project** tool window, open the context menu for (right-click) your `template.yaml` file.
2. From the `template.yaml` context menu, choose **Sync Serverless Application (formerly Deploy)** to open the **Confirm development stack** dialog.
3. Confirm that you are working from a development stack to open the **Sync Serverless Application** dialog.



4. Complete the steps in the **Sync Serverless Application** dialog, then choose **Sync** to begin the AWS SAM Sync process. To learn more about the **Sync Serverless Application** dialog, see the [the section called " Sync Serverless Application Dialog "](#) section located below.
5. During the sync process, the AWS Toolkit for JetBrains **Run Window** is updated with the deployment status.
6. Following a successful sync, the name of your AWS CloudFormation stack is added to the **AWS Explorer**.

If the sync fails, troubleshooting details can be found in the JetBrains **Run Window** or the AWS CloudFormation **event logs**. To learn more about viewing AWS CloudFormation event logs, see the [Viewing event logs for a stack](#) topic in this User Guide.

Sync Serverless Application Dialog

The **Sync serverless application dialog** assists you with the AWS SAM sync process. The following sections are descriptions and details for each of the different dialog components.

Create Stack or Update Stack

Required: To create a new deployment stack, enter a name in the provided field to create and set the AWS CloudFormation stack for your serverless application deployment.

Alternatively, to deploy to an existing AWS CloudFormation stack, select the stack name from the auto-populated list of stacks associated with your AWS account.

Template Parameters

Optional: Populates with a list of parameters detected from your project `template.yaml` file. To specify parameter values, enter a new parameter value into the provided text-field located in the **value** column.

S3 Bucket

Required: To choose an existing Amazon Simple Storage Service (Amazon S3) bucket for storing your AWS CloudFormation template, select it from the list.

To create and use new Amazon S3 bucket for storage, choose **Create** and follow the prompts.

ECR Repository

Required, only visible when working with an Image package type: Choose an existing Amazon Elastic Container Registry (Amazon ECR) repository URI for deployment of your serverless application.

For information about AWS Lambda package types, see the [Lambda deployment packages](#) section in the *AWS Lambda Developer Guide*.

CloudFormation Capabilities

Required: Choose the capabilities that AWS CloudFormation is allowed to use when creating stacks.

Tags

Optional: Enter your preferred tags in the provided text fields to tag a parameter.

Build Function Inside a Container

Optional, Docker required: Selecting this options builds your serverless-application functions inside of a local Docker container, prior to deployment. This option is useful if a function depends on packages with natively compiled dependencies or programs.

For more information, see the [Building applications](#) topic in the *AWS Serverless Application Model Developer Guide*.

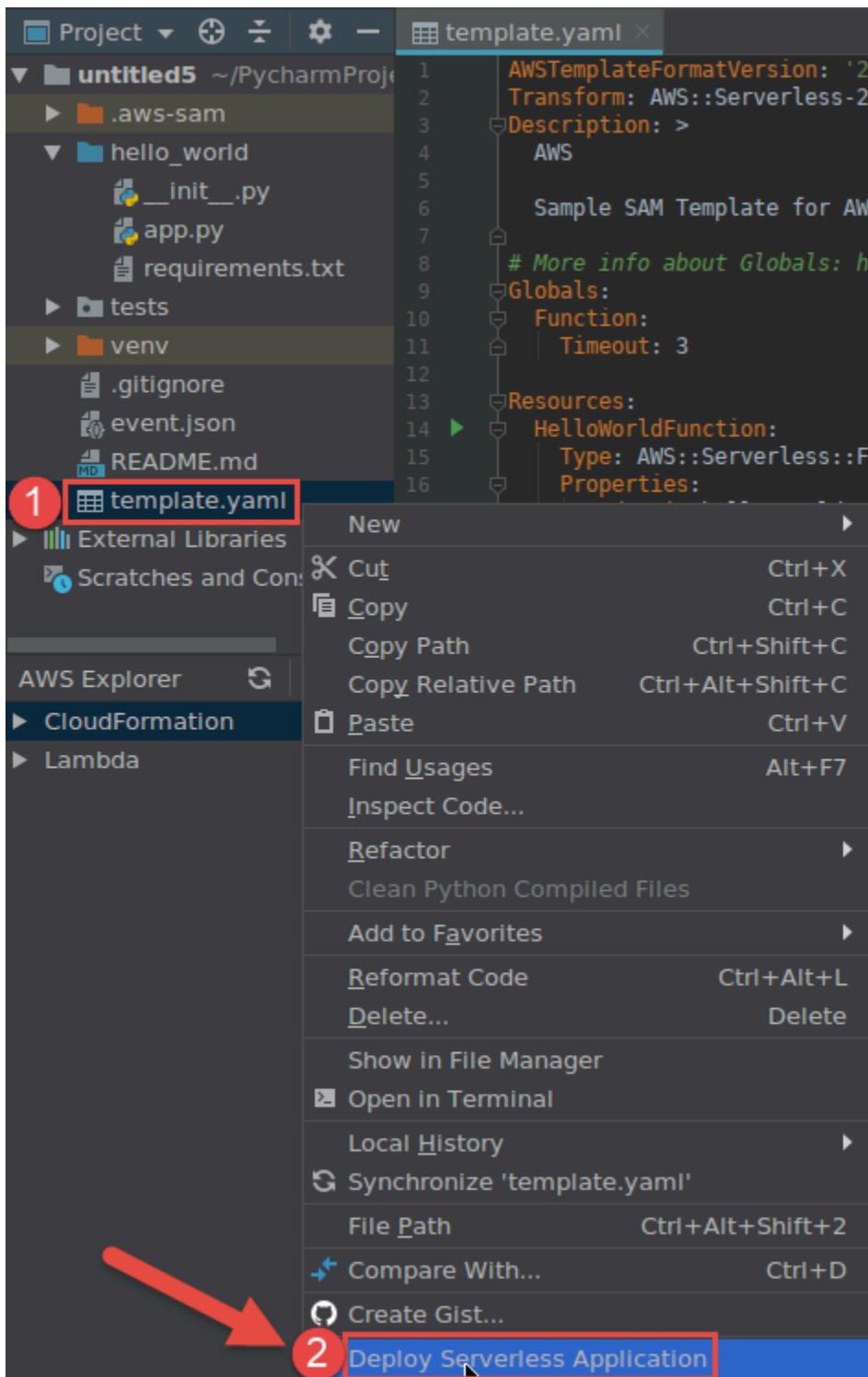
Changing (updating) AWS Serverless application settings by using the AWS Toolkit for JetBrains

You must first deploy the AWS serverless application that you want to change, if you haven't deployed it already.

Note

To deploy a serverless application that contains an AWS Lambda function, and deploy that function with any nondefault or optional properties, you must first set those properties in the function's corresponding AWS SAM template file (for example, in a file named `template.yaml` within the project). For a list of available properties, see [AWS::Serverless::Function](#) in the [awslabs/serverless-application-model](#) repository on GitHub.

1. With the **Project** tool window already open and displaying the project that contains the serverless application's files, open the project's `template.yaml` file. Change the file's contents to reflect the new settings, and then save and close the file.
2. If you need to switch to a different AWS Region to deploy the serverless application to, do that now.
3. Right-click the project's `template.yaml` file, and then choose **Deploy Serverless Application**.



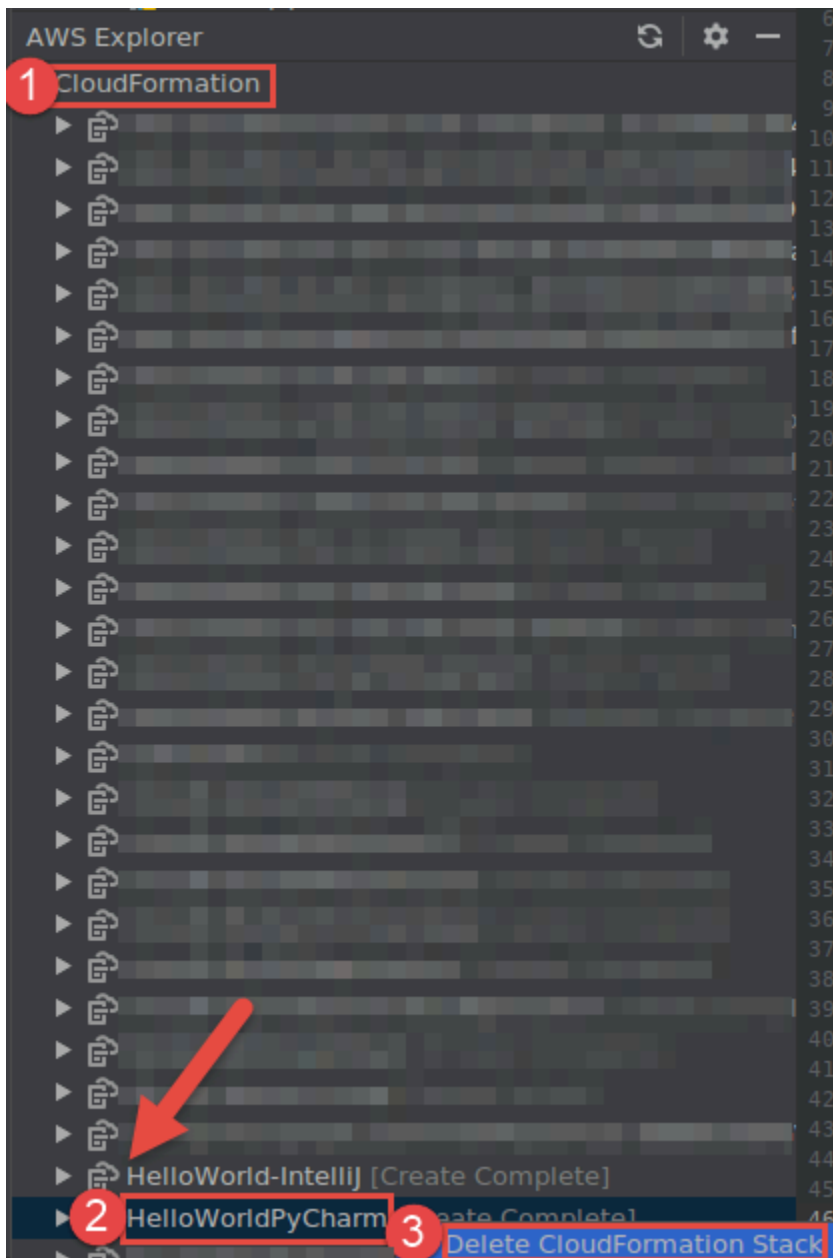
4. Complete the [Deploy Serverless Application](#) dialog box, and then choose **Deploy**. The AWS Toolkit for JetBrains updates the corresponding AWS CloudFormation stack for the deployment.

If the deployment fails, you can try to determine why by viewing event logs for the stack.

Deleting an AWS serverless application by using the AWS Toolkit for JetBrains

Before deleting an AWS serverless application, you must first deploy it.

1. Open AWS Explorer, if it isn't already open. If you need to switch to a different AWS Region that contains the serverless application, do that now.
2. Expand **CloudFormation**.
3. Right-click the name of the AWS CloudFormation stack that contains the serverless application you want to delete, and then choose **Delete CloudFormation Stack**.



4. Enter the stack's name to confirm the deletion, and then choose **OK**. If the stack deletion succeeds, the AWS Toolkit for JetBrains removes the stack name from the **CloudFormation** list in **AWS Explorer**. If the stack deletion fails, you can try to determine why by viewing event logs for the stack.

Working with Amazon Simple Queue Service from the AWS Toolkit for JetBrains

The following topics describe how work with the Amazon Simple Queue Service service from the AWS Toolkit for JetBrains.

Topics

- [Working with Amazon Simple Queue Service queues](#)
- [Using Amazon SQS with AWS Lambda in the AWS Toolkit for JetBrains](#)
- [Using Amazon SQS with Amazon SNS in the AWS Toolkit for JetBrains](#)

Working with Amazon Simple Queue Service queues

The following topics describe how to use the AWS Toolkit for JetBrains to work with Amazon Simple Queue Service queues and messages.

Standard and FIFO (First-In-Last-Out) are the two kinds of messages you can send using Amazon SQS in the AWS Toolkit for JetBrains.

To create an Amazon SQS queue

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, open the context menu for (right-click) the **Amazon SQS** service, and choose **Create Queue...**
3. Provide a queue name and choose the queue type.

Note

For more information on queue types, see the [Amazon SQS standard queues](#) and [Amazon SQS FIFO \(First-In-First-Out\) queues](#) topics in the *Amazon Simple Queue Service Developer Guide*.

4. Choose **Create**.

To view Amazon SQS messages

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue you want to view and choose **View Messages**.
4. Choose **View Messages** to view the messages in this queue.

To edit Amazon SQS queue properties

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue that you want to edit and choose **Edit Queue Properties...**
4. In the **Edit Queue Properties** dialog box that opens, review and modify your queue properties. For more information on Amazon SQS properties, see [Configuring queue parameters \(console\)](#) in the *Amazon Simple Queue Service Developer Guide*.

To send Standard messages

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue for sending your message and choose **Send a message**.
4. Populate the message and choose **Send**. After you send the message, you see a confirmation that includes the message ID.

To send FIFO messages

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue for sending your message and choose **Send a message**.
4. Populate the message, group id, and an optional deduplication id.

Note

If no deduplication id is provided, one will be generated.

5. Choose **Send**. After you send the message, you see a confirmation that includes the message ID.

To delete an Amazon SQS queue

1. Verify that a queue is empty before you delete it. For more information see [Confirming that a queue is empty](#) in the *Amazon Simple Queue Service Developer Guide*.
2. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
3. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
4. Right-click **Amazon SQS**, and choose **Delete Queue...**
5. Confirm that you want to delete the queue, and choose **OK** in the deletion dialog box.

Using Amazon SQS with AWS Lambda in the AWS Toolkit for JetBrains

The following procedure details how to configure Amazon SQS queues as Lambda triggers in the AWS Toolkit for JetBrains.

To configure an Amazon SQS queue as a Lambda triggers

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue you want to work with and choose **Configure Lambda Trigger**.
4. In the dialog box, from the drop-down menu, choose the Lambda function that you want to trigger.
5. Choose **Configure**.
6. If the Lambda function lacks the necessary IAM permissions for Amazon SQS to run it, the toolkit generates a minimal policy that you can add to the IAM role for the Lambda function.

Choose **Add Policy**.

After you configure your queue, you get a status message about the applied changes, including any applicable error messages.

Using Amazon SQS with Amazon SNS in the AWS Toolkit for JetBrains

The following procedure details how to subscribe Standard Amazon SQS queues to Amazon SNS topics using the AWS Toolkit for JetBrains.

Note

You can't subscribe FIFO Amazon SQS queues to Amazon SNS topics.

To subscribe a Standard Amazon SQS queue to an Amazon SNS topic

1. From the AWS Toolkit for JetBrains, expand the AWS Explorer to view your AWS services.
2. From the AWS Explorer, expand the **Amazon SQS** service to view a list of your existing queues.
3. Right-click the queue you want to work with and choose **Subscribe to SNS topic....**
4. In the dialog box, from the drop-down menu, choose an Amazon SNS topic, and then choose **Subscribe**.

Working with resources

In addition to accessing AWS services that are listed by default in the AWS Explorer, you can also go to **Resources** and choose from hundreds of resources to add to the interface. In AWS, a **resource** is an entity you can work with. Some of the resources that can be added include Amazon AppFlow, Amazon Kinesis Data Streams, AWS IAM roles, Amazon VPC, and Amazon CloudFront distributions.

After making your selection, you can go to **Resources** and expand the resource type to list the available resources for that type. For example, if you select the `AWS::Lambda::Function` resource type, you can access the resources that define different functions, their properties, and their attributes.

After adding a resource type to **Resources**, you can interact with it and its resources in the following ways:

- View a list of existing resources that are available in the current AWS Region for this resource type.

- View a read-only version of the JSON file that describes a resource.
- Copy the resource identifier for the resource.
- View the AWS documentation that explains the purpose of the resource type and the schema (in JSON and YAML formats) for modelling a resource.
- Create a new resource by editing and saving a JSON-formatted template that conforms to a schema.*
- Update or delete an existing resource.*

Important

*In the current release of the AWS Toolkit for JetBrains the option to create, edit, and delete resources is an *experimental feature*. Because experimental features continue to be tested and updated, they may have usability issues. And experimental features may be removed from the AWS Toolkit for JetBrains without notice.

To allow the use of experimental features for resources, open the **Settings** pane in your JetBrains IDE, and expand **Tools**, and then choose **AWS, Experimental Features**. Select **JSON Resource Modification** to allow you to create, update, and delete resources.

For more information, see [Working with experimental features](#).

IAM permissions for accessing resources

You require specific AWS Identity and Access Management permissions to access the resources associated with AWS services. For example, an IAM entity, such as a user or a role, requires Lambda permissions to access `AWS::Lambda::Function` resources.

In addition to permissions for service resources, an IAM entity requires permissions to permit the AWS Toolkit for JetBrains to call AWS Cloud Control API operations on its behalf. Cloud Control API operations allow the IAM user or role to access and update the remote resources.

The easiest way to grant permissions is to attach the AWS managed policy, **PowerUserAccess**, to the IAM entity that's calling these API operations using the Toolkit interface. This [managed policy](#) grants a range of permissions for performing application development tasks, including calling API operations.

For specific permissions that define allowable API operations on remote resources, see the [AWS Cloud Control API User Guide](#).

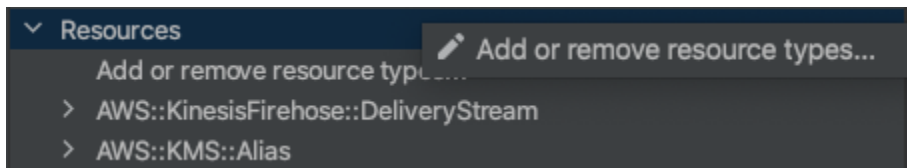
Adding and interacting with existing resources

1. In the **AWS Explorer**, right-click **Resources** and choose **Add or remove resources**.

Additional Explorer Resources in the **Settings** pane displays a list of resource types that are available for selection.

Note

You can also display the list of resource types by double-clicking the **Add or remove resources** node, which is under **Resources**.



2. In the **Additional Explorer Resources**, select the resource types to add to the **AWS Explorer** and press **Return** or choose **OK** to confirm.

The resource types that you selected are listed under **Resources**.

Note

If you've already added a resource type to the **AWS Explorer** and then clear the checkbox for that type, it's no longer listed under **Resources** after you choose **OK**. Only those resource types that are currently selected are visible in the **AWS Explorer**.

3. To view the resources that already exist for a resource type, expand the entry for that type.

A list of available resources is displayed under their resource type.

4. To interact with a specific resource, right-click its name and choose one of the following options:
 - **View resource**: View a read-only version of the JSON-formatted template that describes the resource.

After the template is displayed, you can change it by choosing **Edit** if you have the required [???](#) enabled.

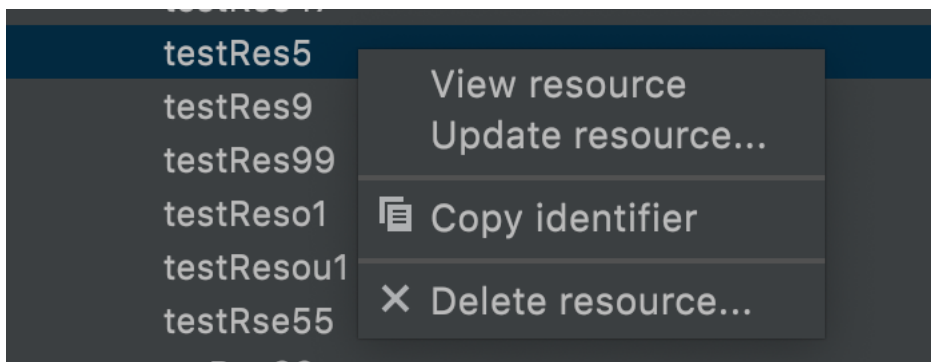
Note

You can also view the resource by double-clicking it.

- **Copy identifier:** Copy the identifier for the specific resource to the clipboard. (For example, the `AWS::DynamoDB::Table` resource can be identified using the `TableName` property.)
- **Update resource:** Edit the JSON-formatted template for the resource in a JetBrains editor. For more information, see [Creating and updating resources](#).
- **Delete resource:** Delete the resource by confirming the deletion in a dialog box that is displayed. (Deleting resources is currently an [???](#) in this version of the AWS Toolkit for JetBrains.)

Warning

If you delete a resource, any AWS CloudFormation stack that uses that resource will fail to update. To fix this update failure, you need to either recreate the resource or remove the reference to it in the stack's AWS CloudFormation template. For more information, see this [Knowledge Center article](#).



Creating and updating resources

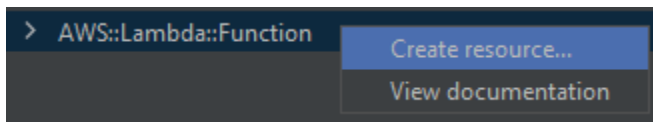
Important

The creation and updating of resources is currently an [???](#) in this version of the AWS Toolkit for JetBrains.

Creating a new resource involves adding a resource type to the **Resources** list and then editing a JSON-formatted template that defines the resource, its properties, and its attributes.

For example, a resource that belongs to the `AWS::SageMaker::UserProfile` resource type is defined with a template that creates a user profile for Amazon SageMaker Studio. The template that defines this user profile resource must conform to the resource type schema for `AWS::SageMaker::UserProfile`. If the template doesn't comply with the schema because of missing or incorrect properties, for example, the resource can't be created or updated.

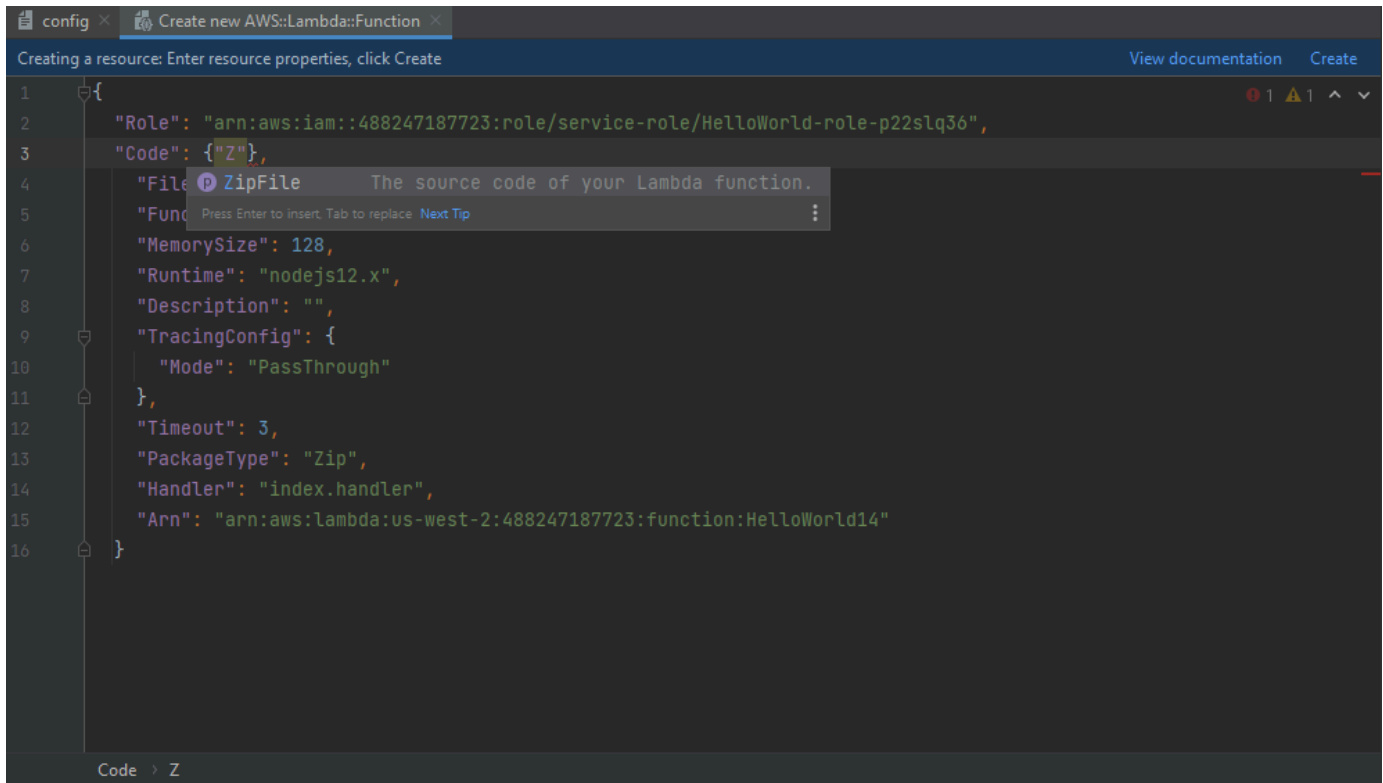
1. Add the resource type for the resource you want to create by right-clicking **Resources** and choosing **Add or remove resources**.
2. After the resource type is added under **Resources**, right-click its name and choose **Create resource**. You can also access information about how to model the resource by choosing **View documentation**.



3. In the editor, start to define properties that make up the resource template. The autocomplete feature suggests property names that conform with your template's schema. When your template fully conforms with JSON syntax, the error count is replaced by a green checkmark. For detailed information about the schema, choose **View documentation**.

Note

As well as conforming to basic JSON syntax, your template must conform to the schema that models the resource type. Your template is validated against the schema model when you try to create or update the remote resource.



```
1 {
2   "Role": "arn:aws:iam::488247187723:role/service-role/HelloWorld-role-p22slq36",
3   "Code": {"Z"},
4   "File ZipFile The source code of your Lambda function.
5   "Func Press Enter to insert. Tab to replace Next Tip
6   "MemorySize": 128,
7   "Runtime": "nodejs12.x",
8   "Description": "",
9   "TracingConfig": {
10    "Mode": "PassThrough"
11  },
12  "Timeout": 3,
13  "PackageType": "Zip",
14  "Handler": "index.handler",
15  "Arn": "arn:aws:lambda:us-west-2:488247187723:function:HelloWorld14"
16 }
```

4. After you finish declaring your resource, choose **Create** to validate your template and save the resource to the remote AWS Cloud. (Choose **Update** if you're modifying an existing resource.)

If your template defines the resource in accordance with its schema, a message displays to confirm that the resource was created. (If the resource already exists, the message confirms that the resource was updated.)

After the resource is created, it's added to the list under the resource type heading.

5. If your file contains errors, a message displays to explain that the resource couldn't be created or updated. Open the **Event Log** to identify the template elements that you need to fix.

User interface reference for the AWS Toolkit for JetBrains

For help working with the AWS Toolkit for JetBrains user interface, see the following topics.

Topics

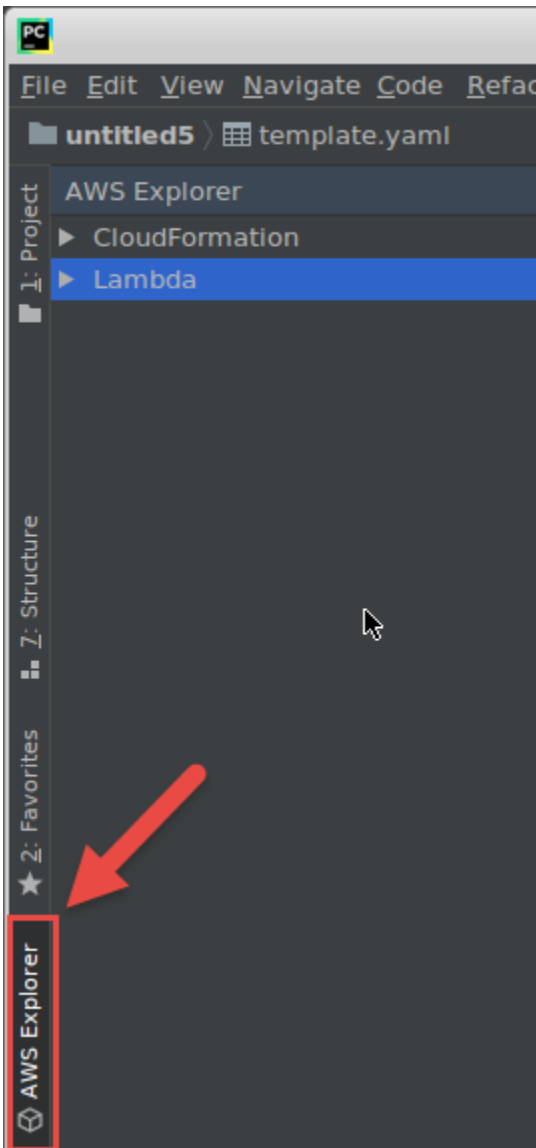
- [AWS Explorer](#)
- [Create Function dialog box](#)
- [Deploy Serverless Application dialog box](#)
- [New Project dialog box](#)
- [Run/Debug Configurations dialog box](#)
- [Update Code dialog box](#)
- [Update Configuration dialog box](#)

AWS Explorer

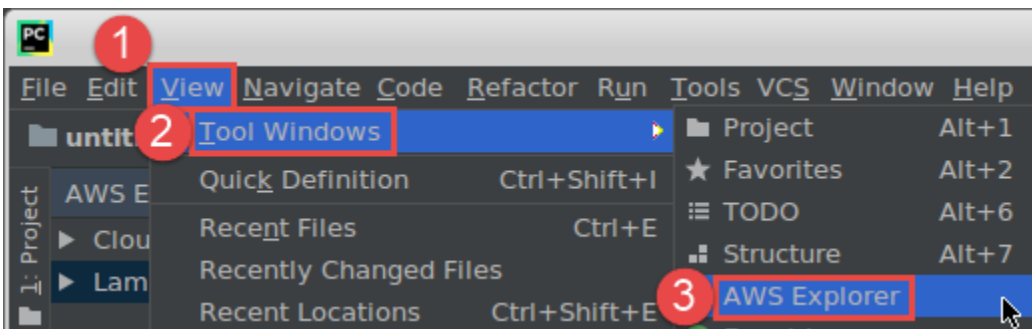
AWS Explorer provides convenient access to several features in the AWS Toolkit for JetBrains. These include managing connections from the toolkit to AWS accounts, switching AWS Regions, working with AWS Lambda functions and AWS CloudFormation stacks in accounts, and more.

To open AWS Explorer, with the AWS Toolkit for JetBrains installed and with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider running, do one of the following:

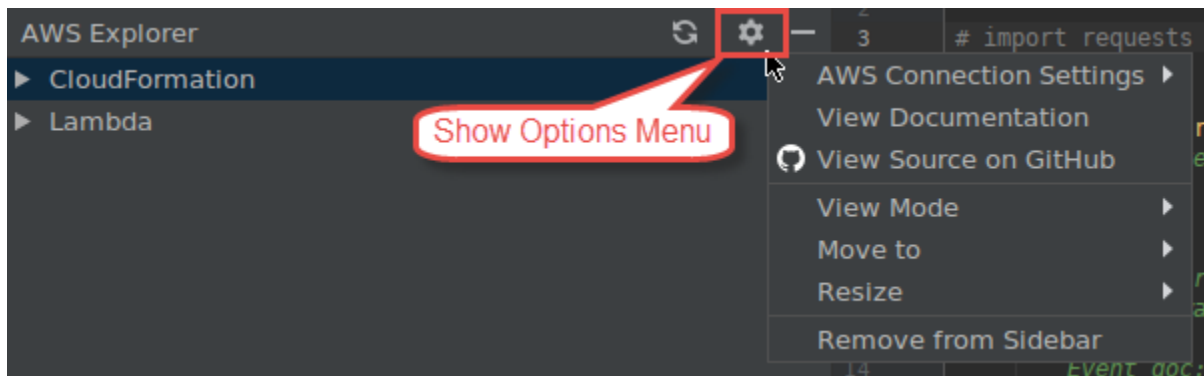
- On the tool window bar, choose **AWS Explorer**.



- On the main menu, choose **View, Tool Windows, AWS Explorer**.



In **AWS Explorer**, choose the settings icon (**Show Options Menu**) for the following options:



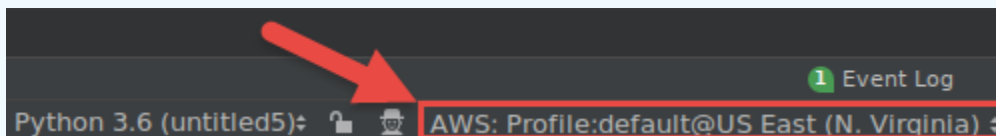
AWS Connection Settings

Contains the following options:

- **AWS Regions list** – The AWS Toolkit for JetBrains uses the selected Region. To have the toolkit use a different Region, choose another listed Region.
- **Recent Credentials list** – Lists recent connections made from the AWS Toolkit for JetBrains to AWS accounts. The toolkit uses the selected connection. To have the toolkit use a different recent connection, choose that connection's name.
- **All Credentials** – Lists all available connections that you can make from the AWS Toolkit for JetBrains to AWS accounts. The toolkit uses the selected connection. To have the toolkit use a different connection, choose that connection's name. To do other connection tasks, choose **AWS Edit Credential file(s)**.

Note

The **AWS Connection Settings** area in the status bar displays the AWS account connection and Region that the AWS Toolkit for JetBrains is currently using.



Choose this area to view the same **AWS Connection Settings** options as the **Show Options Menu**.

View Documentation

Goes to the [AWS Toolkit for JetBrains User Guide](#) (this guide).

View Source on GitHub

Goes to the [aws/aws-toolkit-jetbrains](https://github.com/aws/aws-toolkit-jetbrains) repository on the GitHub website.

View Mode

Adjusts the **AWS Explorer** tool window so that you can quickly access it and save space when you work in the editor or other tool windows.

For IntelliJ IDEA view modes, see [Tool window view modes](#) on the IntelliJ IDEA Help website.

For PyCharm view modes, see [Tool window view modes](#) on the PyCharm Help website.

For WebStorm view modes, see [Tool window view modes](#) on the WebStorm Help website.

For JetBrains Rider view modes, see [Tool window view modes](#) on the JetBrains Rider Help website.

Move to

Moves the **AWS Explorer** tool window to a different location in IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider.

Resize

Changes the size of the **AWS Explorer** tool window.

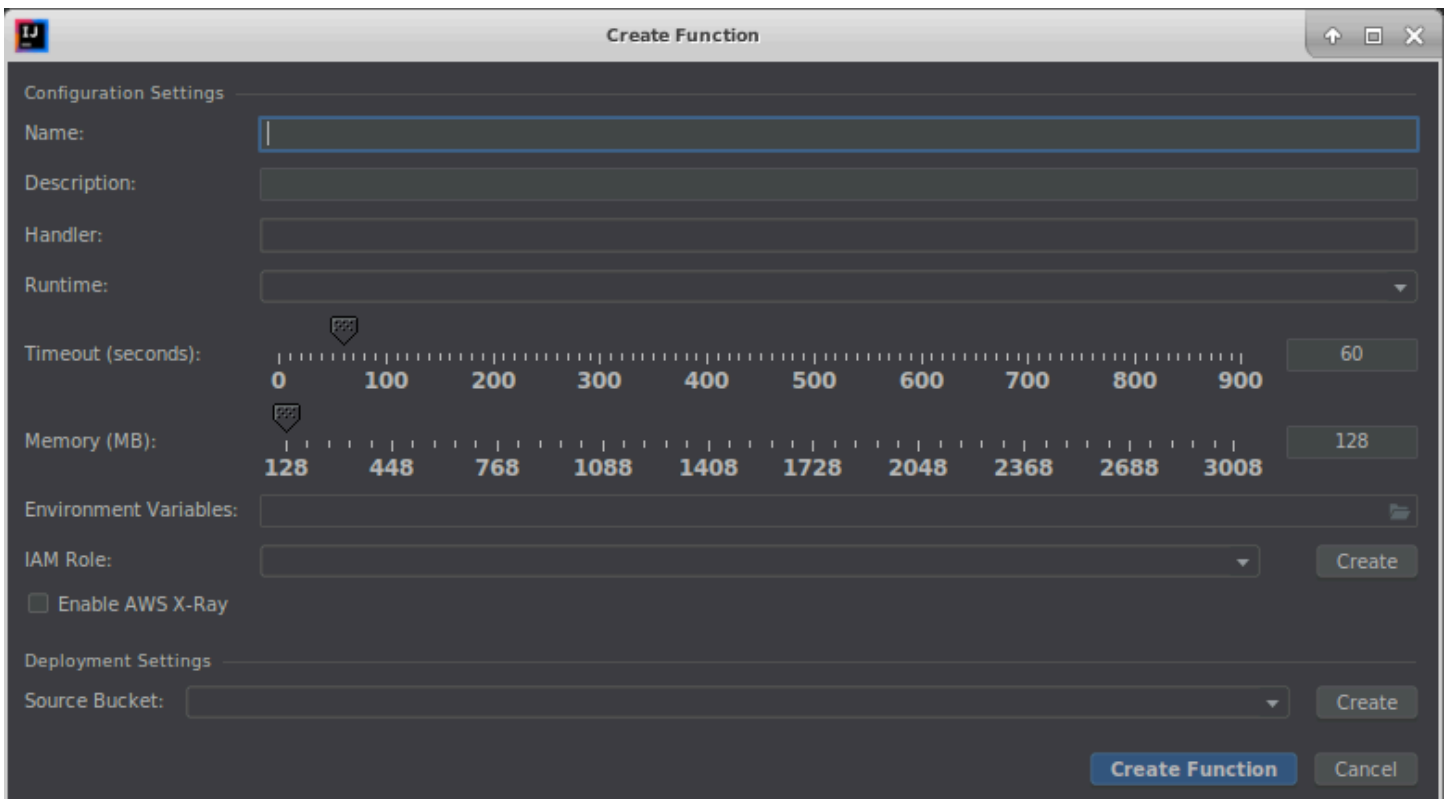
Remove from Sidebar

Removes the **AWS Explorer** tool window from the tool window bar. To display it again, on the main menu bar, choose **View, Tool Windows, AWS Explorer**.

You can also use AWS Explorer to work with Lambda functions and work with AWS CloudFormation stacks in AWS accounts.

Create Function dialog box

The **Create Function** dialog box in the AWS Toolkit for JetBrains is displayed when you create a standalone AWS Lambda function.



The **Create Function** dialog box contains the following items:

Name

(Required) The function's name. This can contain only the uppercase letters A through Z, the lowercase letters a through z, the numbers 0 through 9, hyphens (-), and underscores (_). The name must be less than 64 characters in length.

Description

(Optional) Any meaningful description about the function.

Handler

(Required) The ID of the corresponding function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#).

Runtime

(Required) The ID of the [Lambda runtime](#) to use.

Timeout (seconds)

(Required) The amount of time that Lambda allows a function to run before stopping it. Specify an amount up to 900 seconds (15 minutes).

Memory (MB)

(Required) The amount of memory available to the function as it runs. Specify an amount [between 128 MB and 3,008 MB](#) in 64-MB increments.

Environment Variables

(Optional) Any [environment variables](#) for the Lambda function to use, specified as key-value pairs. To add, change, or delete environment variables, choose the folder icon, and then follow the on-screen instructions.

IAM Role

(Required) Choose an available [Lambda execution role](#) in the connected AWS account for Lambda to use for the function. To create an execution role in the account and have Lambda use that role instead, choose **Create**, and then follow the on-screen instructions.

Enable AWS X-Ray

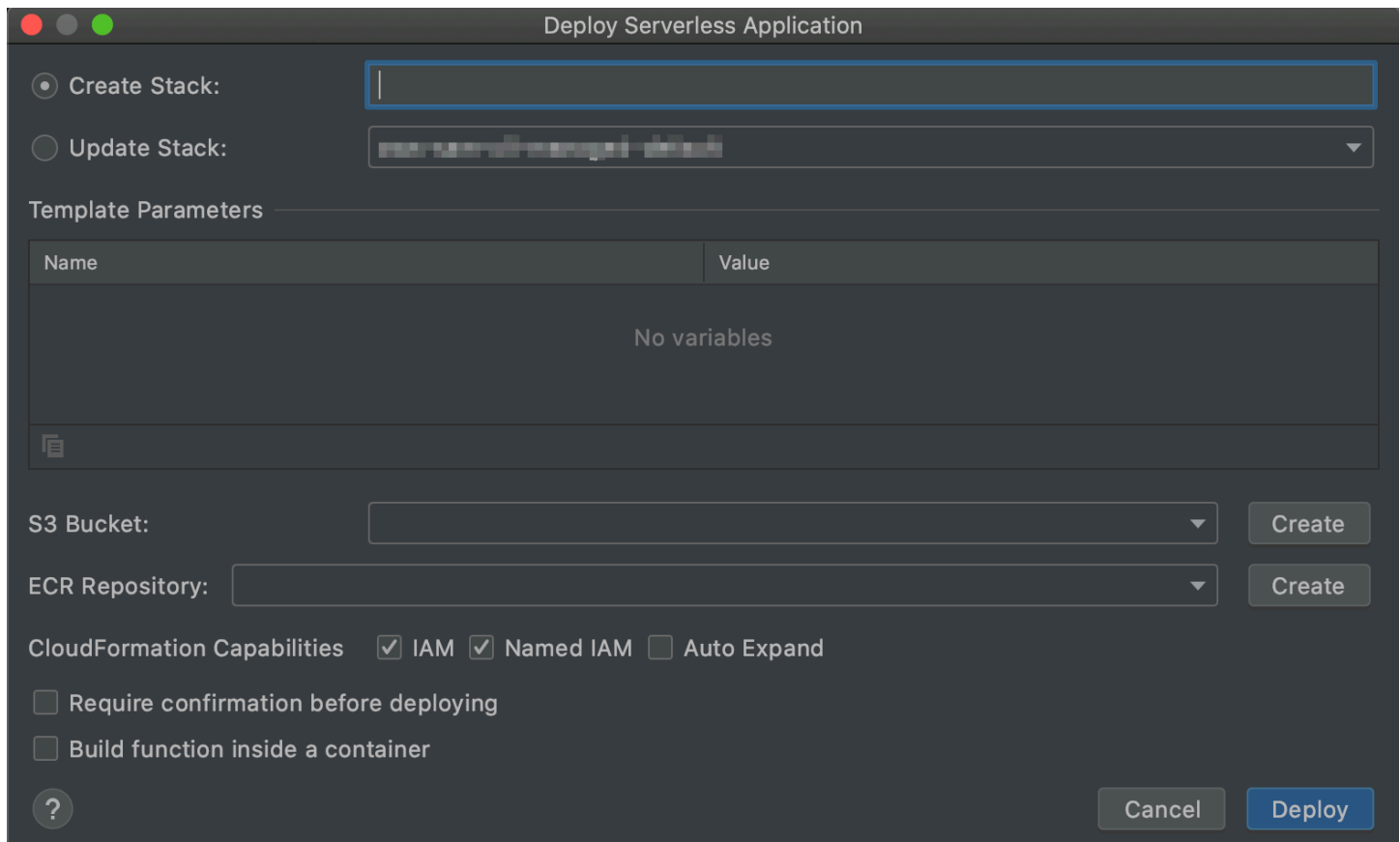
(Optional) If selected, [Lambda enables AWS X-Ray](#) to detect, analyze, and optimize performance issues with the function. X-Ray collects metadata from Lambda and any upstream or downstream services that make up your function. X-Ray uses this metadata to generate a detailed service graph that shows performance bottlenecks, latency spikes, and other issues that impact function performance.

Source Bucket

(Required) Choose an available Amazon Simple Storage Service (Amazon S3) bucket in the connected AWS account for the AWS Serverless Application Model (AWS SAM) command line interface (CLI) to use to deploy the function to Lambda. To create an Amazon S3 bucket in the account and have the AWS SAM CLI use that one instead, choose **Create**, and then follow the on-screen instructions.

Deploy Serverless Application dialog box

The **Deploy Serverless Application** dialog box in the AWS Toolkit for JetBrains is displayed when you deploy an AWS serverless application.



The **Deploy Serverless Application** dialog box contains the following items:

Create Stack

(Required) Provide the name of the stack for the AWS Serverless Application Model (AWS SAM) command line interface (CLI) to create in AWS CloudFormation for the connected AWS account. The AWS SAM CLI then uses this stack to deploy the AWS serverless application.

Update Stack

(Required) Choose the name of an existing AWS CloudFormation stack in the connected AWS account for the AWS SAM CLI to use to deploy the AWS serverless application.

Note

Either **Create Stack** or **Update Stack** is required, but not both.

Template Parameters

(Optional) Any parameters that the AWS Toolkit for JetBrains detects in the corresponding project's `template.yaml` file. To specify a value for a parameter, choose the box in the **Value** column next to the parameter, enter the value, and then press **Enter**. For more information, see [Parameters](#) in the *AWS CloudFormation User Guide*.

S3 Bucket

(Required) Choose an existing Amazon Simple Storage Service (Amazon S3) bucket in the connected AWS account for the AWS SAM CLI to use to deploy the AWS serverless application. To create an Amazon S3 bucket in the account and have the AWS SAM CLI use that bucket instead, choose **Create**, and then follow the on-screen instructions.

ECR Repository

(Required for Image package type only) Choose an existing Amazon Elastic Container Registry (Amazon ECR) repository URI in the connected AWS account for the AWS SAM CLI to use to deploy the AWS serverless application. For information about AWS Lambda package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

Require confirmation before deploying

(Optional) If selected, instructs AWS CloudFormation to wait for you to finish creating or updating the corresponding stack by [executing the stack's current change set in AWS CloudFormation](#). If you don't execute this change set, the AWS serverless application doesn't move on to the deployment phase.

Build function inside a container

(Optional) If selected, the AWS SAM CLI builds any of the serverless application's functions inside of a Lambda-like Docker container locally before deployment. This is useful if the function depends on packages that have natively compiled dependencies or programs. For more information, see [Building applications](#) in the *AWS Serverless Application Model Developer Guide*.

New Project dialog box

The **New Project** dialog box in the AWS Toolkit for JetBrains is displayed when you create an AWS serverless application.

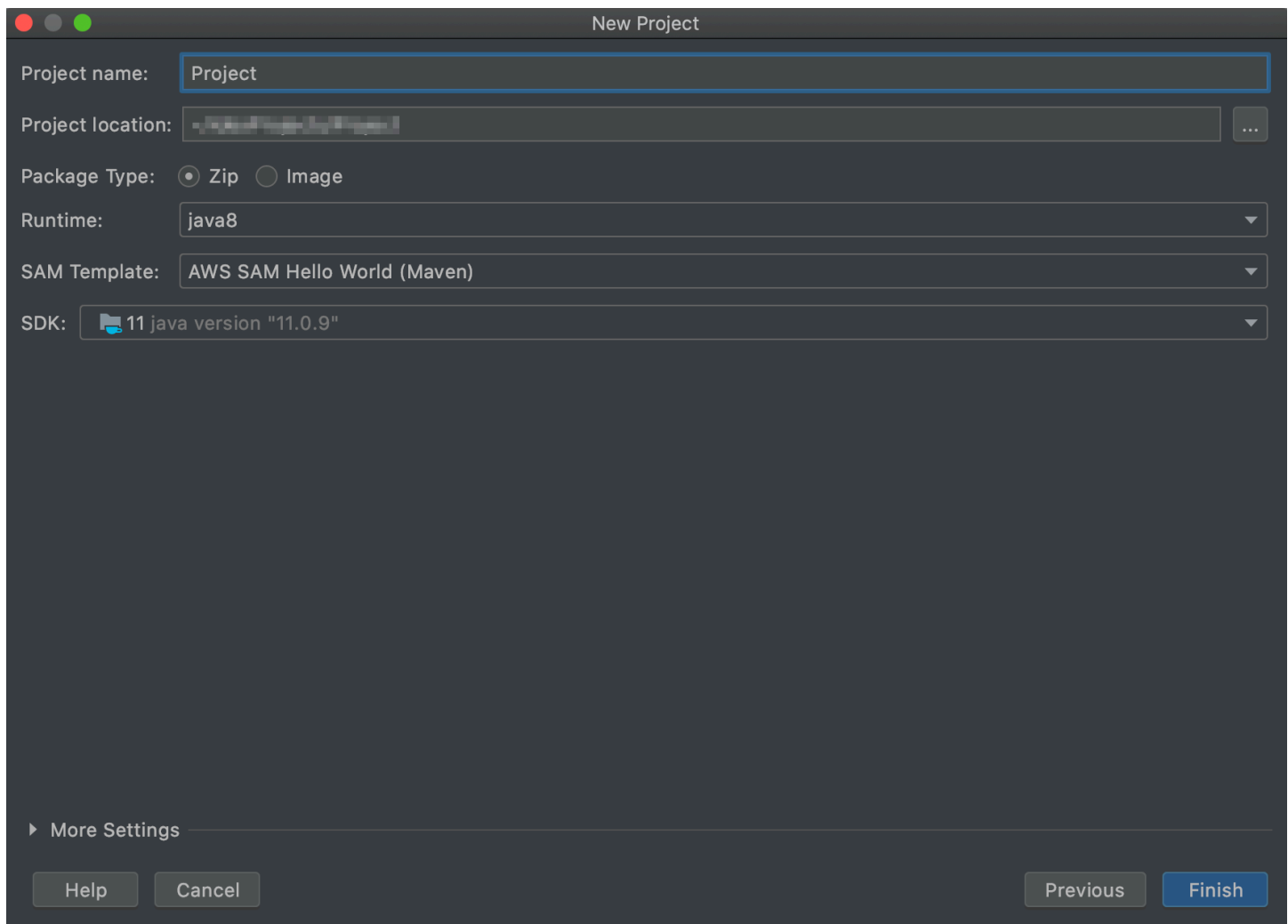
Topics

- [New Project dialog box \(IntelliJ IDEA, PyCharm, and WebStorm\)](#)
- [New Project dialog box \(JetBrains Rider\)](#)

New Project dialog box (IntelliJ IDEA, PyCharm, and WebStorm)

Note

The following screenshot shows the **New Project** dialog box for IntelliJ IDEA, but the field descriptions also apply to PyCharm and WebStorm.



The **New Project** dialog box contains the following items:

Project name

(Required) The name of the project.

Project location

(Required) The location where IntelliJ IDEA creates the project.

Package Type

(Required) The AWS Lambda function's deployment package type, which can be either Zip or Image. For information about the difference between Zip and Image package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

Runtime

(Required) The ID of the [Lambda runtime](#) to use.

SAM Template

(Required) The name of the AWS Serverless Application Model (AWS SAM) template to use.

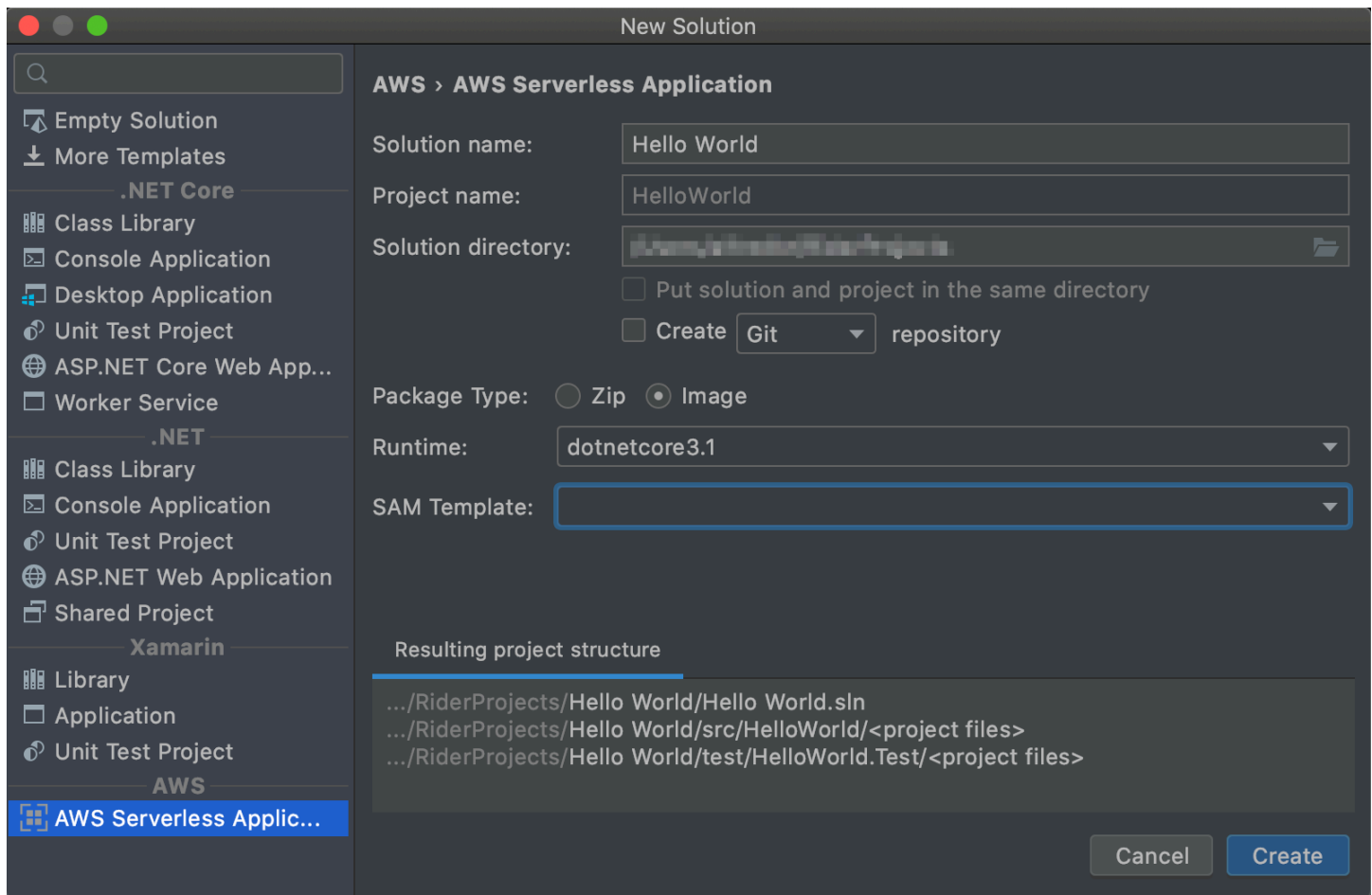
Project SDK

(Required) The Java development kit (JDK) to use. For more information, see [Java Development Kit \(JDK\)](#) on the IntelliJ IDEA Help website.

New Project dialog box (JetBrains Rider)

Note

When you create a new solution, this dialog box contains the title **New Solution** instead of **New Project**. However, the dialog box's contents are the same.



The **New Project** dialog box contains the following items:

Solution name

(Required) The name of the solution.

Project name

(Required) The name of the project.

Solution directory

(Required) The path to the solution's directory.

Put solution and project in the same directory

(Optional) If selected, puts the solution's files in the same location as the project's files.

Create repository

(Optional) If selected, creates a remote repository for the project with the specified provider.

Package Type

(Required) The Lambda function's package type, which can be either Zip or Image. For information about the difference between Zip and Image package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

Runtime

(Required) The ID of the Lambda runtime to use.

SAM Template

(Required) The name of the AWS SAM template to use.

Resulting project structure

(Non-editable) The paths for the created project's directories and files.

Run/Debug Configurations dialog box

The **Run/Debug Configurations** dialog box in the AWS Toolkit for JetBrains is displayed whenever you want to alter the run/debug configurations, whether locally, remotely, or in an Amazon Elastic Container Service (Amazon ECS) cluster.

Topics

- [Run/Debug Configurations dialog box \(local function settings\)](#)
- [Run/Debug Configurations dialog box \(remote function settings\)](#)
- [Edit configuration dialog box \(Amazon ECS cluster\)](#)

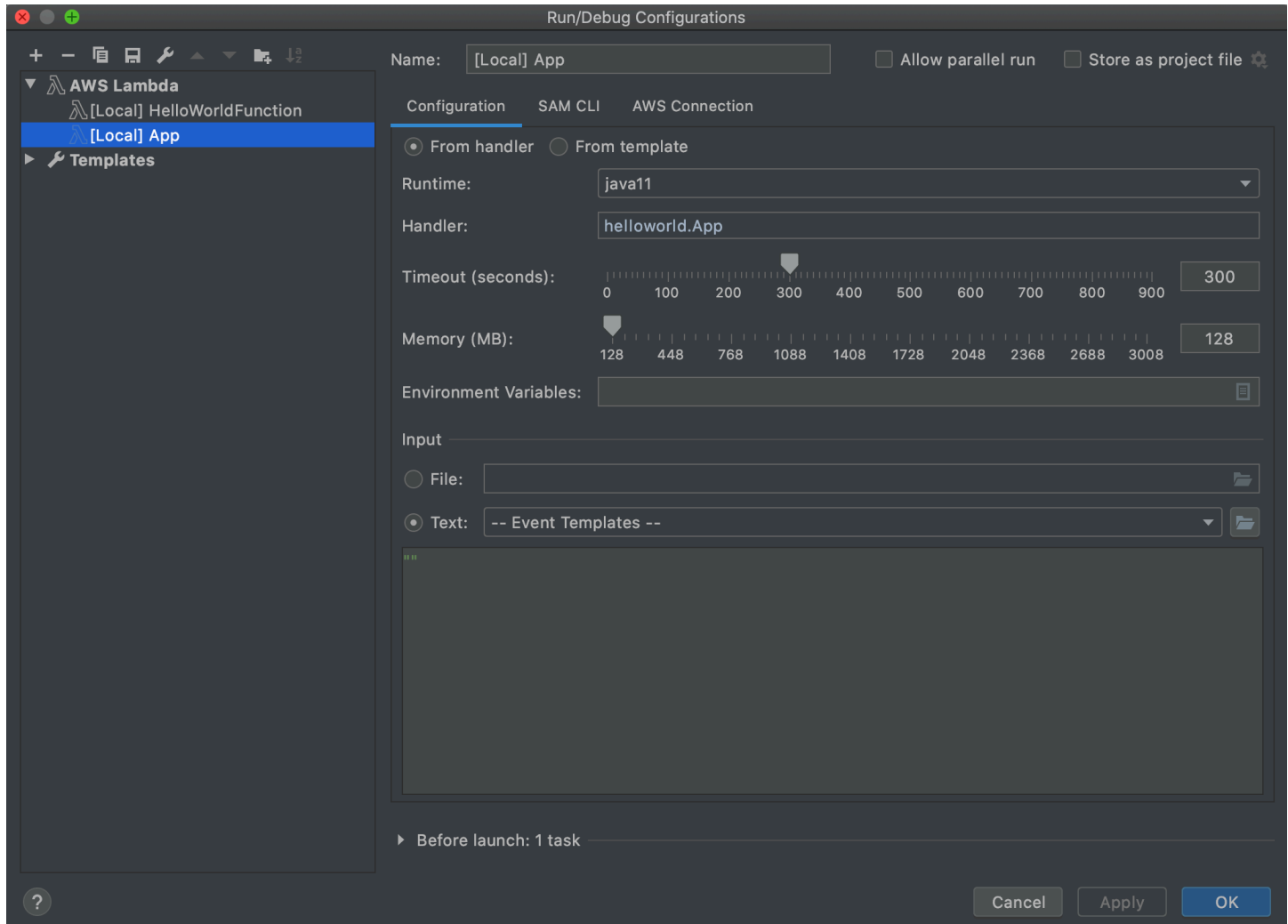
Run/Debug Configurations dialog box (local function settings)

This dialog box is displayed whenever you update settings for the *local* version of an AWS Lambda function.

Note

To update settings for the *remote* version of that same function (the function's source code is in Lambda in your AWS account), see [Run/Debug Configurations dialog box \(remote function settings\)](#) instead.

This dialog box contains three tabs: **Configuration**, **SAM CLI**, and **AWS Connection**.



The **Configuration** tab of the **Run/Debug Configurations** dialog box for local function settings contains the following items:

Name

(Required) The name of this configuration.

Allow parallel run / Allow running in parallel

(Optional) If selected, allows IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider to launch as many instances of the configuration to run in parallel as needed.¹

From handler / From template

(Required) Depending on which option you choose, you must configure additional settings.

Runtime

(Required) The ID of the [Lambda runtime](#) to use.

Handler

(Required for the **From handler** option) The identifier of the corresponding function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#).

Timeout (seconds)

(Required for the **From handler** option) The amount of time that Lambda allows a function to run before stopping it. Specify an amount up to 900 seconds (15 minutes).

Memory (MB)

(Required for the **From handler** option) The amount of memory available to the function as it runs. Specify an amount [between 128 MB and 3,008 MB](#) in 64-MB increments.

Environment Variables

(Optional for the **From handler** option) Any [environment variables](#) for the Lambda function to use, specified as key-value pairs. To add, change, or delete environment variables, choose the folder icon, and then follow the on-screen instructions.

Template

(Required for the **From template** option) The location and file name of the AWS Serverless Application Model (AWS SAM) template (for example, `template.yaml`) to use for this configuration, and the resource in that template to associate with this configuration.

File

(Required) The location and file name of the event data to pass to the function, in JSON format. For event data examples, see [Invoke the Lambda function](#) in the *AWS Lambda Developer Guide* and [Generating sample event payloads](#) in the *AWS Serverless Application Model Developer Guide*.

Text

(Required) The event data to pass to the function, in JSON format. For event data examples, see [Invoke the Lambda function](#) in the *AWS Lambda Developer Guide* and [Generating sample event payloads](#) in the *AWS Serverless Application Model Developer Guide*.

Note

Either **File** or **Text** is required, but not both.

Before launch: window

(Optional) Lists any tasks that must be performed before starting this configuration.²

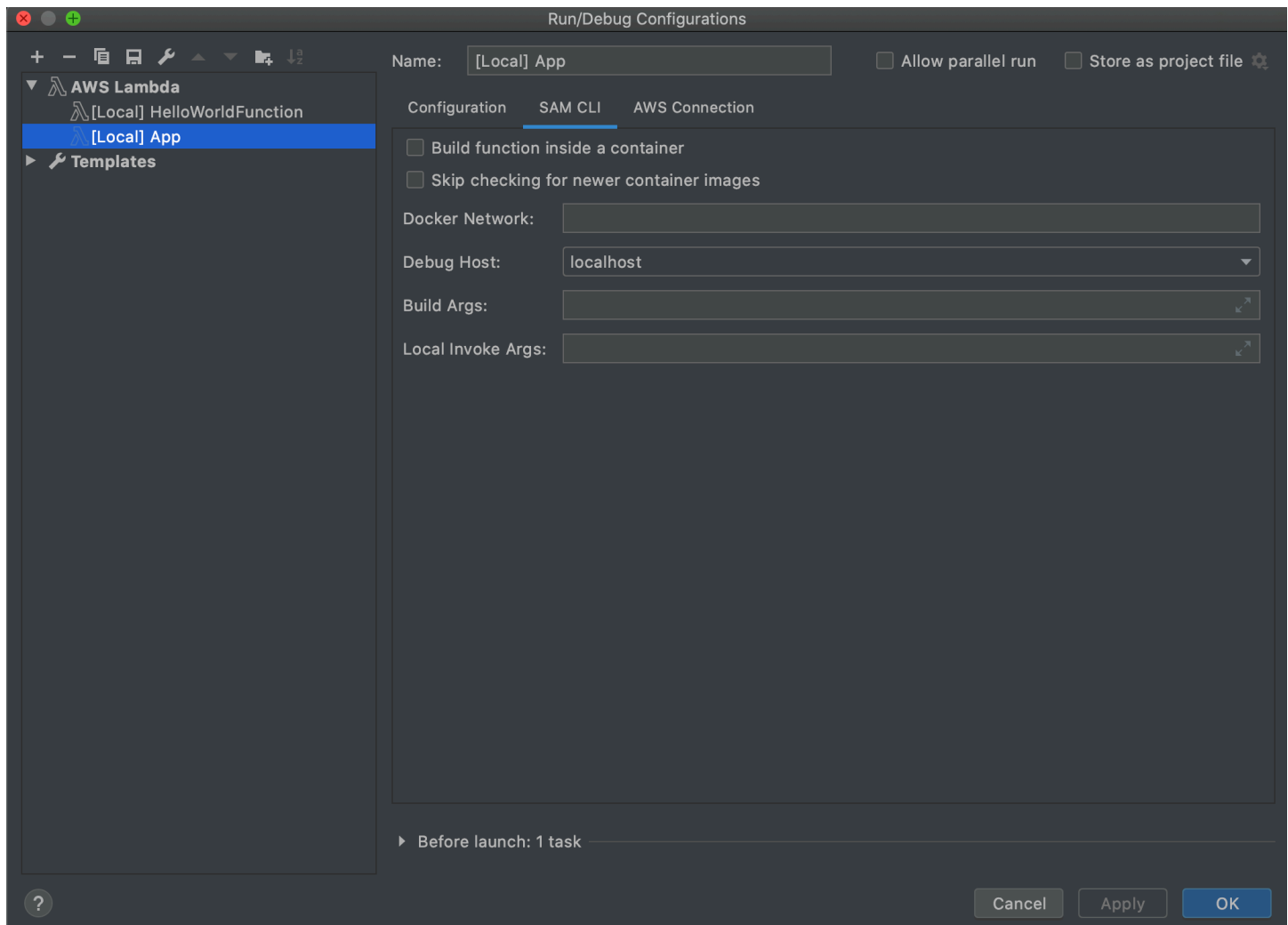
Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Common options](#) on the PyCharm Help website.
- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.



The **SAM CLI** tab of the **Run/Debug Configurations** dialog box for local function settings contains the following items:

Name

(Required) The name of this configuration.

Allow parallel run / Allow running in parallel

(Optional) If selected, allows IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider to launch as many instances of the configuration to run in parallel as needed.¹

Build function inside a container

(Optional) If selected, the AWS SAM CLI builds any of the serverless application's functions inside of a Lambda-like Docker container locally before deployment. This is useful if the

function depends on packages that have natively compiled dependencies or programs. For more information, see [Building applications](#) in the *AWS Serverless Application Model Developer Guide*.

Skip checking for newer container images

(Optional) If selected, the AWS SAM CLI skips pulling down the latest Docker image for the [runtime](#) that is specified on the **Configuration** tab.

Docker Network

(Optional) The name or ID of an existing Docker network for Lambda Docker containers to connect to, with the default bridge network. If not specified, the Lambda containers connect only to the default bridge Docker network.

Before launch: window

(Optional) Lists any tasks that must be performed before starting this configuration.²

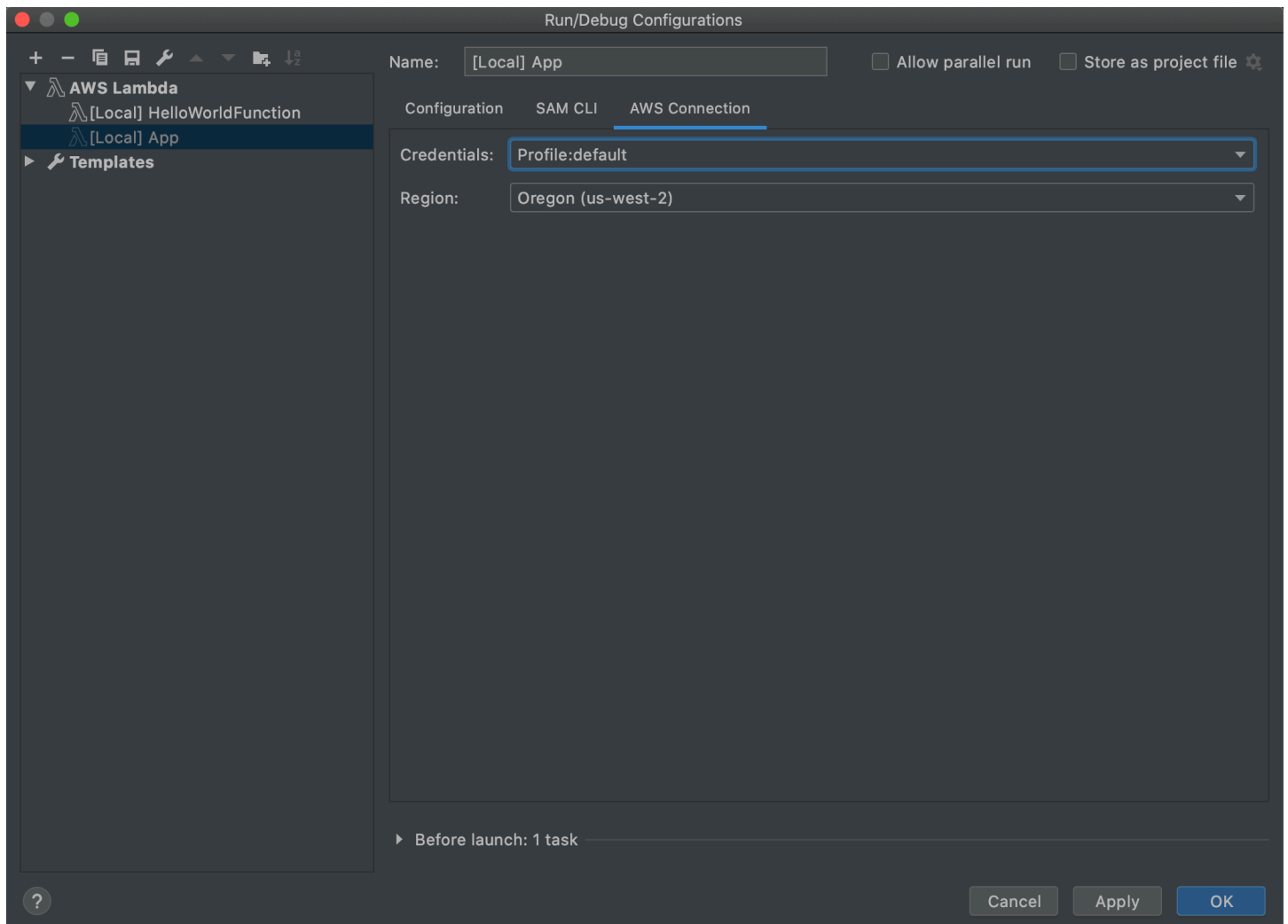
Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Common options](#) on the PyCharm Help website.
- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.



The **AWS Connection** tab of the **Run/Debug Configurations** dialog box for local function settings contains the following items:

Credentials

(Required) The name of the existing AWS account connection to use.

Region

(Required) The name of the AWS Region to use for the connected account.

Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Common options](#) on the PyCharm Help website.

- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.

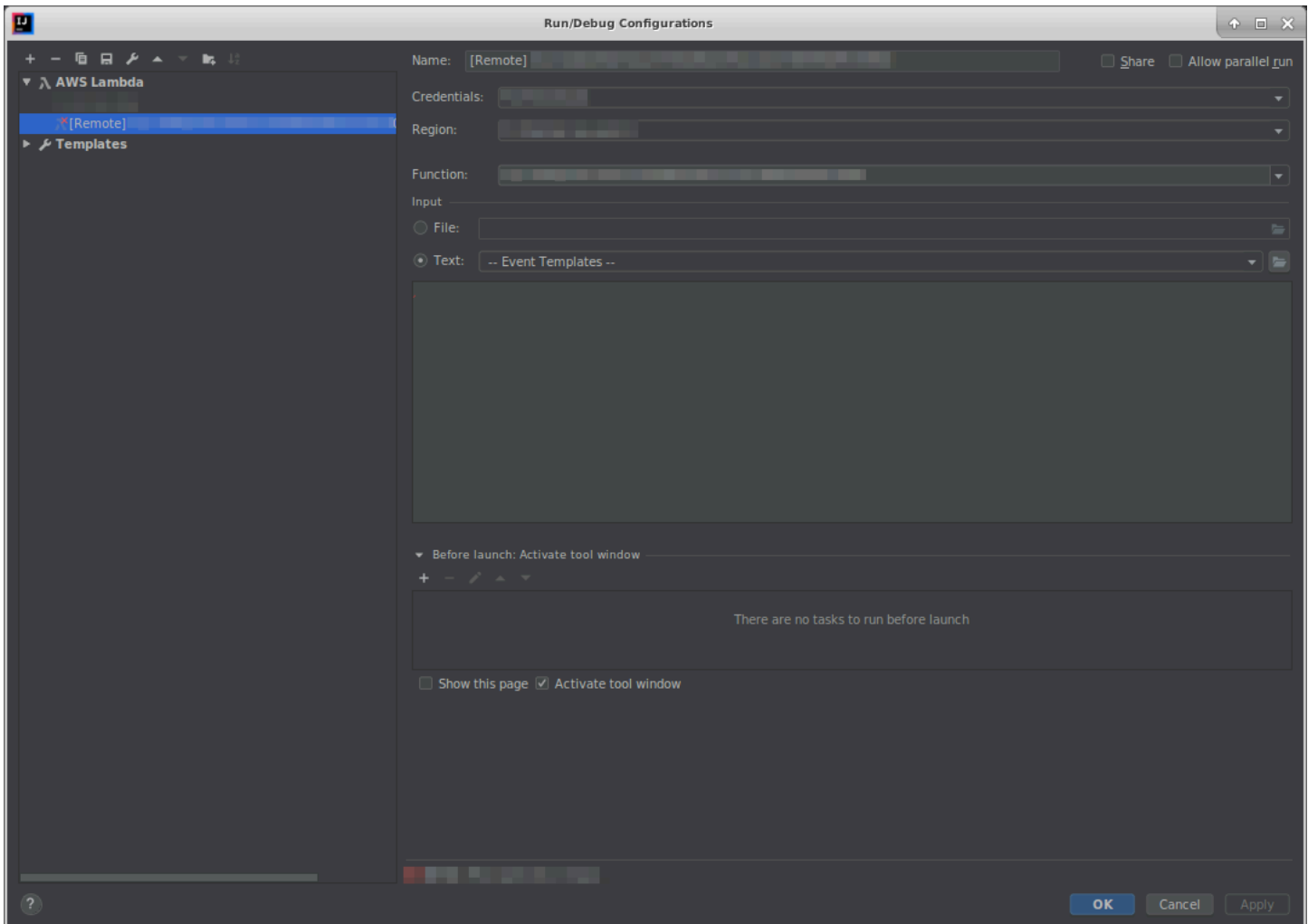
Run/Debug Configurations dialog box (remote function settings)

This dialog box displays whenever you update settings for the *remote* version of an AWS Lambda function (the function's source code is in Lambda in your AWS account).

Note

To update settings for the *local* version of that same function, see [Run/Debug Configurations dialog box \(local function settings\)](#) instead.

Although the name of the dialog box is **Run/Debug Configurations**, you cannot use the AWS Toolkit for JetBrains to debug the remote version of a Lambda function. You can only run the function.



The **Run/Debug Configurations** dialog box for remote function settings contains the following items:

Name

(Required) The name of this configuration.

Share / Share through VCS

(Optional) If selected, makes this configuration available to other team members.¹

Allow parallel run / Allow running in parallel

(Optional) If selected, allows IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider to launch as many instances of the configuration to run in parallel as needed.¹

Credentials

(Required) The name of the existing AWS account connection to use.

Region

(Required) The name of the AWS Region to use for the connected account.

Function

(Required) The name of the Lambda function to use.

File

(Required) The location and file name of the event data to pass to the function, in JSON format. For event data examples, see [Invoke the Lambda function](#) in the *AWS Lambda Developer Guide* and [Generating sample event payloads](#) in the *AWS Serverless Application Model Developer Guide*.

Text

(Required) The event data to pass to the function, in JSON format. For event data examples, see [Invoke the Lambda function](#) in the *AWS Lambda Developer Guide* and [Generating sample event payloads](#) in the *AWS Serverless Application Model Developer Guide*.

Note

Either **File** or **Text** is required, but not both.

Before launch: Activate tool window

(Optional) Lists any tasks that must be performed before starting this configuration.²

Show this page

(Optional) If selected, displays these configuration settings prior to starting this configuration.²

Activate tool window

(Optional) If selected, opens the **Run** or the **Debug** tool window when you start this configuration.²

Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.

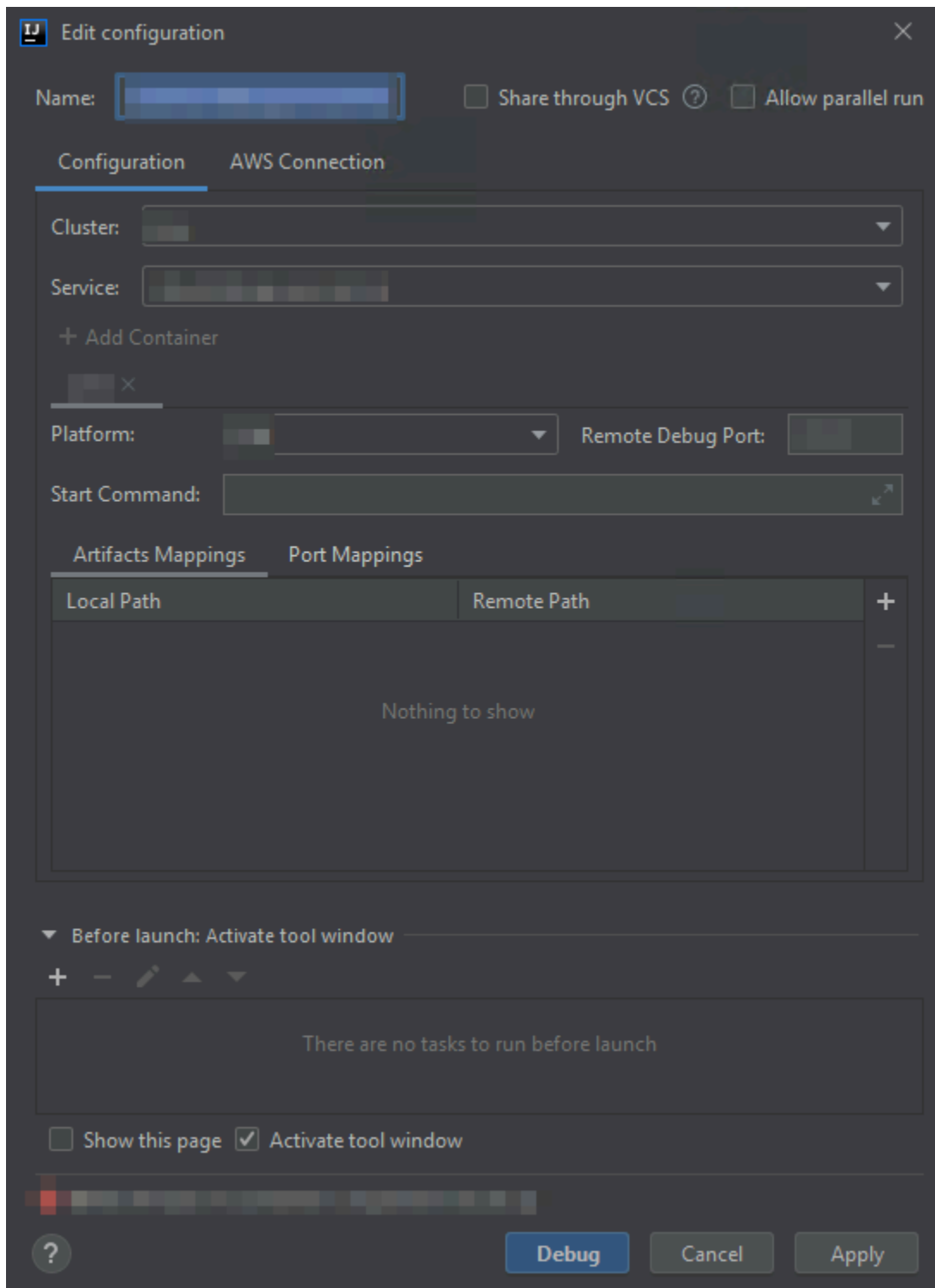
- For PyCharm, see [Common options](#) on the PyCharm Help website.
- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.

Edit configuration dialog box (Amazon ECS cluster)

The **Edit configuration** dialog box contains two tabs: **Configuration** and **AWS Connection**.



The **Configuration** tab of the **Edit configuration** dialog box contains the following items:

Name

(Required) The name of this configuration.

Share / Share through VCS

(Optional) If selected, makes this configuration available to other team members.¹

Allow parallel run / Allow running in parallel

(Optional) If selected, allows IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider to launch as many instances of the configuration to run in parallel as needed.¹

Cluster

(Required) The name of the Amazon Elastic Container Service (Amazon ECS) cluster to debug.

Service

(Required) The name of the Amazon ECS service in the cluster to debug.

Add Container

Adds a container to this configuration. Optional if at least one tab is already visible. Each tab represents a separate container.

The following items apply to the selected container: **Platform**, **Remote Debug Port**, **Start Command**, **Artifacts Mappings**, and **Port Mappings**.

Platform

(Required) The debug platform to use.

Remote Debug Port

(Optional) The port to attach to the debugger. Generally, you shouldn't specify this unless your service uses ports 20020-20030. If it does, specify that port here so that the container doesn't try to bind ports that might otherwise be in use elsewhere.

Start Command

(Required) The command to start your program so that the debugger can attach to it. For Java, it should start with `java` and contain no debugger information, such as `-Xdebug`. For Python, it must start with `python`, `python2`, or `python3`, followed by the path and name of the file to run.

Artifacts Mappings

(Required) A **Local Path** on your local development machine that maps to a **Remote Path** within the container. You must map all code and artifacts that you plan to run. To specify a local and remote path mapping, choose **Add** (the **+** icon).

Port Mappings

(Optional) A **Local Port** on your local development machine that maps to a **Remote Port** within the container. This enables local ports to communicate directly with ports on a remote resource. For example, for the command `curl localhost:3422`, port 3422 maps to some service. To specify a local and remote port mapping, choose **Add** (the + icon).

Before launch: Activate tool window

(Optional) Lists any tasks that must be performed before starting this configuration.²

Show this page

(Optional) If selected, displays these configuration settings before starting this configuration.²

Activate tool window

(Optional) If selected, opens the **Run** or **Debug** tool window when you start this configuration.²

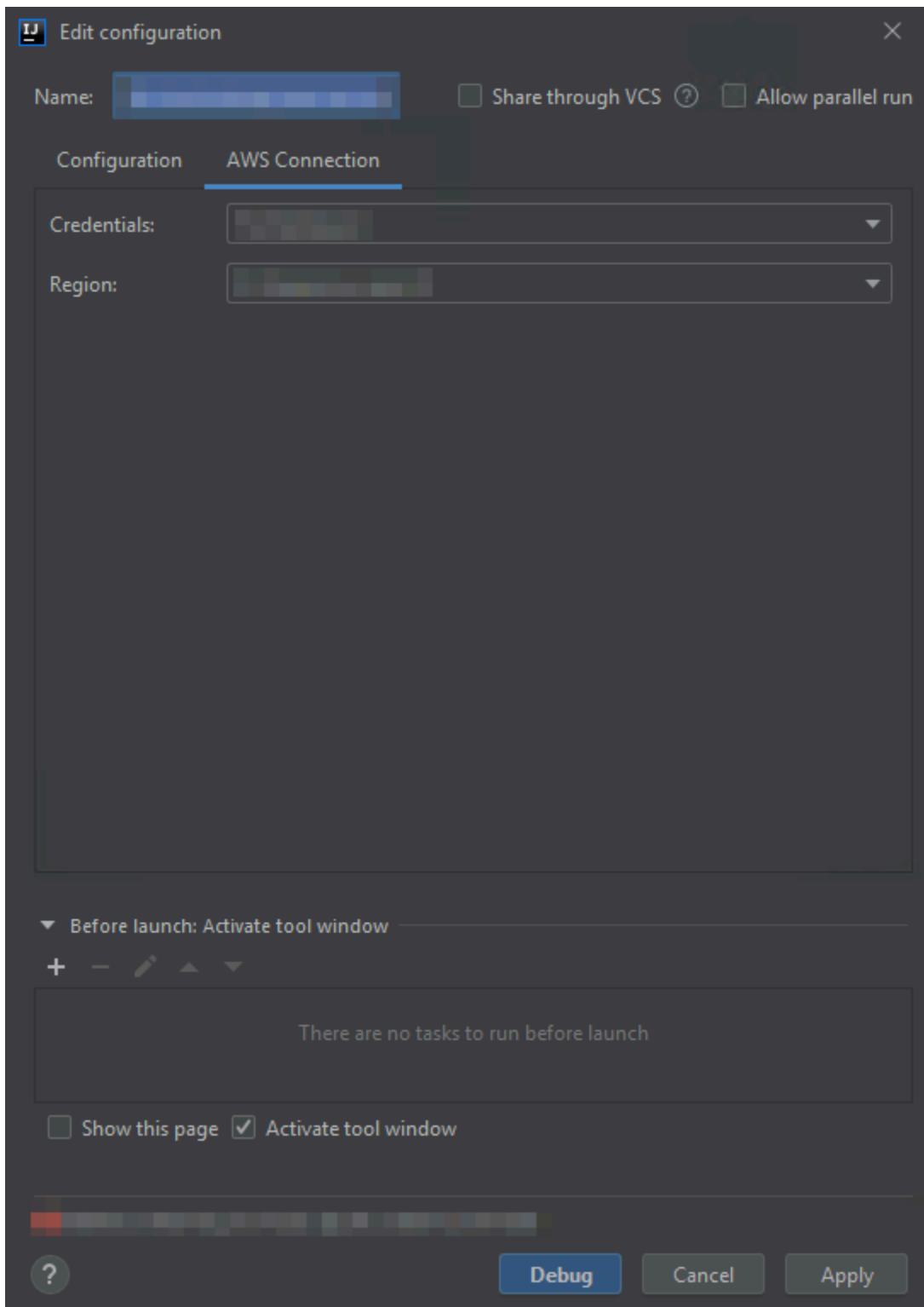
Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Common options](#) on the PyCharm Help website.
- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.



The **AWS Connection** tab of the **Edit configuration** dialog box contains the following items:

Name

(Required) The name of this configuration.

Credentials

(Required) The name of the existing AWS account connection to use.

Region

(Required) The name of the AWS Region to use for the connected account.

Share / Share through VCS

(Optional) If selected, makes this configuration available to other team members.¹

Allow parallel run / Allow running in parallel

(Optional) If selected, allows IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider to launch as many instances of the configuration to run in parallel as needed.¹

Before launch: Activate tool window

(Optional) Lists any tasks that must be performed before starting this configuration.²

Show this page

(Optional) If selected, displays these configuration settings before starting this configuration.²

Activate tool window

(Optional) If selected, opens the **Run** or **Debug** tool window when you start this configuration.²

Notes

¹ For more information, see the following:

- For IntelliJ IDEA, see [Common options](#) on the IntelliJ IDEA Help website.
- For PyCharm, see [Common options](#) on the PyCharm Help website.
- For WebStorm, see [Common options](#) on the WebStorm Help website.
- For JetBrains Rider, see [Common options](#) on the JetBrains Rider Help website.

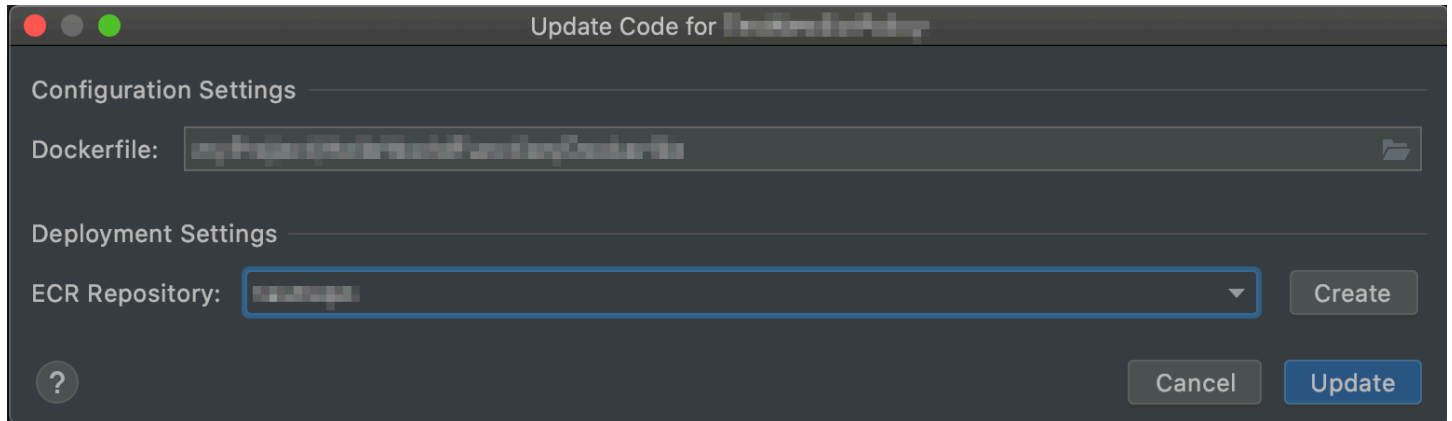
² For more information, see the following:

- For IntelliJ IDEA, see [Before Launch options](#) on the IntelliJ IDEA Help website.

- For PyCharm, see [Before Launch options](#) on the PyCharm Help website.
- For WebStorm, see [Before Launch options](#) on the WebStorm; Help website.
- For JetBrains Rider, see [Before Launch options](#) on the JetBrains Rider Help website.

Update Code dialog box

The **Update Code** dialog box in the AWS Toolkit for JetBrains is displayed whenever you update an AWS Lambda function.



The **Update Code** dialog box contains the following items:

Handler

(Required) The ID of the corresponding Lambda function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#).

Source Bucket

(Required for Zip package type only) Choose an existing Amazon Simple Storage Service (Amazon S3) bucket in the connected AWS account for the AWS Serverless Application Model (AWS SAM) command line interface (CLI) to use to deploy the function to Lambda. To create an Amazon S3 bucket in the account and have the AWS SAM CLI use that bucket instead, choose **Create**, and then follow the on-screen instructions. For information about Lambda package types, see [Lambda deployment packages](#) in the *AWS Lambda Developer Guide*.

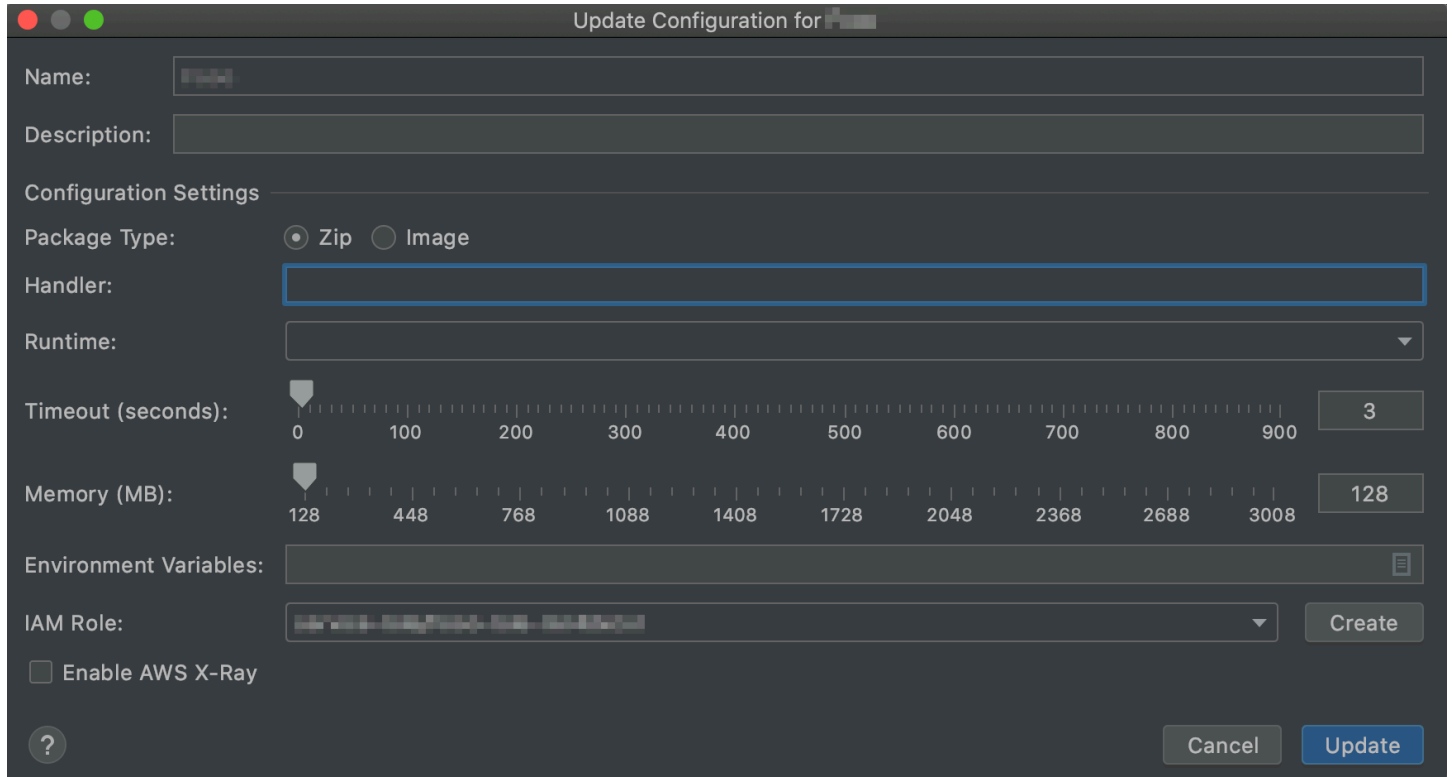
ECR Repository

(Required for Image package type only) Choose an existing Amazon Elastic Container Registry (Amazon ECR) repository in the connected AWS account for the AWS SAM CLI to use to deploy the function to Lambda.

Update Configuration dialog box

The **Update Configuration** dialog box in the AWS Toolkit for JetBrains is displayed whenever you update the configuration for an AWS Lambda function. The information that you provide differs slightly depending on whether the project's Lambda function is of package type Zip or Image.

The **Update Configuration** dialog box for the Zip package type:



The screenshot shows the "Update Configuration for [redacted]" dialog box. It features a dark theme with the following fields and controls:

- Name:** A text input field containing "lambda-1234".
- Description:** An empty text input field.
- Configuration Settings:** A section header.
- Package Type:** Radio buttons for "Zip" (selected) and "Image".
- Handler:** An empty text input field.
- Runtime:** A dropdown menu.
- Timeout (seconds):** A slider ranging from 0 to 900, with a numeric input field set to "3".
- Memory (MB):** A slider ranging from 128 to 3008, with a numeric input field set to "128".
- Environment Variables:** An empty text input field with a list icon on the right.
- IAM Role:** A dropdown menu showing a role name, with a "Create" button to its right.
- Enable AWS X-Ray:** An unchecked checkbox.
- Buttons:** A help icon (?), a "Cancel" button, and an "Update" button.

The **Update Configuration** dialog box for the Image package type:

Update Configuration for [Name]

Name: [Text Field]

Description: [Text Field]

Configuration Settings

Package Type: Zip Image

Timeout (seconds): [Slider 0-900] 300

Memory (MB): [Slider 128-3008] 128

Environment Variables: [Text Field] [List Icon]

IAM Role: [Dropdown Menu] [Create]

Enable AWS X-Ray

[?] [Cancel] [Update]

The **Update Configuration** dialog box contains the following items:

Name

(Required) The function's name. Can contain only the uppercase letters A through Z, the lowercase letters a through z, the numbers 0 through 9, hyphens (-), and underscores (_). The name must be less than 64 characters in length.

Description

(Optional) Any meaningful description about the function.

Package Type

(Required) The Lambda function's package type, which can be either Zip or Image.

Handler

(Required for Zip packages only) The ID of the corresponding Lambda function handler for [Java](#), [Python](#), [Node.js](#), or [C#](#).

Runtime

(Required for Zip packages only) The ID of the [Lambda runtime](#) to use.

Timeout (seconds)

(Required) The amount of time that Lambda allows a function to run before stopping it. Specify an amount up to 900 seconds (15 minutes).

Memory (MB)

(Required) The amount of memory available to the function as it runs. Specify an amount [between 128 MB and 3,008 MB](#) in 64-MB increments.

Environment Variables

(Optional) Any [environment variables](#) for the Lambda function to use, specified as key-value pairs. To add, change, or delete environment variables, choose the folder icon, and then follow the on-screen instructions.

IAM Role

(Required) Choose an available [Lambda execution role](#) in the connected AWS account for Lambda to use for the function. To create an execution role in the account and have Lambda use that role instead, choose **Create**, and then follow the on-screen instructions.

Enable AWS X-Ray

(Optional) If selected, [Lambda enables AWS X-Ray](#) to detect, analyze, and optimize performance issues with the function. X-Ray collects metadata from Lambda and any upstream or downstream services that make up your function. X-Ray uses this metadata to generate a detailed service graph that shows performance bottlenecks, latency spikes, and other issues that impact function performance.

Troubleshooting the AWS Toolkit for JetBrains

The following sections contain general troubleshooting information about the AWS Toolkit for JetBrains.

Topics

- [Troubleshooting best practices](#)
- [Unable to sign out of AWS IAM Identity Center](#)
- [Amazon Q Developer is unavailable for JetBrains 2023.2.0 releases](#)
- [Amazon Q Developer Code Transformation fails to start at sign in](#)

Troubleshooting best practices

The following are recommended best practices when troubleshooting AWS Toolkit for JetBrains issues.

- Attempt to recreate your issue or error prior to sending a report.
- Take detailed notes of each step, setting, and error message during the recreation process.
- Collect AWS Toolkit Logs. For a detailed description of how to locate your AWS Toolkit logs, see the [How to locate your AWS logs](#) procedure, located in this guide topic.
- Check for open requests, known solutions, or report your unresolved issue in the [AWS Toolkit for JetBrains Issues](#) section of the AWS Toolkit for JetBrains GitHub repository.

How to locate your AWS Toolkit logs

1. From the JetBrains main menu, expand **Help**.
2. Choose **Show Log in Explorer** to view the location of the current log file on your hard drive.
3. Copy this log file and include it when creating a new AWS Toolkit for JetBrains issue or request.

Unable to sign out of AWS IAM Identity Center

If you're unable to sign out of your IAM Identity Center credentials after choosing **Sign Out** from the **AWS Connection Settings** menu, and the **expired or invalid** notification is still visible, completing the following steps.

Clearing your IAM Identity Center cache file

1. From your operating system, locate the IAM Identity Center cache file. The default location for the cache file is: `.aws/ssso/cache`.
2. Delete the contents in your cache folder.
3. Restart your JetBrains IDE.
4. You're signed out of your IAM Identity Center credentials.

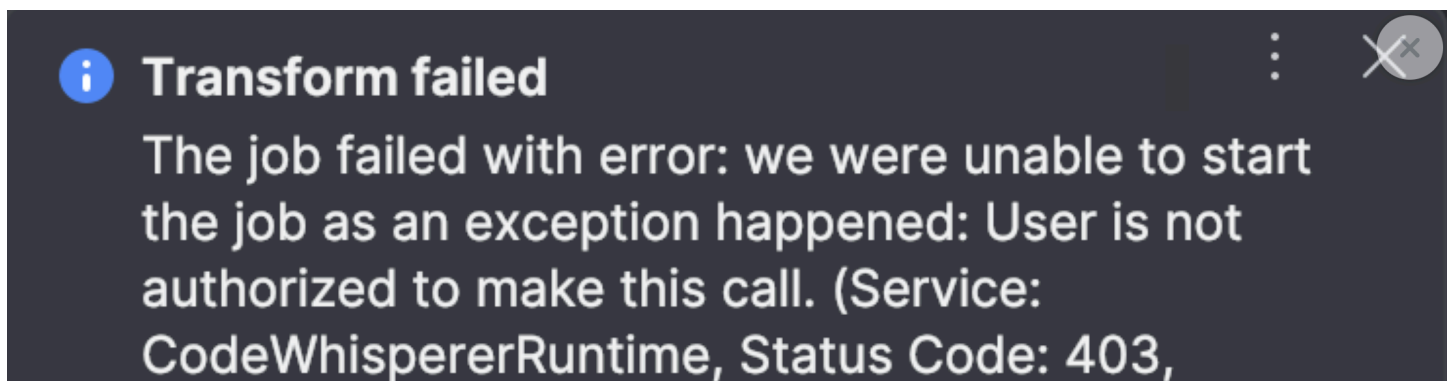
Amazon Q Developer is unavailable for JetBrains 2023.2.0 releases

Amazon Q Developer is unavailable for JetBrains 2023.2.0 releases. This is due to a specific issue with the 2023.2.0 release of JetBrains IDEs. For additional information about this issue, see [Issue JBR-5850](#) in JetBrains issue tracking.

To address this issue, install a later release of JetBrains (2023.2.1 or newer).

Amazon Q Developer Code Transformation fails to start at sign in

If Amazon Q Developer Code Transformation fails to start at sign in and you receive the following error message (see image below), you need to re-authenticate your credentials. Re-authenticate, then initiate Amazon Q Developer Code Transformation.



Security for this AWS Product or Service

Cloud security at Amazon Web Services (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 all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS Compliance Programs](#).

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Topics

- [Data protection in AWS Toolkit for JetBrains](#)
- [Identity and Access Management](#)
- [Compliance Validation for this AWS Product or Service](#)
- [Resilience for this AWS Product or Service](#)
- [Infrastructure Security for this AWS Product or Service](#)

Data protection in AWS Toolkit for JetBrains

The AWS [shared responsibility model](#) applies to data protection in AWS Toolkit for JetBrains. 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 AWS Toolkit for JetBrains 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.

Identity and Access Management

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 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 services work with IAM](#)
- [Troubleshooting AWS identity and access](#)

Audience

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

Service user – If you use AWS services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS 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, see [Troubleshooting AWS identity and access](#) or the user guide of the AWS service you are using.

Service administrator – If you're in charge of AWS resources at your company, you probably have full access to AWS. It's your job to determine which AWS 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, see the user guide of the AWS service you are using.

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. To view example AWS identity-based policies that you can use in IAM, see the user guide of the AWS service you are using.

Authenticating with identities

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

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

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

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

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

AWS account root user

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

Federated identity

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

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

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

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

IAM users and groups

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

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

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

IAM roles

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

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

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

To get a high-level view of how AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

To learn how to use a specific AWS service with IAM, see the security section of the relevant service's User Guide.

Troubleshooting AWS identity and access

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

Topics

- [I am not authorized to perform an action in AWS](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my AWS resources](#)

I am not authorized to perform an action in AWS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but doesn't have the fictional `awes:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awes:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the `my-example-widget` resource by using the `awes:GetWidget` action.

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

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.

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. 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 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 supports these features, see [How AWS services work with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Compliance Validation for this AWS Product or Service

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

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

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

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

Note

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Resilience for this AWS Product or Service

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Infrastructure Security for this AWS Product or Service

This AWS product or service uses managed services, and therefore is protected by the 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 this AWS Product or Service 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.

This AWS product or service follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Document history for the AWS Toolkit for JetBrains User Guide

The following table lists key documentation updates for the *AWS Toolkit for JetBrains User Guide*.

For a detailed list of changes to the AWS Toolkit for JetBrains, see the [.changes](#) directory in the [aws/aws-toolkit-jetbrains](#) repository on the GitHub website.

Change	Description	Date
Created user guide for Amazon Elastic Container Service Exec	This is an overview of the Amazon ECS Exec.	December 16, 2021
Support for experimental features	Added support for turning on experimental features for AWS services.	October 14, 2021
Support for AWS resources	Added support for accessing resource types along with interface options to create, edit, and delete resources.	October 14, 2021
Working with AWS App Runner now available	Using the AWS Toolkit for JetBrains to work with App Runner to deploy from source code or a container image directly to a scalable and secure web application in the AWS Cloud.	May 26, 2021
Working with Lambda container images with serverless applications now available	Using the AWS Toolkit to work with AWS Lambda container images with serverless applications is now available.	December 1, 2020

[Working with CloudWatch Logs Insights now available](#)

Using the AWS Toolkit for JetBrains to work with CloudWatch Logs Insights is now available.

November 24, 2020

[Working with Amazon SQS now available](#)

Using the AWS Toolkit for JetBrains to work with Amazon Simple Queue Service (Amazon SQS) is now available.

November 24, 2020

[Working with Amazon RDS and Amazon Redshift now available](#)

Using the AWS Toolkit to work with Amazon Relational Database Service (Amazon RDS) and Amazon Redshift is now available.

September 23, 2020

[Support for IAM Identity Center now available](#)

Support for AWS IAM Identity Center now available in the AWS Toolkit.

September 23, 2020

[AWS Toolkits now available for four more JetBrains IDEs](#)

AWS Toolkits are now available as plugins for four additional JetBrains IDEs:

May 28, 2020

- [AWS Toolkit for CLion](#) (for C & C++ development)
- [AWS Toolkit for GoLand](#) (for Go development)
- [AWS Toolkit for PhpStorm](#) (for PHP development)
- [AWS Toolkit for RubyMine](#) (for Ruby development)

[Working with CloudWatch Logs now available](#)

Using the AWS Toolkit to work with Amazon CloudWatch Logs is now available.

April 15, 2020

Working with Amazon S3 buckets and objects now available	Using the AWS Toolkit to work with Amazon Simple Storage Service (Amazon S3) buckets and objects is now available.	March 27, 2020
Working with EventBridge Schemas now available	Using the AWS Toolkit to work with Amazon EventBridge Schemas is now available.	December 2, 2019
Debugging code in Amazon ECS clusters now available in beta	Using the AWS Toolkit to debug code in Amazon Elastic Container Service (Amazon ECS) clusters is now available in beta.	November 25, 2019
AWS Toolkit for Rider now available	The AWS Toolkit for Rider is now available.	November 25, 2019
AWS Toolkit for WebStorm now available	The AWS Toolkit for WebStorm is now available.	October 23, 2019
AWS Toolkit for IntelliJ now generally available	The AWS Toolkit for IntelliJ is now generally available. The corresponding documentation has been refreshed accordingly.	March 27, 2019
Initial release	This is the initial release of the <i>AWS Toolkit for JetBrains User Guide</i> . The AWS Toolkit for PyCharm is now generally available. The AWS Toolkit for IntelliJ is still in Developer Preview.	November 27, 2018