Table of Contents

What is the AWS Toolkit for JetBrains? ........................................................................................................ 1
What the AWS Toolkit for JetBrains includes ............................................................................................... 1
How to get started ....................................................................................................................................... 1
What you can do with the AWS Toolkit for JetBrains ................................................................................. 2
Related information ..................................................................................................................................... 2
  Related videos ........................................................................................................................................ 2
  Related webpages ................................................................................................................................... 2
  Questions and help ................................................................................................................................... 2
  Report a bug with the AWS Toolkit or make a feature request .............................................................. 3
  Contribute to the AWS Toolkit ................................................................................................................ 3
Key tasks .................................................................................................................................................... 4
  Install the AWS Toolkit for JetBrains .................................................................................................... 4
  Update the AWS Toolkit for JetBrains .................................................................................................. 6
  Configure the AWS Toolkit for JetBrains to Use an HTTP Proxy .......................................................... 7
  Work with connections from the AWS Toolkit for JetBrains to AWS accounts ..................................... 8
    Connect to an AWS Account for the first time ....................................................................................... 8
    Add multiple connections .................................................................................................................... 10
    Switch between connections .............................................................................................................. 11
    Change connection settings ............................................................................................................... 12
    Delete a connection ............................................................................................................................ 12
  Get the current connection .................................................................................................................... 13
  Get the current AWS Region .................................................................................................................. 14
  Switch between AWS Regions ................................................................................................................ 15
  Open AWS Explorer within the AWS Toolkit for JetBrains .................................................................... 16
  Work with AWS Serverless Applications ............................................................................................... 17
    Create a serverless application ......................................................................................................... 17
    Deploy a serverless application ........................................................................................................... 21
    Change (update) the settings for a serverless application .................................................................. 22
    Delete a serverless application ............................................................................................................ 23
  Work with AWS Lambda Functions ......................................................................................................... 24
    Create a function ................................................................................................................................. 25
    Run (invoke) or debug the local version of a function ......................................................................... 26
    Run (invoke) the remote version of a function ..................................................................................... 27
    Change (update) the configuration for a function ............................................................................... 28
    Delete a function ................................................................................................................................. 30
  Work with AWS CloudFormation stacks .................................................................................................. 30
    Create a stack ....................................................................................................................................... 31
    Change stack settings ......................................................................................................................... 31
    View event logs for a stack .................................................................................................................. 31
    Deleting a stack .................................................................................................................................... 32
  Work with Amazon CloudWatch Logs ..................................................................................................... 33
  Work with Amazon ECS clusters ............................................................................................................. 33
    Debug code in a cluster ....................................................................................................................... 34
  Working with Amazon EventBridge schemas ......................................................................................... 35
  Work with Amazon S3 buckets and objects ............................................................................................ 35
Setting up .................................................................................................................................................... 36
  Installing the AWS Toolkit ..................................................................................................................... 36
    Installing AWS Toolkit for JetBrains Early Access Program (EAP) builds ........................................ 38
  Updating the AWS Toolkit ..................................................................................................................... 38
  Setting AWS credentials ......................................................................................................................... 39
    Connecting for the first time ............................................................................................................... 39
    Getting the current connection ......................................................................................................... 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>107</td>
</tr>
<tr>
<td>User interface reference</td>
<td>80</td>
</tr>
<tr>
<td>Working with AWS services</td>
<td>49</td>
</tr>
<tr>
<td>AWS Serverless</td>
<td>49</td>
</tr>
<tr>
<td>Creating an application</td>
<td>49</td>
</tr>
<tr>
<td>Deploying an application</td>
<td>53</td>
</tr>
<tr>
<td>Changing (updating) application settings</td>
<td>54</td>
</tr>
<tr>
<td>Deleting an application</td>
<td>55</td>
</tr>
<tr>
<td>AWS Lambda</td>
<td>56</td>
</tr>
<tr>
<td>Creating a function</td>
<td>57</td>
</tr>
<tr>
<td>Running (invoking) or debugging a local function</td>
<td>58</td>
</tr>
<tr>
<td>Running (invoking) a remote function</td>
<td>59</td>
</tr>
<tr>
<td>Changing (updating) function settings</td>
<td>60</td>
</tr>
<tr>
<td>Deleting a function</td>
<td>62</td>
</tr>
<tr>
<td>AWS CloudFormation</td>
<td>62</td>
</tr>
<tr>
<td>Viewing event logs for a stack</td>
<td>63</td>
</tr>
<tr>
<td>Deleting a stack</td>
<td>64</td>
</tr>
<tr>
<td>Amazon CloudWatch Logs</td>
<td>64</td>
</tr>
<tr>
<td>Working with CloudWatch log events</td>
<td>65</td>
</tr>
<tr>
<td>Amazon ECS</td>
<td>68</td>
</tr>
<tr>
<td>Debugging code in a cluster</td>
<td>68</td>
</tr>
<tr>
<td>Amazon EventBridge</td>
<td>74</td>
</tr>
<tr>
<td>Working with Amazon EventBridge schemas</td>
<td>75</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>77</td>
</tr>
<tr>
<td>Working with Amazon S3 buckets</td>
<td>77</td>
</tr>
<tr>
<td>Working with Amazon S3 objects</td>
<td>78</td>
</tr>
<tr>
<td>User interface reference</td>
<td>80</td>
</tr>
<tr>
<td>AWS Explorer</td>
<td>80</td>
</tr>
<tr>
<td>Create Function dialog box</td>
<td>83</td>
</tr>
<tr>
<td>Deploy Serverless Application dialog box</td>
<td>84</td>
</tr>
<tr>
<td>Edit Configuration dialog box</td>
<td>85</td>
</tr>
<tr>
<td>Edit Configuration dialog box (Lambda function)</td>
<td>85</td>
</tr>
<tr>
<td>Edit Configuration dialog box (Amazon ECS cluster)</td>
<td>89</td>
</tr>
<tr>
<td>New Project dialog box</td>
<td>93</td>
</tr>
<tr>
<td>New Project dialog box (IntelliJ IDEA)</td>
<td>94</td>
</tr>
<tr>
<td>New Project dialog box (PyCharm)</td>
<td>95</td>
</tr>
<tr>
<td>New Project dialog box (WebStorm)</td>
<td>96</td>
</tr>
<tr>
<td>New Project dialog box (JetBrains Rider)</td>
<td>96</td>
</tr>
<tr>
<td>Run/Debug Configurations dialog box</td>
<td>98</td>
</tr>
<tr>
<td>Run/Debug Configurations dialog box (local function settings)</td>
<td>98</td>
</tr>
<tr>
<td>Run/Debug Configurations dialog box (remote function settings)</td>
<td>102</td>
</tr>
<tr>
<td>Update Code dialog box</td>
<td>104</td>
</tr>
<tr>
<td>Update Configuration dialog box</td>
<td>105</td>
</tr>
<tr>
<td>Security</td>
<td>107</td>
</tr>
<tr>
<td>Data protection</td>
<td>107</td>
</tr>
<tr>
<td>Identity and Access Management</td>
<td>108</td>
</tr>
<tr>
<td>Compliance Validation</td>
<td>108</td>
</tr>
<tr>
<td>Resilience</td>
<td>109</td>
</tr>
<tr>
<td>Infrastructure Security</td>
<td>109</td>
</tr>
<tr>
<td>Document history</td>
<td>110</td>
</tr>
</tbody>
</table>
What is the AWS Toolkit for JetBrains?

To start using the AWS Toolkit for JetBrains right away, skip ahead to the installation (p. 4) and first-time connection (p. 8) instructions.

The AWS Toolkit for JetBrains is an open source plugin for the integrated development environments (IDEs) from JetBrains. The toolkit makes it easier for developers to develop, debug, and deploy serverless applications that use Amazon Web Services (AWS).

Topics
- What the AWS Toolkit for JetBrains includes (p. 1)
- How to get started (p. 1)
- What you can do with the AWS Toolkit for JetBrains (p. 2)
- Related information (p. 2)

What the AWS Toolkit for JetBrains includes

The 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)

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.

How to get started

To start using the AWS Toolkit for JetBrains, follow the installation (p. 4) and first-time connection (p. 8) instructions.
After you install the AWS Toolkit and connect it to an AWS account, you can use it to work with
AWS serverless applications (p. 17), AWS Lambda functions (p. 24), AWS CloudFormation
stacks (p. 30), and Amazon ECS clusters (p. 33) in that account.

For brief instructions about how to use other available AWS Toolkit features, see the key tasks (p. 4).

What you can do with the AWS Toolkit for JetBrains

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

- Create (p. 17), deploy (p. 21), change (p. 22), and delete (p. 23) AWS serverless applications
  in an AWS account.
- Create (p. 25), run (invoke) and debug locally (p. 26), run (invoke) remotely (p. 27),
  change (p. 28), and delete (p. 30) AWS Lambda functions in an AWS account.
- View event logs (p. 31) for and delete (p. 32) AWS CloudFormation stacks in an AWS account.
- Debug code in Amazon ECS (p. 34) clusters in an AWS account. (Debugging code in Amazon ECS
  clusters is currently in beta.)
- Work with Amazon EventBridge (p. 35) schemas in an AWS account.
- Switch AWS credentials to connect with a different set of access permissions within the same AWS
  account or another one (p. 11).
- Switch to working with AWS resources in a different AWS Region for the connected AWS
  account (p. 15).
- Use an HTTP proxy (p. 7) and update it (p. 6) as needed.

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)
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.)
Key tasks for the AWS Toolkit for JetBrains

Use the following brief instructions to complete key tasks with the AWS Toolkit for JetBrains.

- Install the AWS Toolkit for JetBrains (p. 4)
- Update the AWS Toolkit for JetBrains (p. 6)
- Configure the AWS Toolkit for JetBrains to use an HTTP proxy (p. 7)
- Work with connections from the AWS Toolkit for JetBrains to AWS accounts (p. 8)
- Get the current AWS Region that the AWS Toolkit for JetBrains is using (p. 14)
- Switch between AWS Regions (p. 15)
- Open AWS Explorer within the AWS Toolkit for JetBrains (p. 16)
- Work with AWS services
  - Work with AWS serverless applications (p. 17)
  - Work with AWS Lambda functions (p. 24)
  - Work with AWS CloudFormation stacks (p. 30)
  - Work with Amazon CloudWatch Logs (p. 33)
  - Work with Amazon ECS clusters in an account (p. 33)
  - Work with Amazon EventBridge schemas (p. 35)
  - Work with Amazon S3 buckets and objects (p. 35)

Install the AWS Toolkit for JetBrains

Setting up your AWS account to use AWS Toolkit for JetBrains

1. Create an AWS account, if you don’t have an account already.
2. Create an administrator user and group in AWS Identity and Access Management (IAM) in the account, if you haven’t done that already.
   
   **Note**
   
   We recommend that you create or use a special type of user and group in the account for the AWS Toolkit for JetBrains to use, which we call an administrator IAM user and group. Although you can create a regular IAM user and group in the account for the toolkit to use, this approach might not allow the toolkit to have full access to all of the AWS resources and AWS serverless applications in that account. We support, but strongly discourage, using an AWS account root user with the AWS Toolkit for JetBrains.
3. Create an access key for the user, if you don’t have an access key for that user already.
   
   **Note**
   
   An access key contains both an access key ID value and a secret access key value. The AWS Toolkit for JetBrains needs to use both of these values later. Be sure to store them in a secure location. If you lose them, they’re gone forever and can’t be retrieved. However, you can always delete a lost access key, and then create a replacement access key. If you ever do this, you also need to change your toolkit connection settings (p. 12). We support, but strongly discourage, creating an access key for an AWS account root user for the AWS Toolkit for JetBrains to use.
Installing and configuring AWS Toolkit for JetBrains

1. Ensure that a JetBrains IDE supported by AWS Toolkits is installed and running.
2. Open Settings / Preferences.
3. Choose Plugins.
4. On the Marketplace tab, in Search plugins in marketplace, begin entering AWS Toolkit. When AWS Toolkit by Amazon Web Services is displayed, choose it.
5. Choose Install.
Update the AWS Toolkit for JetBrains

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

1. **AWS Command Line Interface (AWS CLI)**
2. **Docker** (Docker must always be running whenever you develop, test, analyze, or deploy serverless applications or functions)
3. **AWS Serverless Application Model Command Line Interface (AWS SAM CLI)**

Before you can use the AWS Toolkit for JetBrains to develop, test, analyze, and deploy AWS serverless applications or Lambda functions, be sure you have the following tools installed. Install the tools in this order:

8. Before you can use the AWS Toolkit for JetBrains to develop, test, analyze, and deploy AWS serverless applications or Lambda functions, be sure you have the following tools installed. Install the tools in this order:

6. When the **Third-party Plugins Privacy Note** is displayed, choose **Accept**.
7. Choose **Restart IDE**, and when prompted, choose **Restart**.

9. Before you can use the AWS Toolkit for JetBrains to debug code in Amazon ECS clusters, you must complete additional steps. For instructions, see the prerequisites (p. 69).

10. After you install the AWS Toolkit for JetBrains (and, if you're working with AWS serverless applications, Lambda functions, or Amazon ECS clusters, you've installed the preceding additional required tools, in order), **connect to an AWS account for the first time (p. 8)**.

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:

**Update the AWS Toolkit for JetBrains**

After you install the AWS Toolkit for JetBrains, you can check for updates to the toolkit at any time and install them.

To do this, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do the following:
Configure the AWS Toolkit for JetBrains to Use an HTTP Proxy

After you install the AWS Toolkit for JetBrains (p. 4), you can configure it to use an HTTP proxy.

With IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

- **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.

After you complete the preceding instructions, the toolkit begins using those HTTP proxy settings.

Top (p. 4)
Work with connections from the AWS Toolkit for JetBrains to AWS accounts

After you install the AWS Toolkit for JetBrains (p. 4), use the toolkit to do the following with AWS accounts:

- Connect to an AWS account for the first time (p. 8)
- Get the current connection (p. 13)
- Add multiple connections (p. 10)
- Switch between connections (p. 11)
- Change connection settings (p. 12)
- Delete a connection (p. 12)

Connect to an AWS Account for the first time

We assume that you already installed the AWS Toolkit for JetBrains (p. 4). To complete this procedure, you need an access key (which contains both an access key ID value and a secret access key value) for a user in IAM (which we recommend), or for an AWS account root user (which we strongly discourage). If you don't have an access key for a user in IAM already, create one.

1. With your access key ID value and secret access key value ready, do one of the following:
   - On the status bar, choose AWS: No credentials selected, and then choose Edit AWS Credential file(s).
   - Open AWS Explorer (p. 16), if it isn't already open. Choose Configure AWS Connection, and then choose Edit AWS Credential file(s).
2. In the file, under [default], for `aws_access_key_id`, replace `[accessKey1]` with your access key ID value (for example, `AKIAIOSFODNN7EXAMPLE`).

   If prompted, choose **I want to edit this file anyway**, and then choose **OK**.

3. For `aws_secret_access_key`, replace `[secretKey1]` with your secret access key value (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`).

The final results should look as shown here, following the named profile format.

```
... Other file contents omitted for brevity ...
[default]
# ... Some comments ...
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
# ... Some more comments ...
# ... Some more comments ...
# ... Some more comments ...
# ... Some more comments ...
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
... Other file contents omitted for brevity ...
```

**Note**

The AWS Toolkit for JetBrains currently supports the following configuration variables:

- `aws_access_key_id`
- `aws_secret_access_key`
- `aws_session_token`
- `credential_process`
- `mfa_serial`
- `role_arn`
- `source_profile`

For more information, see [AWS CLI Configuration Variables](#) in the [AWS CLI Command Reference](#).
Add multiple connections

To complete this procedure, you must first have the additional access key (which contains both an access key ID value and a secret access key value) for a user in IAM (recommended) or AWS account root user (strongly discouraged). If you don’t have an access key for a user IAM already, create one.

1. **Connect for the first time (p. 8)**, if you have not done so already.
2. With the additional access key ID value and secret access key value ready, do one of the following:
   - On the status bar, choose **AWS Connection Settings**, and then choose **All Credentials, Edit AWS Credential file(s)**.
   - **Open AWS Explorer (p. 16)**, if it isn’t already open, and then choose **Show Options Menu** (the settings icon). Choose **AWS Connection Settings, All Credentials, Edit AWS Credential file(s)**.

After connecting, you can use the toolkit to work with AWS resources in that account, such as **AWS serverless** applications, **AWS Lambda** functions, and **AWS CloudFormation** stacks.

You can also have more than one connection (p. 10) available, so that you can switch between them (p. 11).

After you connect, the AWS Toolkit for JetBrains selects the default AWS Region automatically. You might need to switch connections to work with different AWS resources (p. 15).
3. In the file, add a named profile for each additional connection. Profile names can contain only the uppercase letters A through Z, the lowercase letters a through z, the numbers 0 through 9, the hyphen character (-), and the underscore character (_). Profile names must be less than 64 characters in length.

For example, for a named profile named myuser, use the following format.

```
[profile myuser]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

**Note**
The AWS Toolkit for JetBrains currently supports named profiles with only the following characters: A-Z, a-z, 0-9, underscore (_), and hyphen (-).
Currently, the toolkit supports only the following configuration variables:

- aws_access_key_id
- aws_secret_access_key
- aws_session_token
- credential_process
- mfa_serial
- role_arn
- source_profile

For more information, see [AWS CLI Configuration Variables](#) in the [AWS CLI Command Reference](#).

4. Save and then close the file. The AWS Toolkit for JetBrains displays the new connection in the AWS Connection Settings menu in both the status bar and in AWS Explorer.

Now that you have multiple connections, you can switch between them (p. 11).

After you connect, you might need to switch connections to work with (p. 15).

Top (p. 4)

**Switch between connections**

1. Add multiple connections (p. 10), if you haven’t done so already.

2. Do one of the following:
   - On the status bar, choose AWS Connection Settings.
   - Open AWS Explorer (p. 16), if it isn’t already open, and then choose AWS Connection Settings.

3. Choose the named profile to use for the new connection. If it isn’t listed, choose All Credentials, and then choose the named profile to use.

The AWS Toolkit for JetBrains switches to the new connection. This connection is now selected in the AWS Connection Settings menu in both the status bar and AWS Explorer.
After you connect, you might need to switch to working with AWS resources in that account that are in a different AWS Region (p. 15).

Top (p. 4)

**Change connection settings**

1. Do one of the following:
   - On the status bar, choose **AWS Connection Settings, All Credentials, Edit AWS Credential file(s)**.
   - Open **AWS Explorer** (p. 16), if it isn’t already open, and then choose **Show Options Menu** (the settings icon). Then choose **AWS Connection Settings, All Credentials, Edit AWS Credential file(s)**.

2. Make your changes to the file, and then save and close the file.

Top (p. 4)

**Delete a connection**

1. Do one of the following:
   - On the status bar, choose **AWS Connection Settings, All Credentials, Edit AWS Credential file(s)**.
To check which connection the AWS Toolkit for JetBrains is currently using, do one of the following:

- On the status bar, see the current connection displayed in the **AWS Connection Settings** area.

- Open AWS Explorer (p. 16), if it’s not already open, and then choose **Show Options Menu** (the settings icon). Choose **AWS Connection Settings**. The current connection is selected.
You can also have more than one connection (p. 10) available, so that you can switch between them (p. 11).

Top (p. 4)

Get the current AWS Region

To check which AWS Region the AWS Toolkit for JetBrains is currently using, do one of the following:

- On the status bar, see the current Region displayed in the AWS Connection Settings area.

- Open AWS Explorer (p. 16), if it isn't already open, and then choose Show Options Menu (the settings icon). Choose AWS Connection Settings. The current Region is selected.

You can also switch to a different AWS Region (p. 15).

Top (p. 4)
Switch between AWS Regions

To switch AWS Regions, do one of the following:

- On the status bar, choose **AWS Connection Settings**, and then choose the AWS Region that you want to switch to.

- Open AWS Explorer (p. 16), if it isn't already open. Choose **Show Options Menu** (the settings icon), and then choose **AWS Connection Settings**. If the AWS Region that you want to switch to is listed, choose it. Otherwise, choose **All Regions**, and then choose the Region to switch to.
Open AWS Explorer within the AWS Toolkit for JetBrains

To complete this procedure, you must first install the AWS Toolkit (p. 4). Then, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

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

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

After you open **AWS Explorer** for the first time, use it to connect to an AWS account for the first time (p. 8). After that, you can use **AWS Explorer** to work with AWS Lambda (p. 24) functions and AWS CloudFormation (p. 30) stacks in the account.
After you install the AWS Toolkit for JetBrains (p. 4) and then use it to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with AWS serverless applications in an account, as follows:

- Create a serverless application (p. 17)
- Deploy a serverless application (p. 21)
- Change (update) the settings for a serverless application (p. 22)
- Delete a serverless application (p. 23)

Create a serverless application

To complete this procedure, you must first install the AWS Toolkit for JetBrains (p. 4), and if you haven't yet, connect to an AWS account for the first time (p. 8).

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.
Create a serverless application

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

For JetBrains Rider, choose **AWS Serverless Application**.
3. Complete the **New Project dialog box** (or the **New Solution dialog box** for JetBrains Rider) (p. 93), 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** (p. 26) that is contained in that application.

You can also **deploy the serverless application** (p. 21). After you deploy it, you can **run (invoke) the remote version of a Lambda function** (p. 27) that is part of that deployed application.

**Deploy a serverless application**

Before you can use this procedure to deploy an AWS serverless application, you must first **create the AWS serverless application** (p. 17). Then follow these steps.

**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 Serverless Application Model (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](https://github.com/awslabs/serverless-application-model) repository on GitHub.

1. If you need to **switch to a different AWS Region** (p. 15) to deploy the serverless application, do that now.
2. With the **Project** tool window already open and displaying the project that contains the serverless application's files, right-click the project's `template.yaml` file. Then choose **Deploy Serverless Application**.
3. Complete the **Deploy Serverless Application** (p. 84) dialog box, and then choose **Deploy**.

The AWS Toolkit for JetBrains creates a corresponding AWS CloudFormation stack for the deployment. It also adds the name of the stack to the **CloudFormation** list in **AWS Explorer**. If the deployment fails, you can try to figure out why by viewing event logs for the stack (p. 31).

After you deploy it, you can run (invoke) the remote version of an AWS Lambda function (p. 27) that is part of that deployed application.

**Top** (p. 4)

**Change (update) the settings for a serverless application**

Before you can use this procedure to change settings for a serverless application, you must first deploy the AWS serverless application (p. 21) that you want to change. Then follow these steps.

**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`).
Delete a serverless application

Before you can use this procedure to delete a serverless application, you must first deploy the AWS serverless application (p. 21) that you want to delete. Then follow these steps.

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 (p. 15) 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** (p. 84) 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 figure out why by viewing event logs for the stack (p. 31).
1. Open AWS Explorer (p. 16), if it isn't already open. If you need to switch to a different AWS Region (p. 15) 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 figure out why by viewing event logs for the stack (p. 31).

Work with AWS Lambda Functions

After you install the AWS Toolkit for JetBrains (p. 4) and then use the toolkit to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with Lambda functions in the account, as follows.

- Create a function (p. 25)
- Run (invoke) or debug the local version of a function (p. 26)
- Run (invoke) the remote version of a function (p. 27)
- Change (update) the configuration for a function (p. 28)
- Delete a function (p. 30)
Create a function

You can use the AWS Toolkit for JetBrains to create a Lambda function that is part of an AWS serverless application (p. 25), or you can create a Lambda function by itself (p. 25).

Create a serverless application that contains a Lambda Function

See the instructions earlier in this topic about creating an AWS serverless application (p. 17).

Create a standalone function

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 (p. 8).

Then with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

- Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) to create the function in, do that now. Then right-click Lambda, and choose Create new AWS Lambda.

  ![](image1)

  Complete the Create Function (p. 83) 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 figure out why by viewing event logs for the stack (p. 31).

- 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 (p. 15) 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 (p. 83) dialog box, and then choose Create Function.

  ![](image2)

  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 (p. 15) 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 (p. 83) 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 (p. 26) or run (invoke) the remote version (p. 27).

Top (p. 4)

Run (invoke) or debug the local version of a function

A local version of an AWS Lambda function is a function whose source code already exists on your local development computer.

To complete this procedure, you must first create the AWS Lambda function (p. 25) that you want to run (invoke) or debug, if you haven't 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]'.
Run (invoke) the remote version of a function

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 (p. 4) and, if you haven't yet, connect to an AWS account for the first time (p. 8). Then with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider running, do the following.

1. Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) 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 (p. 25) that you want to run (invoke).

   If you created the function as part of an AWS serverless application (p. 25), you must also deploy that application (p. 21).

   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 (p. 83) 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 (p. 85) 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 (p. 28), 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.

Change (update) the configuration for a 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 (p. 98) dialog box, and then choose OK.
- Open AWS Explorer (p. 16), if it isn't already open. If you need to switch to a different AWS Region (p. 15) 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 (p. 105) 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 (p. 98) 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 (p. 104) dialog box, and then choose Update.
Delete a function

You can use the AWS Toolkit for JetBrains to delete an AWS Lambda function that is part of an AWS serverless application (p. 30). Or you can delete a standalone Lambda function (p. 30).

Delete a serverless application that contains a function

See the instructions for deleting a serverless application (p. 23), earlier in this topic.

Delete a standalone function

1. Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) 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.

Work with AWS CloudFormation stacks

After you install the AWS Toolkit for JetBrains (p. 4) and then use the toolkit to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with AWS CloudFormation stacks in the account, as follows:
Create a stack

Currently, you can’t use the AWS Toolkit for JetBrains to create an AWS CloudFormation stack directly. However, whenever you use the toolkit to deploy an AWS serverless application (p. 21) or to create and then deploy an AWS Lambda function (p. 25), the toolkit deploys these by first creating a corresponding stack in AWS CloudFormation, and then using that stack for the deployment.

Change stack settings

Currently, you can’t use the AWS Toolkit for JetBrains to change the settings for an AWS CloudFormation stack directly. However, you can change (update) the settings for an AWS serverless application (p. 22) that belongs to a stack, or change (update) the configuration for an AWS Lambda function (p. 28) that belongs to a stack. Then you deploy that serverless application (p. 21) again or deploy that function, as part of the lifecycle of running (invoking) the remote version of that function (p. 27), again.

View event logs for a stack

1. Open AWS Explorer (p. 16), 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 (p. 15) 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 a stack

1. Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) 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 (p. 31).

Top (p. 4)

**Work with Amazon CloudWatch Logs**

After you install the AWS Toolkit for JetBrains (p. 4) and then use the toolkit to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with Amazon CloudWatch Logs in the account, as follows.

- Viewing CloudWatch log groups and log streams (p. 65)
- Working with CloudWatch log events (p. 66)

Top (p. 4)

**Work with Amazon ECS clusters**

After you install the AWS Toolkit for JetBrains (p. 4) and then use the toolkit to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with Amazon ECS clusters in the account, as follows.
Debug code in a cluster

After you complete the prerequisites (p. 69), do the following.

1. Open AWS Explorer (p. 16), if it isn't already open. If the Amazon ECS cluster (p. 71) is in an AWS Region that's different from the current one, switch to a different AWS Region (p. 15) that contains it.
2. Expand ECS, and then expand Clusters.
3. Expand your Amazon ECS cluster, right-click your service, and then choose Enable Cloud Debugging. For example, in the following screenshot, the cluster is named java, and the service is named java-service.
4. When prompted, choose your Amazon ECS task role (p. 71), and then choose OK.

The status bar displays the message Configuring Cloud Debugging resource. Wait until the Build Output tab of the Build tool window displays a successful configuration message. (A related pop-up also is displayed in the lower-right corner.) This will take several minutes.

Note
As you enable code debugging in your AWS account for the first time, the AWS Toolkit for JetBrains creates an Amazon S3 bucket in your AWS account. The bucket's name follows the format of do-not-delete-cloud-debug-Region-ID-account-ID. The JetBrains Toolkit stores information in this bucket to enable code debugging. Do not delete this bucket or modify its contents. If you do, code debugging might stop working or produce unexpected results. If you accidentally delete or modify this bucket, the JetBrains Toolkit will try to recreate the bucket. You can also force the JetBrains Toolkit to recreate the bucket by choosing Enable Cloud Debugging again as described earlier, or by choosing Disable Cloud Debugging as described later in this procedure.

4. With the code you want to debug displayed, in the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. A service is displayed with a debug icon next to it. This indicates the service is now enabled for cloud debugging. Right-click the service with the debug icon, and then choose Debug.

6. Complete the Edit configuration (p. 89) dialog box, and then choose Debug.

Note
To make changes to this configuration later, on the menu bar, choose Run, Edit Configurations. Then expand Amazon ECS Service Cloud Debug, and choose the service's name.

7. Use the IDE's built-in debugging tools to debug your code as usual.
8. If you make changes to your code, you can start debugging again. In the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. Right-click your service with the debug icon next to it, and then choose Debug.

9. If you make changes to the associated Dockerfile, you must rebuild and republish the Docker image, and then repeat this procedure from the beginning.

10. To disable debugging, in the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. Right-click your service with the debug icon next to it, and then choose Disable Cloud Debugging. A pop-up is displayed, confirming that debugging is disabled.

---

**Working with Amazon EventBridge schemas**

See Working with Amazon EventBridge schemas (p. 75).

---

**Work with Amazon S3 buckets and objects**

After you install the AWS Toolkit for JetBrains (p. 4) and then use the toolkit to connect to an AWS account for the first time (p. 8), you can use the toolkit to work with Amazon S3 buckets and objects in the account, as follows.

- Work with Amazon S3 buckets (p. 77)
- Work with Amazon S3 objects (p. 78)
Installing the AWS Toolkit

Installing the AWS Toolkit for JetBrains is a two-stage process. First, you configure your AWS account and permissions so you can interact with AWS services within the JetBrains IDE. Next, you install and configure the Toolkit as an IDE plugin.

Setting up your AWS account to use AWS Toolkit for JetBrains

1. Create an AWS account, if you don't have an account already.
2. Create an administrator user and group in AWS Identity and Access Management (IAM) in the account, if you haven't done that already.
   
   **Note**
   
   We recommend that you create or use a special type of user and group in the account for the AWS Toolkit for JetBrains to use, which we call an administrator IAM user and group. Although you can create a regular IAM user and group in the account for the toolkit to use, this approach might not allow the toolkit to have full access to all of the AWS resources and AWS serverless applications in that account. We support, but strongly discourage, using an AWS account root user with the AWS Toolkit for JetBrains.

3. Create an access key for the user, if you don't have an access key for that user already.

   **Note**
   
   An access key contains both an access key ID value and a secret access key value. The AWS Toolkit for JetBrains needs to use both of these values later. Be sure to store them in a secure location. If you lose them, they're gone forever and can't be retrieved. However, you can always delete a lost access key, and then create a replacement access key. If you ever do this, you also need to change your toolkit connection settings (p. 12). We support, but strongly discourage, creating an access key for an AWS account root user for the AWS Toolkit for JetBrains to use.

Installing and configuring AWS Toolkit for JetBrains

1. Ensure that a JetBrains IDE supported by AWS Toolkits is installed and running.
2. Open Settings / Preferences.
3. Choose Plugins.
4. On the Marketplace tab, in Search plugins in marketplace, begin entering AWS Toolkit. When AWS Toolkit by Amazon Web Services is displayed, choose it.

5. Choose Install.

**Note**
To use the AWS Toolkit for JetBrains to work with its available features for AWS CloudFormation, AWS Lambda, AWS Serverless, and Amazon ECS, you must install version 1.8 or later of the AWS Toolkit.
6. When the **Third-party Plugins Privacy Note** is displayed, choose **Accept**.
7. Choose **Restart IDE**, and when prompted, choose **Restart**.
8. Before you can use the AWS Toolkit for JetBrains to develop, test, analyze, and deploy AWS serverless applications or Lambda functions, be sure you have the following tools installed. Install the tools in this order:
   1. **AWS Command Line Interface (AWS CLI)**
   2. **Docker** (Docker must always be running whenever you develop, test, analyze, or deploy serverless applications or functions)
   3. **AWS Serverless Application Model Command Line Interface (AWS SAM CLI)**
9. Before you can use the AWS Toolkit for JetBrains to debug code in Amazon ECS clusters, you must complete additional steps. For instructions, see the **prerequisites** (p. 69).
10. After you install the AWS Toolkit for JetBrains (and, if you're working with AWS serverless applications, Lambda functions, or Amazon ECS clusters, you've installed the preceding additional required tools, in order), connect to an AWS account for the first time (p. 8).

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

AWS occasionally releases AWS Toolkit for JetBrains preview and experimental plugins and features through Early Access Program (EAP) builds.

You can automatically get updated EAP builds as they become available, as follows.

1. Make sure a **JetBrains IDE supported by AWS Toolkits** is running.
2. Open **Settings / Preferences**.
3. Choose **Plugins**.
4. Choose **Manage Repositories, Configure Proxy or Install Plugin from Disk** (the settings icon).
5. Choose **Manage Plugin Repositories**.
6. Choose **Add** (the + icon).
7. Enter the following URL to the EAP repository for the AWS Toolkit for JetBrains: **https://plugins.jetbrains.com/plugins/eap/aws.toolkit**. Then press **Enter**, and choose **OK**.
8. If prompted, choose **Restart IDE**. Then when prompted, choose **Restart**.
   - Now, whenever a later EAP build is available, choose **Update** next to the **AWS Toolkit** entry in **Plugins**. When prompted, choose **Restart IDE**. Then choose **Restart**.
9. To remove a reference to the custom repository, in the **Preferences** dialog box, choose **Plugins**. Then choose **Manage Repositories, Configure Proxy or Install Plugin from Disk** (the settings icon), **Manage Plugin Repositories**. Select the URL to the custom repository, and then choose **Remove** (the - icon). Then choose **OK**.

### Updating the AWS Toolkit for JetBrains

After you install the AWS Toolkit for JetBrains (p. 4), you can check for updates to the toolkit at any time and install them.

To do this, with a **JetBrains IDE supported by AWS Toolkits** already running, do the following.

1. Open **Settings / Preferences**.
2. Choose **Updates**. (If no updates are displayed, you might need to choose **Check new updates**.)
3. Follow any on-screen instructions to finish updating the AWS Toolkit for JetBrains.
4. Restart the JetBrains IDE.

Setting AWS credentials for the AWS Toolkit for JetBrains

To access an AWS account by using the AWS Toolkit for JetBrains, you must first connect the toolkit to that account. You do this by specifying which AWS credentials for that account the toolkit will use to make that connection.

Complete the following procedures to make an initial connection, switch between connections, change connection settings, delete connections, and more.

Topics
- Connecting for the first time (p. 39)
- Getting the current connection (p. 41)
- Adding multiple connections (p. 42)
- Switching between connections (p. 44)
- Changing connection settings (p. 44)
- Deleting a connection (p. 45)

Connecting for the first time

You should have already installed the AWS Toolkit for JetBrains (p. 4). To complete this procedure, you must have an access key (which contains both an access key ID value and a secret access key value) for a user in IAM (which we recommend), or an AWS account root user (which we strongly discourage). If you don’t have an access key for a user in IAM, create one.
1. With your access key ID value and secret access key value ready, do one of the following:
   - On the status bar, choose **AWS: No credentials selected**, and then choose **Edit AWS Credential file(s)**.

   ![AWS: No credentials selected](image1)

   ![Edit AWS Credential file(s)](image2)

   ![Edit AWS Credential file(s)](image3)

   - Open AWS Explorer (p. 16), if it isn’t already open. Choose **Configure AWS Connection**, and then choose **Edit AWS Credential file(s)**.

   ![Configure AWS Connection](image4)

   ![Edit AWS Credential file(s)](image5)

   2. In the file, under **[default]**, for **aws_access_key_id**, replace **[accessKey1]** with your access key ID value (for example, **AKIAIOSFODNN7EXAMPLE**).

      If prompted, choose **I want to edit this file anyway**, and then choose **OK**.

   3. For **aws_secret_access_key**, replace **[secretKey1]** with your secret access key value (for example, **wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY**).

      The final results should look as shown here, following the **named profile** format.
Getting the current connection

To check which connection the AWS Toolkit for JetBrains is currently using, do one of the following:

- On the status bar, see the current connection displayed in the AWS Connection Settings area.

- Open AWS Explorer (p. 16), if it's not already open, and then choose Show Options Menu (the settings icon). Choose AWS Connection Settings. The current connection is selected.
You can also have more than one connection (p. 10) available, so that you can switch between them (p. 11).

Adding multiple connections

To complete this procedure, you must first have the additional access key (which contains both an access key ID value and a secret access key value) for a user in IAM (which we recommend) or AWS account root user (which we strongly discourage). If you don’t have an access key for a user IAM already, create one.

1. Connect for the first time (p. 8), if you have not done so already.
2. With the additional access key ID value and secret access key value ready, do one of the following:

   • On the status bar, choose AWS Connection Settings, and then choose All Credentials, Edit AWS Credential file(s).

   • Open AWS Explorer (p. 16), if it isn’t already open, and then choose Show Options Menu (the settings icon). Choose AWS Connection Settings, All Credentials, Edit AWS Credential file(s).
3. In the file, add a named profile for each additional connection. Profile names can contain only the uppercase letters A through Z, the lowercase letters a through z, the numbers 0 through 9, the hyphen character (-), and the underscore character (_). Profile names must be less than 64 characters in length.

For example, for a named profile named myuser, use the following format.

```
[profile myuser]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

**Note**

The AWS Toolkit for JetBrains currently supports named profiles with only the following characters: A-Z, a-z, 0-9, underscore (_), and hyphen (-).

Currently, the toolkit supports only the following configuration variables:

- `aws_access_key_id`
- `aws_secret_access_key`
- `aws_session_token`
- `credential_process`
- `mfa_serial`
- `role_arn`
- `source_profile`

For more information, see [AWS CLI Configuration Variables](#) in the [AWS CLI Command Reference](#).

4. Save and then close the file. The AWS Toolkit for JetBrains displays the new connection in the **AWS Connection Settings** menu in both the status bar and in **AWS Explorer**.

Now that you have multiple connections, you can **switch between them** (p. 11), if you want.
After you connect, you might need to switch to working with AWS resources in that account that are in a different AWS Region (p. 15).

Switching between connections

1. Add multiple connections (p. 10), if you haven’t done so already.
2. Do one of the following:
   - On the status bar, choose AWS Connection Settings.
   - Open AWS Explorer (p. 16), if it isn’t already open, and then choose AWS Connection Settings.
3. Choose the named profile to use for the new connection. If it isn’t listed, choose All Credentials, and then choose the named profile to use.

The AWS Toolkit for JetBrains switches to the new connection. This connection is now selected in the AWS Connection Settings menu in both the status bar and AWS Explorer.

After you connect, you might need to switch to working with AWS resources in that account that are in a different AWS Region (p. 15).

Changing connection settings

1. Do one of the following:
   - On the status bar, choose AWS Connection Settings, All Credentials, Edit AWS Credential file(s).
   - Open AWS Explorer (p. 16), if it isn’t already open, and then choose Show Options Menu (the settings icon). Then choose AWS Connection Settings, All Credentials, Edit AWS Credential file(s).
Deleting a connection

1. Do one of the following:
   - On the status bar, choose **AWS Connection Settings**, **All Credentials**, **Edit AWS Credential file(s)**.
   
   ![AWS Explorer](image)

   - **Open AWS Explorer** (p. 16), if it isn't already open, and then choose **Show Options Menu** (the settings icon). Then choose **AWS Connection Settings**, **All Credentials**, **Edit AWS Credential file(s)**.

2. Make your changes to the file, and then save and close the file.
Setting an AWS Region

When you configure the AWS Toolkit for JetBrains to connect to an AWS account, the toolkit sets the default AWS Region automatically. This topic describes how to get the current AWS Region or to change it.

**Topics**

- Getting the current AWS Region (p. 46)
- Switching AWS Regions (p. 47)

**Getting the current AWS Region**

To check which AWS Region the AWS Toolkit for JetBrains is currently using, do one of the following:

- On the status bar, see the current Region displayed in the **AWS Connection Settings** area.

- Open AWS Explorer (p. 16), if it isn't already open, and then choose **Show Options Menu** (the settings icon). Choose **AWS Connection Settings**. The current Region is selected.
You can also switch to a different AWS Region (p. 15), if you want.

**Switching AWS Regions**

To change the AWS Region Do one of the following.

- On the status bar, choose **AWS Connection Settings**, and then choose the AWS Region that you want to switch to.

- Open AWS Explorer (p. 16), if it isn’t already open. Choose **Show Options Menu** (the settings icon), and then choose **AWS Connection Settings**. If the AWS Region that you want to switch to is listed, choose it. Otherwise, choose **All Regions**, and then choose the Region to switch to.
HTTP proxy setup for the AWS Toolkit for JetBrains

After you install the AWS Toolkit for JetBrains (p. 4), you can configure it to use an HTTP proxy. To do this, you must have a JetBrains IDE supported by AWS Toolkits already running. Then follow the appropriate instructions for the HTTP proxy you want:

- **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.

After you complete the preceding instructions, the AWS Toolkit for JetBrains begins using those HTTP proxy settings.
Working with AWS services by using the AWS Toolkit for JetBrains

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

Topics
- Working with AWS serverless applications by using the AWS Toolkit for JetBrains (p. 49)
- Working with AWS Lambda by using the AWS Toolkit for JetBrains (p. 56)
- Working with AWS CloudFormation by using the AWS Toolkit for JetBrains (p. 62)
- Working with CloudWatch Logs by using the AWS Toolkit for JetBrains (p. 64)
- Working with Amazon Elastic Container Service by Using the AWS Toolkit for JetBrains (p. 68)
- Working with Amazon EventBridge by using the AWS Toolkit for JetBrains (p. 74)
- Working with Amazon S3 by using the AWS Toolkit for JetBrains (p. 77)

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 (p. 49)
- Deploying an AWS serverless application by using the AWS Toolkit for JetBrains (p. 53)
- Changing (updating) AWS Serverless application settings by using the AWS Toolkit for JetBrains (p. 54)
- Deleting an AWS serverless application by using the AWS Toolkit for JetBrains (p. 55)

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

To complete this procedure, you must first install the AWS Toolkit (p. 4) and, if you haven't yet, connect to an AWS account for the first time (p. 8). 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**.
3. Complete the **New Project dialog box** (or the **New Solution dialog box** for JetBrains Rider) (p. 93), 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 (p. 26) that is contained in that application.

You can also deploy the serverless application (p. 21). After you deploy it, you can run (invoke) the remote version of a Lambda function (p. 27) that is part of that deployed application.

**Deploying an AWS serverless application by using the AWS Toolkit for JetBrains**

To complete this procedure, you must first create the AWS serverless application (p. 17) that you want to deploy, if you haven't created 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 Serverless Application Model (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](https://awslabs.github.io/serverless-application-model/reference/guide) in the [awslabs/serverless-application-model](https://github.com/awslabs/serverless-application-model) repository on GitHub.

1. If you need to switch to a different AWS Region (p. 15) to deploy the serverless application, do that now.
2. With the **Project** tool window already open and displaying the project that contains the serverless application's files, right-click the project's `template.yaml` file. Then choose **Deploy Serverless Application**.
3. Complete the Deploy Serverless Application (p. 84) dialog box, and then choose Deploy.

The AWS Toolkit for JetBrains creates a corresponding AWS CloudFormation stack for the deployment. It also adds the name of the stack to the CloudFormation list in AWS Explorer. If the deployment fails, you can try to figure out why by viewing event logs for the stack (p. 31).

After you deploy it, you can run (invoke) the remote version of an AWS Lambda function (p. 27) that is part of that deployed application.

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

You must first deploy the AWS serverless application (p. 21) 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.
Deleting an application

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 (p. 15) 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 (p. 84) 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 figure out why by viewing event logs for the stack (p. 31).

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

Before deleting an AWS serverless application, you must first deploy it (p. 21).

1. Open AWS Explorer (p. 16), if it isn't already open. If you need to switch to a different AWS Region (p. 15) 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 figure out why by viewing event logs for the stack (p. 31).

Working with AWS Lambda by using the AWS Toolkit for JetBrains

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

**Topics**

- Creating an AWS Lambda function by using the AWS Toolkit for JetBrains (p. 57)
- Running (invoking) or debugging the local version of an AWS Lambda function by using the AWS Toolkit for JetBrains (p. 58)
- Running (invoking) the remote version of an AWS Lambda function by using the AWS Toolkit for JetBrains (p. 59)
- Changing (updating) AWS Lambda function settings by using the AWS Toolkit for JetBrains (p. 60)
- Deleting an AWS Lambda function by using the AWS Toolkit for JetBrains (p. 62)
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 (p. 49) instead.

To create a standalone Lambda function, you must first install the AWS Toolkit for JetBrains (p. 4) and, if you haven't yet, connect to an AWS account for the first time (p. 8). Then, with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

- Open AWS Explorer (p. 16), if it isn't already open. If you need to switch to a different AWS Region (p. 15) to create the function in, do that now. Then right-click Lambda, and choose Create new AWS Lambda.

![Create new AWS Lambda]

Complete the Create Function (p. 83) 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 figure out why by viewing event logs for the stack (p. 31).

- 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 (p. 15) 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 (p. 83) dialog box, and then choose Create Function.

![Create new AWS Lambda]

**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 (p. 15) 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 (p. 83) dialog box, and then choose Create Function.
Running (invoking) or debugging a local 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 (p. 26) or run (invoke) the remote version (p. 27).

### 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 (p. 25) 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]'.

   ![Image](app.png)

   - 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]'.

   ![Image](app.png)
Running (invoking) a remote function

2. Complete the Edit configuration (p. 85) 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 (p. 28), 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 (p. 4) AWS Toolkit for JetBrains and, if you haven’t yet, connect to an AWS account for the first time (p. 8). Then with IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider running, do the following.

1. Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) 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 (p. 25) that you want to run (invoke).

   If you created the function as part of an AWS serverless application (p. 25), you must also deploy that application (p. 21).

   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 (p. 83) 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** 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 (p. 28), 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** (p. 16), if it isn't already open. If you need to switch to a different **AWS Region** (p. 15) 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 (p. 98)** 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 (p. 104)** 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 (p. 55) instead.

To delete a standalone Lambda function, do the following.

1. Open AWS Explorer (p. 16), if it isn’t already open. If you need to switch to a different AWS Region (p. 15) 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.

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 (p. 63)
Viewing event logs for an AWS CloudFormation stack by using the AWS Toolkit for JetBrains

1. Open AWS Explorer (p. 16), 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 (p. 15) 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 (p. 16), if it isn't already open. If you need to switch to a different AWS Region (p. 15) 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 (p. 31).

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 (p. 65)
• Working with CloudWatch log events in log streams by using the AWS Toolkit for JetBrains (p. 66)

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 (p. 65)
• Viewing log streams with the Lambda node (p. 65)
• Viewing log streams with the Amazon ECS node (p. 66)

Viewing log groups and log streams with the CloudWatch Logs node

1. Open AWS Explorer (p. 16), 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 (p. 46) 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 (p. 66).

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 (p. 16), 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 (p. 46) 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 (p. 66).

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 (p. 16), 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 (p. 46) 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 (p. 66).

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 (p. 67)
- Working with log actions (p. 67)
- Exporting CloudWatch log events to a file or an editor (p. 68)
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 (p. 65)).

   **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** (p. 67) 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** (p. 67) pane.

2. Choose one of the following log actions:
AWS Toolkit for JetBrains User Guide
Amazon ECS

- **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 (p. 68) 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 (p. 67) 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 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**
- Debugging code in an Amazon Elastic Container Service cluster by using the AWS Toolkit for JetBrains (p. 68)

**Debugging code in an Amazon Elastic Container Service cluster by using the AWS Toolkit for JetBrains**

You can use the AWS Toolkit for JetBrains to debug code in an Amazon Elastic Container Service (Amazon ECS) cluster in an AWS account.

**Note**
Debugging code in Amazon ECS clusters is currently in beta. This feature is intended for use a development environment. Do not use this in a production environment. Debugging code in an Amazon ECS cluster changes the state of resources in your AWS account including, but not limited to, stopping associated Amazon ECS services and changing their configurations. Also, manually changing the state of resources while code debugging is enabled could lead to unpredictable results.
Prerequisites

Before you begin debugging your code, you must have the following:

1. The Docker image that you want to use to debug your code. This image can be hosted in either of the following:
   - Your AWS account's Amazon Elastic Container Registry (Amazon ECR).
   - To find an existing image in Amazon ECR, see Pulling an Image in the Amazon Elastic Container Registry User Guide. See also Using Amazon ECR Images with Amazon ECS in the Amazon Elastic Container Registry User Guide.
   - To create a new image in Amazon ECR, see Pushing an Image in the Amazon Elastic Container Registry User Guide. See also Using Amazon ECR Images with Amazon ECS in the Amazon Elastic Container Registry User Guide.
   - Docker Hub. (Images that are not hosted in Docker Hub—such as microsoft-dotnet-core-runtime—are not supported.)
   - To find an existing image in Docker Hub, see the Explore - Docker Hub on the Docker Hub website.
   - To create a new image in Docker Hub, see the Docker Hub Quickstart on the Docker Documentation website.

   Note
   If you don't already have an image available, we recommend one of the following:
   - For Java, use amazoncorretto for the latest version of Amazon Corretto (a no-cost, multiplatform, production-ready distribution of the Open Java Development Kit (OpenJDK)), or one of the other amazoncorretto images listed on the Docker Hub website that is compatible with the code you want to debug.
   - For Python, use python for the latest version of Python, or one of the other python images listed on the Docker Hub website that is compatible with the code you want to debug.
   - For Node.js, use node for the latest version of Node.js, or one of the other node images listed on the Docker Hub website that is compatible with the code you want to debug.

2. In your AWS account, an AWS Identity and Access Management (IAM) role with AWS permissions that are needed by the code you want to debug. This role will be used as the task role by Amazon Elastic Container Service (Amazon ECS). This task role must also have a trust relationship with the ecs-tasks.amazonaws.com service principal and must contain a reference to the AmazonSSMManagedInstanceCore AWS managed policy. For more information, see how to set up the Amazon ECS task role (p. 71).

3. In your AWS account, an Amazon ECS cluster that contains the service you want to debug. For more information, see how to set up the Amazon ECS cluster (p. 71).

4. In your AWS account, a specific IAM customer managed policy that you add to the appropriate IAM entity (such as an IAM user, group, or role) that is associated with AWS credentials you specify when connecting to the AWS Toolkit for JetBrains. For more information, see how to add the IAM customer managed policy to the IAM entity (p. 72).

5. On your local development machine, a copy of the code you want to debug.

Debugging code

After you complete the preceding prerequisites (p. 69), you can debug your code as follows:
1. Open AWS Explorer (p. 16), if it isn’t already open. If the Amazon ECS cluster (p. 71) is in an AWS Region that’s different from the current one, switch to a different AWS Region (p. 15) that contains it.

2. Expand ECS, and then expand Clusters.

3. Expand your Amazon ECS cluster, right-click your service, and then choose Enable Cloud Debugging. For example, in the following screenshot, the cluster is named java, and the service is named java-service.

4. When prompted, choose your Amazon ECS task role (p. 71), and then choose OK.

   The status bar displays the message Configuring Cloud Debugging resource. Wait until the Build Output tab of the Build tool window displays a successful configuration message. (A related pop-up also is displayed in the lower-right corner.) This will take several minutes.

   **Note**
   As you enable code debugging in your AWS account for the first time, the AWS Toolkit for JetBrains creates an Amazon S3 bucket in your AWS account. The bucket’s name follows the format of do-not-delete-cloud-debug-Region-ID-account-ID. The JetBrains Toolkit stores information in this bucket to enable code debugging. **Do not delete this bucket or modify its contents.** If you do, code debugging might stop working or produce unexpected results. If you accidentally delete or modify this bucket, the JetBrains Toolkit will try to recreate the bucket. You can also force the JetBrains Toolkit to recreate the bucket by choosing **Enable Cloud Debugging** again as described earlier, or by choosing **Disable Cloud Debugging** as described later in this procedure.

5. With the code you want to debug displayed, in the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. A service is displayed with a debug icon next to it. This indicates the service is now enabled for cloud debugging. Right-click the service with the debug icon, and then choose Debug.

6. Complete the **Edit configuration** (p. 89) dialog box, and then choose Debug.

   **Note**
   To make changes to this configuration later, on the menu bar, choose Run, Edit Configurations. Then expand Amazon ECS Service Cloud Debug, and choose the service’s name.

7. Use the IDE’s built-in debugging tools to debug your code as usual.

8. If you make changes to your code, you can start debugging again. In the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. Right-click your service with the debug icon next to it, and then choose Debug.
9. If you make changes to the associated Dockerfile, you must rebuild and republish the Docker image, and then repeat this procedure from the beginning.

10. To disable debugging, in the AWS Explorer, expand ECS, expand Clusters, and then expand your cluster. Right-click your service with the debug icon next to it, and then choose Disable Cloud Debugging. A pop-up is displayed, confirming that debugging is disabled.

### Setting up the Amazon ECS task role

Note that the following information applies to permissions that Amazon ECS needs, which is different from permissions that the AWS Toolkit for JetBrains needs (p. 72).

To debug code in Amazon Elastic Container Service (Amazon ECS) clusters, you must first have in your AWS account an AWS Identity and Access Management (IAM) role with AWS permissions that are needed by the code you want to debug. This role will be used as the task role by Amazon Elastic Container Service (Amazon ECS). This task role must also have a trust relationship with the ecs-tasks.amazonaws.com service principal and must contain a reference to the AmazonSSMManagedInstanceCore AWS managed policy.

To create a role that meets these requirements, see Creating a Role for an AWS Service (Console) in the IAM User Guide, specifying the following settings:

1. For Choose the service that will use this role, choose Elastic Container Service.
2. For Select your use case, choose Elastic Container Service Task.
3. For Attach permissions policies, choose AmazonSSMManagedInstanceCore.

To add additional AWS permissions to an existing Amazon ECS task role, see “To change the permissions allowed by a role (console)” in Modifying a Role (Console) in the IAM User Guide.

### Setting up the Amazon ECS cluster

To debug code in Amazon Elastic Container Service (Amazon ECS) clusters, you must first have in your AWS account an Amazon ECS cluster that contains the service you want to debug.

### Setting up a Fargate Cluster

To quickly create a Fargate cluster, service, and task definition that meets the minimum requirements, see Getting Started with Amazon ECS using Fargate in the Amazon Elastic Container Service User Guide for AWS Fargate. The only required settings are in Step 1: Container and Task. Specifically, after you specify a name for the container, for Container definition, choose Configure. Then specify an Image that is compatible with the code you want to debug.

### Setting up an Amazon EC2 Cluster

For information on how to create an Amazon EC2 managed cluster, see Getting Started with Amazon ECS Using Amazon EC2 in the Amazon Elastic Container Service Developer Guide

**Note**

If you don't already have an image available, we recommend one of the following:

- For Java, use `amazoncorretto` for the latest version of Amazon Corretto, or one of the other `amazoncorretto` images listed on the Docker Hub website that is compatible with the code you want to debug.
- For Python, use `python` for the latest version of Python, or one of the other `python` images listed on the Docker Hub website that is compatible with the code you want to debug.
- For Node.js, use `node` for the latest version of Node.js, or one of the other `node` images listed on the Docker Hub website that is compatible with the code you want to debug.
For advanced scenarios, you can create a cluster, task definition, and service independently. To do so, see the following in the Amazon Elastic Container Service Developer Guide:

- **Creating a Cluster** – For **Select cluster template**, you can choose either **Networking only (for Fargate)** or **EC2 Linux + Networking (for EC2)**.
- **Creating a Task Definition** – For **Select launch type compatibility**, choose the corresponding launch type.
- **Creating a Service** – For **Configure service**, choose the corresponding **Launch Type**.

### Adding the IAM customer managed policy

Note that the following information applies to permissions that the AWS Toolkit for JetBrains needs, which is different from permissions that Amazon ECS needs (p. 71).

When setting up to debug code in Amazon ECS clusters, we strongly recommend that you follow the AWS security best practice of granting least privilege. Granting least privilege means granting only the permissions required to perform a task. To grant least privilege for debugging code in Amazon ECS clusters, you must attach a specific IAM customer managed policy as follows to an IAM entity (such as an IAM user, group, or role). This IAM entity must be associated with the credentials that you specify when you connect to the AWS Toolkit for JetBrains (p. 39).

In the following policy statement, permission is granted to two Amazon ECS services named `MyService` and `MyOtherService` as well as to two Amazon ECS task roles named `MyTaskRole` and `MyOtherTaskRole` and two Amazon ECS execution task roles named `MyExecutionTaskRole` and `MyOtherExecutionTaskRole`. Change the names of these example services and task roles to match your own, and then attach this policy to the appropriate IAM entity.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowedECSServices",
      "Effect": "Allow",
      "Action": [
        "ecs:UpdateService"
      ],
      "Resource": [
        "arn:aws:ecs:*:*:service/*/cloud-debug-*",
        "arn:aws:ecs:*:*:service/*/MyService",
        "arn:aws:ecs:*:*:service/*/MyOtherService"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:ListRoles",
        "iam:SimulatePrincipalPolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowedIAMRoles",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/MyTaskExecutionRole",
        "arn:aws:iam::*:role/MyOtherTaskExecutionRole",
        "arn:aws:iam::*:role/MyTaskRole",
```

72
"arn:aws:iam::*:role/MyOtherRole"
],
"Condition": {
  "StringEquals": {
    "iam:PassedToService": "ecs-tasks.amazonaws.com"
  }
},
{
  "Effect": "Allow",
  "Action": [
    "iam:PassRole"
  ],
  "Resource": [
    "arn:aws:iam::*:role/aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "s3:CreateBucket",
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::do-not-delete-cloud-debug-*"
},
{
  "Effect": "Allow",
  "Action": [
    "ecs:ListClusters",
    "ecs:ListServices",
    "ecs:DescribeServices",
    "ecs:ListTasks",
    "ecs:DescribeTasks",
    "ecs:DescribeTaskDefinition",
    "elasticloadbalancing:DescribeLoadBalancers",
    "elasticloadbalancing:DescribeLoadBalancerAttributes",
    "elasticloadbalancing:DescribeTargetGroups",
    "ecr:GetAuthorizationToken",
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream"
  ],
  "Resource": [
    "arn:aws:logs::*:cloud-debug*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "ecs:CreateService",
    "ecs:DeleteService"
  ],
  "Resource": "arn:aws:ecs::*:service/*/cloud-debug*"
},
You can use tools such as the IAM console within the AWS Management Console to create an IAM customer managed policy and then add the policy to the appropriate IAM entity (such as an IAM user, group, or role).

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

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 (p. 36).

Contents
- View an available schema (p. 75)
- Find an available schema (p. 75)
- Generate code for an available schema (p. 75)
- Create an AWS Serverless Application Model application that uses an available schema (p. 76)

View an available schema

1. With the AWS Explorer (p. 80) 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 (p. 80) 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 (p. 80) 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** (p. 53)
- **Change (update) the application’s settings** (p. 54)
- **Delete the deployed application** (p. 55)

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** (p. 58)
- **Run (invoke) the remote version of a function** (p. 59)
- **Change a function’s settings** (p. 60)
- **Delete a function** (p. 62)
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 (p. 77)
- Working with Amazon S3 objects by using the AWS Toolkit for JetBrains (p. 78)

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 (p. 77)
- Viewing Amazon S3 buckets (p. 77)
- Deleting an Amazon S3 bucket (p. 78)

Creating an Amazon S3 bucket

1. Open AWS Explorer (p. 16), 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 Developer Guide.

4. Choose Create.

Viewing Amazon S3 buckets

1. Open AWS Explorer (p. 16), 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 (p. 46) are displayed beneath the Amazon S3 node.

Deleting an Amazon S3 bucket

1. Open AWS Explorer (p. 16), 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 (p. 78)
• Opening an object in the IDE (p. 79)
• Uploading an object (p. 79)
• Downloading an object (p. 79)
• Deleting an object (p. 79)

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 (p. 16), 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 (p. 78)).
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 (p. 78)).
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 (p. 78)).
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 (p. 78)).
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.
User interface reference for the AWS Toolkit for JetBrains

Use the following information to help you work with the AWS Toolkit for JetBrains user interface.

Topics

• AWS Explorer (p. 80)
• Create Function dialog box (p. 83)
• Deploy Serverless Application dialog box (p. 84)
• Edit Configuration dialog box (p. 85)
• New Project dialog box (p. 93)
• Run/Debug Configurations dialog box (p. 98)
• Update Code dialog box (p. 104)
• Update Configuration dialog box (p. 105)

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 already installed (p. 4) and IntelliJ IDEA, PyCharm, WebStorm, or JetBrains Rider already running, do one of the following:

• On the tool window bar, choose AWS Explorer.
On the main menu, choose **View, Tool Windows, AWS Explorer**.

**AWS Explorer** contains the following items.

The **Show Options Menu** menu (the settings icon) in **AWS Explorer** contains the following commands.

**AWS Connection Settings**

- **AWS Regions list** – The AWS Toolkit for JetBrains uses the selected AWS Region. To switch the toolkit to use a different AWS 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 switch the toolkit to use a different recent connection instead, 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 switch the toolkit to use
a different connection instead, choose that connection's name. You can also do other connection tasks (p. 8) by choosing AWS Edit Credential file(s).

View Documentation

Goes to this AWS Toolkit for JetBrains User Guide.

View Source on GitHub

Goes to the 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 Windows Viewing Modes on the IntelliJ IDEA Help website.

For PyCharm view modes, see Tool Windows Viewing Modes on the PyCharm Help website.

For WebStorm view modes, see Tool Windows Viewing Modes on the WebStorm Help website.

For JetBrains Rider view modes, see Tool Windows Viewing 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 windows 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 AWS Lambda (p. 24) functions and work with AWS CloudFormation (p. 30) stacks in AWS accounts.

Note

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

Clicking this area enables you to complete some of the tasks that you can also do with AWS Explorer, as follows.

AWS Regions list

The AWS Toolkit for JetBrains uses the selected AWS Region. Choose another listed Region to switch the Region that the toolkit uses.

Recent Credentials list

Lists recent connections made from the AWS Toolkit for JetBrains to AWS accounts. The toolkit uses the selected connection. To switch the recent connection that the toolkit uses, 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 switch the connection that the toolkit uses, choose that connection’s name. You can also do other connection tasks (p. 8) by choosing AWS Edit Credential file(s).

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 (p. 25).

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, the hyphen character (-), and the underscore character (_). Name must be less than 64 characters in length.

Description

Optional. Any meaningful description about the function.

Handler

Required. The identifier of the corresponding function handler for Java, Python, Node.js, or C#.

Runtime

Required. The identifier of the runtime for Lambda to use.

Timeout (seconds)

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

Memory (MB)

Required. The amount of memory available to the function during its execution. Specify an amount between 128 MB and 3,008 MB in 64-MB increments.
Environment Variables

Optional. Any environment variables for the 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 one instead, choose Create, and then follow the on-screen instructions. For more information, see AWS Lambda Execution Role in the AWS Lambda Developer Guide.

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 the Lambda service 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 the performance of your function. For more information, see Using AWS X-Ray in the AWS Lambda Developer Guide.

Source Bucket

Required. Choose an available Amazon S3 bucket in the connected AWS account for the AWS Serverless Application Model Command Line Interface (AWS SAM CLI) to use to deploy the function to Lambda. To create an Amazon Simple Storage Service (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 in the AWS Toolkit for JetBrains is displayed when you deploy an AWS serverless application (p. 21).

The Deploy Serverless Application dialog box contains the following items.
Create Stack

Either Create Stack or Update Stack is required (but not both). Provide the name of the stack for the AWS Serverless Application Model Command Line Interface (AWS SAM 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

Either Create Stack or Update Stack is required (but not both). Choose the name of an existing AWS CloudFormation stack for the AWS SAM CLI to use in the connected AWS account to deploy the AWS serverless application.

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 one instead, choose Create, and then follow the on-screen instructions.

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 won'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 an AWS 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 with Dependencies in the AWS Serverless Application Model Developer Guide.

Edit Configuration dialog box

The Edit configuration dialog box in the AWS Toolkit for JetBrains is displayed when you change (update) the configuration for an AWS Lambda function (p. 28) or debug code in an Amazon ECS cluster (p. 34).

- Edit Configuration Dialog Box (Lambda Function) (p. 85)
- Edit Configuration Dialog Box (Amazon ECS Cluster) (p. 89)

Edit Configuration dialog box (Lambda function)

The Edit configuration dialog box contains two tabs: Configuration and SAM CLI.
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.  

**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.  

**From template**

*Required.* 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.

**Runtime**

*Required.* The identifier of the runtime for Lambda to use.

**Handler**

*Required.* The identifier of the corresponding function handler for Java, Python, or Node.js.
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.

Credentials

Required. The name of the existing AWS account connection (p. 8) to use.

Region

Required. The name of the AWS Region (p. 15) to use for the connected account.

File

Either File or Text is required (but not both). The location and file name of the event data to pass into the function, expressed 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

Either File or Text is required (but not both). The event data to pass into the function, expression 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.

Before launch: Activate tool window

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

Show this page

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

Activate tool window

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

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 **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.  

### 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.

### Build function inside a container

*Optional.* If selected, the AWS SAM CLI builds any of the serverless application's functions inside of an AWS 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 with Dependencies](#) 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 that Lambda Docker containers should connect to, and the default bridge network. If not specified, the Lambda containers will connect to only the default bridge Docker network.

Before launch: Activate tool window

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

Show this page

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

Activate tool window

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

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.

**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.

**Cluster**

*Required.* The name of the 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 should not 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. All code and artifacts that are to be run must be mapped. 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 allows 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.

**Show this page**

*Optional.* If selected, displays these configuration settings before starting this configuration.

**Activate tool window**

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

**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 (p. 8)](p. 8) to use.

**Region**

*Required.* The name of the [AWS Region (p. 15)](p. 15) 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
New Project dialog box

The New Project dialog box in the AWS Toolkit for JetBrains is displayed when you create an AWS serverless application (p. 17).

Topics

- New Project dialog box (IntelliJ IDEA) (p. 94)
- New Project dialog box (PyCharm) (p. 95)
- New Project dialog box (WebStorm) (p. 96)
- New Project dialog box (JetBrains Rider) (p. 96)
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 will create the project.

**Runtime**

*Required.* The identifier of the runtime for AWS Lambda to use.

**SAM Template**

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

**Project SDK**

*Required.* The Java SDK (JDK) to use. For more information, see Configure the JDK when creating a project on the IntelliJ IDEA Help website.

**Module name**

*Required.* The name of the module to create. For more information, see Modules on the IntelliJ IDEA Help website.

**Content root**

*Required.* The location where IntelliJ IDEA will create the project's content root. For more information, see Content roots on the IntelliJ IDEA Help website.

**Module file location**

*Required.* The location where IntelliJ IDEA will create the module. For more information, see Modules on the IntelliJ IDEA Help website.
Project format

Required. The format of the project that IntelliJ IDEA will create. For more information, see Project formats on the IntelliJ IDEA Help website.

New Project dialog box (PyCharm)

The New Project dialog box contains the following items.

Location

Required. The location where PyCharm will create the project. For more information, see Project on the PyCharm Help website.

SAM Template

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

New environment using / Existing interpreter

Either New environment using or Existing interpreter is required (but not both). Provides information about the interpreter that PyCharm will use when creating the project. For more information, see Configure a Python interpreter on the PyCharm Help website.
New Project dialog box (WebStorm)

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

**Location**

*Required.* The location where WebStorm will create the project.

**Runtime**

*Required.* The identifier of the runtime for AWS Lambda to use.

**SAM Template**

*Required.* The name of the AWS SAM template to use.

**Node interpreter**

*Required.* The location where the Node.js interpreter is installed.

New Project dialog box (JetBrains Rider)

**Note**

When you create a new solution, this dialog box will contain 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.

**Runtime**

*Required.* The Lambda runtime to be used.

**SAM CLI executable**

*Required.* The location where the SAM CLI is installed.

**SAM Template**

*Required.* The name of the SAM template to use to create the project.

**Resulting project structure**

*Required.* The paths for the project's directories and files that will be created.
Run/Debug Configurations dialog box

The Run/Debug Configurations dialog box in the AWS Toolkit for JetBrains is displayed whenever you change (update) the configuration for an AWS Lambda function (p. 28).

This dialog box contains different configuration settings for the Lambda function, depending on whether you are changing (updating) settings for the local version of the function (the function's source code is on the local computer) or the remote version of that same function (the function's source code is within the Lambda service for the account).

Topics
- Run/Debug Configurations dialog box (local function settings) (p. 98)
- Run/Debug Configurations dialog box (remote function settings) (p. 102)

Run/Debug Configurations dialog box (local function settings)

This dialog box is displayed whenever you change (update) settings for the local version of an AWS Lambda function (the function's source code is on the local computer).

Note
To change (update) settings for the remote version of that same function (the function's source code is within the Lambda service for the account), see Run/Debug Configurations dialog box (remote function settings) (p. 102) instead.

This dialog box contains two tabs: Configuration and SAM CLI.
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.

**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

**From template**

*Required.* 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.

**Runtime**

*Required.* The identifier of the runtime for Lambda to use.
Handler

*Required.* The identifier of the corresponding function handler for Java, Python, Node.js, or Node.js.

Environment Variables

*Optional.* Any *environment variables* for the AWS 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.

Credentials

*Required.* The name of the existing AWS account connection (p. 8) to use.

Region

*Required.* The name of the AWS Region (p. 15) to use for the connected account.

File

Either File or Text is *required* (but not both). The location and file name of the event data to pass into the function, expressed 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

Either File or Text is *required* (but not both). The event data to pass into the function, expression 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*.

Before launch: Activate tool window

*Optional.* Lists any tasks that must be performed before starting this configuration.  

Show this page

*Optional.* If selected, displays these configuration settings before starting this configuration.

Activate tool window

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

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.

**Share / Share through VCS**

*Optional.* If selected, makes this configuration available to other team members.

**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.

**Build function inside a container**

*Optional.* If selected, the AWS SAM CLI builds any of the serverless application’s functions inside of an AWS 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 with Dependencies 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 that Lambda Docker containers should connect to, with the default bridge network. If not specified, the Lambda containers connect only to the default bridge Docker network.

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.

Run/Debug Configurations dialog box (remote function settings)

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

Note

To change (update) settings for the local version of that same function (the function's source code is on the local computer), see Run/Debug Configurations dialog box (local function settings) (p. 98) instead.
Although the name of the dialog box is Run/Debug Configurations, you cannot use the AWS Toolkit to debug the remote version of a Lambda function. You can only run it.
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** (p. 8) to use.

**Region**

*Required.* The name of the **AWS Region** (p. 15) to use for the connected account.

**Function**

*Required.* The name of the AWS Lambda function to use.
File

Either **File** or **Text** is *required* (but not both). The location and file name of the event data to pass into the function, expressed in JSON format. For event data examples, see [Invoke the Lambda Function](https://docs.aws.amazon.com/lambda/developer-guide humili) in the *AWS Lambda Developer Guide* and [Generating Sample Event Payloads](https://docs.aws.amazon.com/lambda/developer-guide humili) in the *AWS Serverless Application Model Developer Guide*.

Text

Either **File** or **Text** is *required* (but not both). The event data to pass into the function, expression in JSON format. For event data examples, see [Invoke the Lambda Function](https://docs.aws.amazon.com/lambda/developer-guide humili) in the *AWS Lambda Developer Guide* and [Generating Sample Event Payloads](https://docs.aws.amazon.com/lambda/developer-guide humili) in the *AWS Serverless Application Model Developer Guide*.

**Before launch: Activate tool window**

*Optional*. Lists any tasks that must be performed before starting this configuration.

**Show this page**

*Optional*. If selected, displays these configuration settings prior to starting this configuration.

**Activate tool window**

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

**Notes**

1 For more information, see the following:
   - For IntelliJ IDEA, see [Common options](https://www.jetbrains.com/help/idea) on the IntelliJ IDEA Help website.
   - For PyCharm, see [Common options](https://www.jetbrains.com/help/pycharm) on the PyCharm Help website.
   - For WebStorm, see [Common options](https://www.jetbrains.com/help/webstorm) on the WebStorm Help website.
   - For JetBrains Rider, see [Common options](https://www.jetbrains.com/help/rider) on the JetBrains Rider Help website.

2 For more information, see the following:
   - For IntelliJ IDEA, see [Before Launch options](https://www.jetbrains.com/help/idea) on the IntelliJ IDEA Help website.
   - For PyCharm, see [Before Launch options](https://www.jetbrains.com/help/pycharm) on the PyCharm Help website.
   - For WebStorm, see [Before Launch options](https://www.jetbrains.com/help/webstorm) on the WebStorm Help website.
   - For JetBrains Rider, see [Before Launch options](https://www.jetbrains.com/help/rider) 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** (p. 21).

The **Update Code** dialog box contains the following items.
Handler

Required. The identifier of the corresponding function handler for Java, Python, Node.js, or C#.

Source Bucket

Required. Choose an existing Amazon S3 bucket in the connected AWS account for the AWS Serverless Application Model Command Line Interface (AWS SAM 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.

Update Configuration dialog box

The Update Configuration dialog box in the AWS Toolkit for JetBrains is displayed whenever you change (update) the configuration for an AWS Lambda function (p. 28).

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, the hyphen character (-), and the underscore character (_). This name must be less than 64 characters in length.

Description

Optional. Any meaningful description about the function.

Handler

Required. The identifier of the corresponding Java, Python, Node.js, or C#.

Runtime

Required. The identifier of the runtime for Lambda to use.

Timeout (seconds)

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

Required. The amount of memory available to the function during its execution. Specify an amount between 128 MB and 3,008 MB in 64-MB increments.

Environment Variables

Optional. Any environment variables for the 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 one instead, choose Create, and then follow the on-screen instructions. For more information, see AWS Lambda Execution Role in the AWS Lambda Developer Guide.

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 the Lambda service 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 the performance of your function. For more information, see Using AWS X-Ray in the AWS Lambda Developer Guide.
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 this AWS Product or Service (p. 107)
• Identity and Access Management for this AWS Product or Service (p. 108)
• Compliance Validation for this AWS Product or Service (p. 108)
• Resilience for this AWS Product or Service (p. 109)
• Infrastructure Security for this AWS Product or Service (p. 109)

Data protection in this AWS Product or Service

This AWS product or service conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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.
• 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 personal data that is stored in Amazon S3.
We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with this AWS product or service or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into this AWS product or service or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog](https://aws.amazon.com/security/gdpr/).

### Identity and Access Management for this AWS Product or Service

AWS Identity and Access Management (IAM) is an Amazon Web Services (AWS) service that helps an administrator securely control access to AWS resources. IAM administrators control who can be **authenticated** (signed in) and **authorized** (have permissions) to use resources in AWS services. IAM is an AWS service that you can use with no additional charge.

To use this AWS product or service to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an IAM user to provide access credentials instead of using your AWS account credentials.

For details about working with IAM, see [AWS Identity and Access Management](https://docs.aws.amazon.com/iam/userguide/.)

For an overview of IAM users and why they are important for the security of your account, see [AWS Security Credentials in the Amazon Web Services General Reference](https://docs.aws.amazon.com/.)

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](https://aws.amazon.com/security/) and [AWS services that are in scope of AWS compliance efforts by compliance program](https://aws.amazon.com/compliance/).

### Compliance Validation for this AWS Product or Service

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](https://aws.amazon.com/security/) and [AWS services that are in scope of AWS compliance efforts by compliance program](https://aws.amazon.com/compliance/).

The security and compliance of AWS services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. AWS provides a frequently updated list of AWS services in scope of specific compliance programs at [AWS Services in Scope by Compliance Program](https://aws.amazon.com/compliance/).

Third-party audit reports are available for you to download using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](https://aws.amazon.com/artifact/).

For more information about AWS compliance programs, see [AWS Compliance Programs](https://aws.amazon.com/compliance/).

Your compliance responsibility when using this AWS product or service to access an AWS service is determined by the sensitivity of your data, your organization's compliance objectives, and applicable
laws and regulations. If your use of an AWS service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- **Security and Compliance Quick Start Guides** – Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – A whitepaper that describes how companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – A collection of workbooks and guides that might apply to your industry and location.
- **AWS Config** – A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

### Resilience for this AWS Product or Service

The Amazon Web Services (AWS) global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

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 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.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Toolkits now available for four more JetBrains IDEs (p. 110)</td>
<td>AWS Toolkits are now available as plug-ins for four additional JetBrains IDEs:</td>
<td>May 28, 2020</td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS Toolkit for CLion</strong> (for C &amp; C++ development)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS Toolkit for GoLand</strong> (for Go development)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS Toolkit for PhpStorm</strong> (for PHP development)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS Toolkit for RubyMine</strong> (for Ruby development)</td>
<td></td>
</tr>
<tr>
<td>Working with Amazon CloudWatch Logs now available (p. 110)</td>
<td>Using the AWS Toolkit to work with Amazon CloudWatch Logs is now available.</td>
<td>April 15, 2020</td>
</tr>
<tr>
<td>Working with Amazon S3 buckets and objects now available (p. 110)</td>
<td>Using the AWS Toolkit to work with Amazon S3 buckets and objects is now available.</td>
<td>March 27, 2020</td>
</tr>
<tr>
<td>Working with Amazon EventBridge Schemas now available (p. 110)</td>
<td>Using the AWS Toolkit to work with EventBridge Schemas is now available.</td>
<td>December 2, 2019</td>
</tr>
<tr>
<td>Debugging code in Amazon ECS clusters now available in beta (p. 110)</td>
<td>Using the AWS Toolkit to debug code in Amazon ECS clusters is now available in beta.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>AWS Toolkit for Rider now available (p. 110)</td>
<td>The AWS Toolkit for Rider is now available.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>AWS Toolkit for WebStorm now available (p. 110)</td>
<td>The AWS Toolkit for WebStorm is now available.</td>
<td>October 23, 2019</td>
</tr>
<tr>
<td>AWS Toolkit for IntelliJ now generally available (p. 110)</td>
<td>The AWS Toolkit for IntelliJ is now generally available. The corresponding documentation has been refreshed accordingly.</td>
<td>March 27, 2019</td>
</tr>
<tr>
<td>Initial release (p. 110)</td>
<td>This is the initial release of the <em>AWS Toolkit for JetBrains User Guide</em>. The AWS Toolkit for PyCharm is now generally available. The AWS Toolkit for IntelliJ is still in Developer Preview.</td>
<td>November 27, 2018</td>
</tr>
</tbody>
</table>