# Table of Contents

AWS Toolkit for Visual Studio Code ............................................................................................................................. 1
  What is the AWS Toolkit for Visual Studio Code ............................................................................................................... 1
  Related information ......................................................................................................................................................... 1

Setting Up ........................................................................................................................................................................ 2
  Installing the Toolkit for VS Code .................................................................................................................................. 2
    Prerequisites ................................................................................................................................................................. 2
    Install the Toolkit for VS Code ................................................................................................................................... 3
  Establishing credentials .................................................................................................................................................. 3
    Using AWS-supported credentials ............................................................................................................................... 4
    Using an external credential process .......................................................................................................................... 7

Connecting to AWS .......................................................................................................................................................... 8
  Connect to AWS through the Toolkit for VS Code ......................................................................................................... 8
  Use multiple AWS accounts with the Toolkit .............................................................................................................. 10

Changing AWS Regions .................................................................................................................................................... 10
  Add a Region to the AWS Explorer ............................................................................................................................. 11
  Hide a Region from the AWS Explorer ......................................................................................................................... 12

Configuring your toolchain ........................................................................................................................................... 13
  Configure a toolchain for .NET Core ........................................................................................................................... 13
  Configure a toolchain for Node.js ............................................................................................................................... 13
  Configure a toolchain for Python ................................................................................................................................ 14

Using Your toolchain ..................................................................................................................................................... 14

Navigating the Toolkit for VS Code ................................................................................................................................... 15
  Fundamental UI Components ...................................................................................................................................... 15
    The AWS Explorer ....................................................................................................................................................... 15
    The AWS CDK Explorer .............................................................................................................................................. 16

Command Locations ....................................................................................................................................................... 16
  Visual Studio Code Command Palette ....................................................................................................................... 16
    AWS Explorer Menu ................................................................................................................................................. 18

Working with AWS Services ........................................................................................................................................... 19
  AWS Explorer ............................................................................................................................................................... 19
    Working with AWS Serverless Application .................................................................................................................. 21
    AWS CloudFormation stacks ....................................................................................................................................... 30
    AWS Lambda Functions .............................................................................................................................................. 31
    Amazon S3 ................................................................................................................................................................. 34
    Amazon EventBridge .................................................................................................................................................. 37
    AWS Step Functions ................................................................................................................................................... 39

Amazon ECS .................................................................................................................................................................... 46
  Using IntelliSense for task-definition files .................................................................................................................. 46

AWS CDK Explorer .......................................................................................................................................................... 47
  AWS CDK applications .................................................................................................................................................. 47

Security ........................................................................................................................................................................... 51
  Data protection ............................................................................................................................................................... 51
  Identity and Access Management .................................................................................................................................. 52
  Logging and Monitoring ................................................................................................................................................ 52
  Compliance Validation .................................................................................................................................................... 53
  Resilience ......................................................................................................................................................................... 53
  Infrastructure Security .................................................................................................................................................... 54
  Configuration and vulnerability analysis ....................................................................................................................... 54

Document history ............................................................................................................................................................ 55
AWS Toolkit for Visual Studio Code

This is the user guide for the AWS Toolkit for VS Code. If you are looking for the AWS Toolkit for Visual Studio, see the User Guide for the AWS Toolkit for Visual Studio.

What is the AWS Toolkit for Visual Studio Code

The Toolkit for VS Code is an open-source extension for the Visual Studio Code (VS Code) editor. This extension makes it easier for developers to develop, debug locally, and deploy serverless applications that use Amazon Web Services (AWS).

Topics

- Setting Up the AWS Toolkit for Visual Studio Code (p. 2)
- Navigating the AWS Toolkit for Visual Studio Code (p. 15)
- Working with AWS Services (p. 19)

Related information

Use the following resources to access the source code for the toolkit or view currently open issues.

- Source Code
- Issue Tracker

To learn more about the Visual Studio Code editor, visit https://code.visualstudio.com/.
Setting Up the AWS Toolkit for Visual Studio Code

This section helps you set up the AWS Toolkit for Visual Studio Code. It provides information about how to install and configure the toolkit, set up your credentials, and connect to AWS.

Topics
- Installing the AWS Toolkit for Visual Studio Code (p. 2)
- Establishing credentials for the AWS Toolkit for Visual Studio Code (p. 3)
- Connecting to AWS through the AWS Toolkit for Visual Studio Code (p. 8)
- Changing AWS Regions (p. 10)
- Configuring your toolchain (p. 13)

Installing the AWS Toolkit for Visual Studio Code

This section describes how to install the AWS Toolkit for Visual Studio Code.

Prerequisites

Required

Before you can install the Toolkit for VS Code, you must have the following:

- An Amazon Web Services account – To obtain an AWS account, go to the AWS home page. Choose Create an AWS Account, or Complete Sign Up (if you've visited the site before). Signing up enables you to use all of the services that AWS offers.
- A supported operating system – The Toolkit for VS Code is supported on Windows, Linux, and macOS.
- VS Code version 1.31.1 or later – We try to keep the Toolkit for VS Code current with the default version that's available on the VS Code download page.

Optional

Before you can use certain features of the Toolkit for VS Code, you must have the following:

- Code Development – The relevant SDK for the language that you want to use. You can download from the following links, or use your favorite package manager:
  - .NET SDK: https://dotnet.microsoft.com/download
  - Node.js SDK: https://nodejs.org/en/download
  - Python SDK: https://www.python.org/downloads
- AWS SAM CLI – This is an AWS CLI tool that helps you develop, test, and analyze your serverless applications locally. This isn't required for installing the toolkit. However, we recommend that you install it (and Docker, described next) because it's required for any AWS Serverless Application Model (AWS SAM) functionality, such as Creating serverless applications (p. 21).

For more information, see Installing the AWS SAM CLI in the AWS Serverless Application Model Developer Guide.
• **Docker** – The AWS SAM CLI requires this open-source software container platform. For more information and download instructions, see Docker.

### Install the Toolkit for VS Code

1. Start the VS Code editor.
2. In the **Activity Bar** on the side of the VS Code editor, choose the **Extensions** icon. This opens the **Extensions** view, which allows you to access the VS Code Marketplace.

![Extensions View](image)

3. In the search box for **Extensions**, search for **AWS Toolkit**. Choose the entry to see its details in the right pane.
4. In the right pane, choose **Install**.
5. Once installed, if you're prompted to restart the editor, choose **Reload Required** to finish installation.

After you install the Toolkit for VS Code, you should configure your credentials (p. 3) to enable you to access your AWS resources from within VS Code.

### Establishing credentials for the AWS Toolkit for Visual Studio Code

This section shows you the types of credentials that you can use with the AWS Toolkit for VS Code. It provides information about how to get and configure those credentials.

You can obtain credentials through AWS and provide them to the toolkit by using configuration files. You can also obtain credentials through an external credential process that isn't directly supported by AWS.
Using AWS-supported credentials

AWS credentials can be provided to the AWS Toolkit for VS Code by using your shared AWS config file or your shared AWS credentials file. The methods for using these files are the same as those for the AWS CLI. For general information about how to use these files, see Configuration and Credential Files in the AWS Command Line Interface User Guide.

Obtaining AWS access keys

To access Amazon Web Services (AWS) with the AWS Toolkit for Visual Studio Code, you must configure the toolkit with AWS account credentials. To do this with AWS-supported credentials, you must first obtain appropriate AWS access keys.

For more information about users and credentials that is out of scope for this guide, see the following resources:

- AWS Security Credentials in the Amazon Web Services General Reference
- Overview of Identity Management: Users in the IAM User Guide

What are AWS access keys

Access keys are the credentials that identify you to AWS and enable you to programmatically access AWS services and resources. Access keys can be associated with your AWS account (the account's “root user”) or with users that you create with AWS Identity and Access Management (IAM).

Warning
Because the root user is essentially an administrator with full access to services and resources, we recommend that you instead create an IAM user with only those permissions needed to perform the required tasks. Then, for your credentials, you can use an access key that is associated with that user. For details, see Creating an IAM User in Your AWS Account in the IAM User Guide.

An access key consists of an access key ID, which is similar to a user name, and a secret access key, which is similar to a password. This access key is used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them by using the AWS Management Console. We recommend that you use access keys for an IAM user instead of the keys for your account's root user.

Note
To create access keys, you must have permissions to perform the required IAM actions. For more information, see Granting IAM User Permissions to Manage Password Policy and Credentials in the IAM User Guide.

Get your AWS access keys

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. On the navigation menu, choose Users.
3. Choose your IAM user name (not the check box) to view its details.
4. On the **Security Credentials** tab, choose **Create access key**.
5. To see the new access key, choose **Show**. The credentials resemble the following:
   - Access key ID: AKIAIOSFODNN7EXAMPLE
   - Secret access key: wJalrXUttnFEMI/K7MDENG/bPxRfYEXAMPLEKEY
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location.

**Important**
- Keep the keys confidential to protect your AWS account, and never email them. Do not share them outside of your organization, even if an inquiry appears to come from AWS or Amazon.com. **No one who legitimately represents Amazon will ever ask you for your secret key.**
- You can't recover the secret key if you lose it. Instead, for security reasons, you must create a new key pair and delete the old pair.

After you have obtained your AWS access keys, you can use the AWS Toolkit for VS Code to store them in your shared AWS config file or your shared AWS credentials file. See [Add your AWS access keys to your environment](p. 5) to learn how.

**Setting up your AWS credentials**

To access Amazon Web Services (AWS) with the AWS Toolkit for Visual Studio Code, you must make your AWS account credentials available to the toolkit. To use AWS-supported credentials, continue reading in this topic. To use an external credential process, see [Using an external credential process](p. 7).

**Note**
Some features of the Toolkit for VS Code, such as creating a serverless application, don't require AWS credentials.

**Get your AWS access keys**

If you don't already have appropriate AWS access keys to store in your shared AWS config file or your shared AWS credentials file, you must get them now.

To do so, see [Obtaining AWS access keys](p. 4).

**About shared AWS files**

*Your shared AWS config file and your shared AWS credentials file* are files that you can use to store configuration and credential information for AWS. By default, these files are located in the `.aws` directory within your home directory and are named `config` and `credentials`, respectively. For more information, see [Where Are Configuration Settings Stored?](in the AWS Command Line Interface User Guide).

The Toolkit for VS Code locates and uses AWS access keys through your shared AWS config file and your shared AWS credentials file. This is the method that is used by the AWS CLI and the AWS SDKs. Access keys that you enter in the Toolkit for VS Code are saved to one of these files.

These shared files can contain the credentials for more than one AWS account, stored as *profiles*. Multiple accounts can be useful, for example, to provide developers and administrators with separate resources for development and for release or publication.

**Add your AWS access keys to your environment**

If you have already set your AWS credentials (for example, by using the AWS CLI), the Toolkit for VS Code will automatically detect those credentials and make them available to the toolkit. If you haven't already
set your AWS credentials, or if you want to include additional AWS credentials in your environment or update an existing credentials profile, you can do so through the Toolkit for VS Code, as shown here.

**Note**
As an alternative to these procedures, you can use the `aws configure` AWS CLI command to add AWS credentials to your environment. You can also use `aws configure` to set the default AWS Region, which is needed for certain operations such as creating a serverless application.

**Create the shared AWS credentials file**

- If you already have a shared AWS credentials file, you can skip to the next procedure (p. 7).
- If you already have a shared AWS `config` file and want to use it, you can skip to the next procedure (p. 7).
- If you have only a shared AWS `config` file but do NOT want to use it, you must first create a shared AWS credentials file by using techniques that are normal for your operating-system. After that, you can skip to the next procedure (p. 7).

Follow these steps to create the shared AWS credentials file.

1. Open VS Code.
2. To open the **Command Palette**, on the menu bar, choose **View**, **Command Palette**. Or use the following shortcut keys:
   - Windows and Linux – Press `Ctrl+Shift+P`.
   - macOS – Press `Shift+Command+P`.
3. Search for **AWS** and choose **AWS: Create Credentials Profile**.
4. Enter a name for the initial profile.
5. Enter the **access key ID** for the credentials. If you don’t have an access key ID, see **Obtaining AWS access keys** (p. 4).
6. Enter the **secret access key** for the credentials. If you don’t have a secret access key, see **Obtaining AWS access keys** (p. 4).
After you complete this procedure, you can verify the shared AWS credentials file by opening it directly or by performing the first three steps of the next procedure (without making any updates).

**Update one of your shared files**

Follow these steps to add a new profile to your existing shared AWS config file or shared AWS credentials file. You can also update an existing profile.

1. Open VS Code.
2. To open the Command Palette, on the menu bar, choose View, Command Palette. Or use the following shortcut keys:
   - Windows and Linux – Press **Ctrl+Shift+P**.
   - macOS – Press **Shift+Command+P**.
3. Search for AWS and choose AWS: Create Credentials Profile.
4. When one or both of the shared files opens in the VS Code editor, add or update a profile.
5. When you're finished updating the file, save it.

**Using an external credential process**

If you have a process to generate or lookup credentials that isn't directly supported by AWS, you can configure the AWS Toolkit for VS Code to use that process instead of any stored AWS credentials (p. 4).

The method for specifying such an external credential process is the same as for the AWS CLI, and consists of adding a credential_process definition to your shared AWS config file. For detailed information about how to do this, see Sourcing Credentials with an External Process in the AWS Command Line Interface User Guide.
Connecting to AWS through the AWS Toolkit for Visual Studio Code

To interact with Amazon Web Services (AWS) through the AWS Toolkit for Visual Studio Code, you must establish a connection to AWS.

Connect to AWS through the Toolkit for VS Code

1. Open VS Code.
2. To open the Command Palette, on the menu bar, choose View, Command Palette. Or use the following shortcut keys:
   - Windows and Linux – Press Ctrl+Shift+P.
   - macOS – Press Shift+Command+P.
3. Search for AWS and choose AWS: Connect to AWS.

4. Choose a profile from the list.

If you don’t have a credentials profile set up, you are prompted to set one up. Look for a pop-up in the lower-right corner of the editor. Choose Yes, and then follow the setup wizard to enter a profile name, your access key ID, and your secret access key. For details, see Setting up your AWS credentials (p. 5).

Note
If you want to provide an external credential process instead of using AWS-supported credentials, choose No and see Using an external credential process (p. 7) instead.
5. Open the **AWS: Explorer** Side Bar, which we call the **AWS Explorer**, to verify the connection. You will see either a list of AWS Regions (if you have made any Regions visible (p. 11) in the **AWS Explorer**) or a message to add Regions to the **AWS Explorer**.

Before adding Regions (p. 10) to the **AWS Explorer**, you see the following.

After adding Regions to the **AWS Explorer**, you see something like the following.
Use multiple AWS accounts with the Toolkit

You might have multiple AWS accounts that you want to access from the Toolkit for VS Code. Multiple accounts can be useful, for example, to provide developers and administrators with separate resources for development and for release or publication.

Separate sets of AWS credentials are stored as profiles within the shared AWS config file or the shared AWS credentials file. To choose a different set of credentials, follow the steps in the previous procedure, and choose a different profile.

Changing AWS Regions

When you set up your credentials, the AWS Toolkit for Visual Studio Code automatically selects and shows the default AWS Region for those credentials in the AWS Explorer. This topic describes how to change the list of Regions that is shown in the AWS Explorer.
Add a Region to the AWS Explorer

1. To open the Command Palette, on the menu bar, choose View, Command Palette. Or use the following shortcut keys:
   - Windows and Linux – Press Ctrl+Shift+P.
   - macOS – Press Shift+Command+P.

2. Search for AWS and choose AWS: Show Region in the Explorer.

3. Choose the Region that you want to add to the AWS Explorer.

   Note
   The list contains only those Regions that are available to add to the AWS Explorer. Regions you've already added don't appear in the list.

4. Verify that the Region now appears in the AWS Explorer.
Hide a Region from the AWS Explorer

1. Choose the AWS icon in the Activity bar to open the AWS Explorer.
2. Choose one of the Regions in the list, and open its context menu.
Configuring your toolchain

The AWS Toolkit for Visual Studio Code supports multiple languages that you can use to interact with AWS. This topic describes how to set up the toolchain for each of these languages.

Configure a toolchain for .NET Core

1. Ensure that you have the Toolkit for VS Code installed (p. 3).
2. Install the C# extension. This extension enables VS Code to debug .NET Core applications.
3. Open an AWS Serverless Application Model (AWS SAM) application, or create one (p. 21).
4. Open the folder that contains template.yaml.

Configure a toolchain for Node.js

1. Ensure that you have the Toolkit for VS Code installed (p. 3).
Configure a toolchain for Python

1. Ensure that you have the Toolkit for VS Code installed (p. 3).
2. Install the Python extension for Visual Studio Code. This extension enables VS Code to debug Python applications.
3. Open an AWS SAM application, or create one (p. 21).
4. Open the folder that contains template.yaml.
5. Open a terminal at the root of your application, and configure virtualenv by running `python -m venv ./.venv`.

   **Note**
   You only need to configure virtualenv once per system.
6. Activate virtualenv by running one of the following:

   - Bash shell: `./.venv/Scripts/activate`
   - PowerShell: `./.venv/Scripts/Activate.ps1`

Using Your toolchain

Once you have your toolchain set up, you can use it to run or debug (p. 23) the AWS SAM application.
Navigating the AWS Toolkit for Visual Studio Code

This topic describes how to navigate in the AWS Toolkit for Visual Studio Code. Be sure to first install the toolkit (p. 2) before reading this topic.

For general information about the Visual Studio Code user interface (UI), see the UI documentation for VS Code.

Fundamental UI Components

The following are the basic UI components of the AWS Toolkit for VS Code.

The AWS Explorer

- The following image shows the basic UI components of the **AWS Explorer**. See Working with AWS Services in the AWS Explorer (p. 19) for details about what you can do with it.

The **AWS Explorer** is more formally known as the **AWS: Explorer Side Bar**.
The AWS CDK Explorer

- The following image shows the basic UI components of the AWS CDK Explorer. See Working with the AWS CDK Explorer (p. 47) for details about what you can do with it.

This is prerelease documentation for a feature in preview release. It is subject to change.

Command Locations

You can find the commands for the Toolkit for VS Code in various locations.

Visual Studio Code Command Palette

The VS Code Command Palette contains all the commands you can use for working with AWS services. To see these commands, open the VS Code Command Palette from the View menu. Or use the following shortcuts:

- Windows and Linux – Press Ctrl+Shift+P.
- macOS – Press Shift+Command+P.
Then you can filter for "AWS".
The AWS Explorer has a More Actions menu that contains the most common commands for working with AWS services.
Working with AWS Services

The AWS Toolkit for Visual Studio Code enables you a view and work with various AWS services. It assumes that you've already installed and configured (p. 2) the Toolkit for VS Code on your system.

Topics
- Working with AWS Services in the AWS Explorer (p. 19)
- Working with Amazon Elastic Container Service (p. 46)
- Working with the AWS CDK Explorer (p. 47)

Working with AWS Services in the AWS Explorer

The AWS Explorer gives you a view of some of the AWS services that you can work with when using the AWS Toolkit for Visual Studio Code. See a high-level view of the AWS Explorer in the navigation (p. 15) topic.

This section provides information about how to access and use the AWS Explorer in VS Code. It assumes that you've already installed and configured (p. 2) the Toolkit for VS Code on your system.

Some important points:
- If the toolkit is installed and configured correctly, you should see items in the AWS Explorer. To see the AWS Explorer, choose the AWS icon in the Activity bar.

For example:
• Certain features require certain AWS permissions. For example, to see the AWS Lambda functions in your AWS account, the credentials you configured in Establishing credentials (p. 3) must include at least read-only Lambda permissions. See the following topics for more information about the permissions that each feature needs.

**Topics**
- Working with AWS Serverless Application (p. 21)
- Working with AWS CloudFormation stacks (p. 30)
- Working with AWS Lambda Functions (p. 31)
- Working with Amazon S3 (p. 34)
- Working with Amazon EventBridge (p. 37)
- Working with AWS Step Functions (p. 39)
Working with AWS Serverless Application

The AWS Toolkit for Visual Studio Code provides support for serverless applications. Using the Toolkit for VS Code, you can create serverless applications that contain AWS Lambda functions, and then deploy the applications to an AWS CloudFormation stack.

Topics

- Creating serverless applications with the AWS Toolkit for Visual Studio Code (p. 21)
- Running and debugging serverless applications with the AWS Toolkit for Visual Studio Code (p. 23)
- Deploying serverless applications with the AWS Toolkit for Visual Studio Code (p. 29)

Creating serverless applications with the AWS Toolkit for Visual Studio Code

This example shows how to use the AWS Toolkit for Visual Studio Code to create a serverless application and then run and debug it.

Prerequisites

- Be sure your system meets the required prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- In addition, install the AWS SAM CLI and its prerequisites. See AWS SAM CLI in the setup prerequisites (p. 2). If Visual Studio Code is open when you perform these installations, you might need to close and reopen the editor.
- Identify your default AWS Region in your AWS config file. For more information, see Configuration and Credential Files in the AWS CLI User Guide. If Visual Studio Code is open when you update your config file, you might need to close and reopen the editor.
- After installing your language SDK, be sure to configure your toolchain (p. 13).

Creating serverless applications with the Toolkit for VS Code

This example shows how to create a serverless application with the Toolkit for VS Code by using the AWS Serverless Application Model (AWS SAM).

1. To open the Command Palette, choose View, Command Palette, and then enter AWS.
2. Choose AWS: Create new SAM Application.

Note

If the AWS SAM CLI isn't installed, you will get an error in the lower-right corner of the VS Code editor. If this happens, see the prerequisites (p. 21) described earlier.

3. Choose the runtime for your SAM application. For this example, choose nodejs12.x.
4. Choose a location for your new project. You can use an existing workspace folder if one is open, **Select a different folder** that already exists, or create a new folder and select it. For this example, choose **There are no workspace folders open** to create a folder named **MY-SAM-APP**.

5. Enter a name for your new project. For this example, use **my-sam-app-nodejs**. After you press **Enter**, the Toolkit for VS Code takes a few moments to create the project.

When the project is created, your application is added to your current workspace. You should see it listed in the **Explorer** window.
Running and debugging serverless applications with the AWS Toolkit for Visual Studio Code

With AWS Toolkit for Visual Studio Code you can configure how to debug serverless applications and run them locally in your development environment. You can debug a serverless application that's defined by a SAM (Serverless Application Model) template. This template uses simple YAML syntax to describe resources such as functions, APIs, databases, and event-source mappings that make up a serverless application.

For a breakdown of the AWS SAM template, see the AWS SAM Template Anatomy in the AWS Serverless Application Model Developer Guide.

Alternatively, you can rapidly debug serverless applications that haven't been committed to a SAM template.

You start to configure debug behavior by using VS Code's CodeLens feature to identify an eligible Lambda function. (CodeLens is a Visual Studio Code feature that enables content-aware interactions with your source code.) If you want to use the infrastructure defined by the SAM template, use the CodeLens indicator in the relevant YAML-formatted file. If you want to test the function directly without the template, use the CodeLens indicator for the lambda handler in the application file.

Note
In this example, we're debugging an application that uses JavaScript. But you can use debugging features available in AWS Toolkit for Visual Studio Code with the following languages and runtimes:

- C#: .NET Core 2.1 (.NET Core 3.1 serverless applications can be run but not debugged using the Toolkit for VS Code.)
Working with AWS Serverless Application

- JavaScript: Node.js 10.x, 12.x
- Python: 2.7, 3.6, 3.7, 3.8

Your language choice also affects how VS Code's CodeLens detects eligible lambda handlers. For more information, see Running and debugging serverless functions directly from code (p. 25).

Using SAM templates to run and debug serverless applications

For applications that are run and debugged using a SAM template, a YAML-formatted file describes the application's behavior and the resources it uses. If you create a serverless application with the AWS: Create new SAM Application option in the Command Palette, a file called template.yaml is automatically generated for your project.

In this procedure, you'll use the sample application that was created in Creating serverless applications with the Toolkit for VS Code (p. 21).

**Note**
To ensure that you can access the CodeLens feature in the template file, install a VS Code extension that provides YAML language support (for example, YAML Language Support by Red Hat).

1. To view your application files in VS Code's File Explorer, choose View, Explorer.
2. From the application directory (my-sample-app, for example), open the template.yaml file.

**Note**
If you use a template with a name that's different from template.yaml, the CodeLens indicator isn't automatically available in the YAML file. This means you'll need to manually add a debug configuration.

3. In the editor for template.yaml, go to the Resources section of the template that defines serverless resources. In this case, the HelloWorldFunction resource of type AWS::Serverless::Function.

In the CodeLens indicator for this resource, click Add Debug Configuration.

A new editor displays the launch.json file that provides a debugging configuration with default attributes.

4. Edit or confirm values for the following configuration properties:
   - "name": Enter a reader-friendly name to appear in the Debug launch configuration drop-down.
   - "target": Ensure the value is "template" so that the SAM template is the entry point for the debug session.
   - "templatePath": Enter a relative or absolute path for the template.yaml file.
   - "logicalId": Ensure the name matches the one specified in the Resources section of SAM template. In this case, it's the HelloWorldFunction of type AWS::Serverless::Function.

**Note**
You can use VS Code's IntelliSense feature to find and autocomplete additional debug properties for your application. For more information, see Configuration options for debugging serverless applications (p. 26).

5. If you're happy with your debug configuration, save launch.json, and then click the green "play" button in the RUN view to start debugging.

When the debugging sessions starts, the DEBUG CONSOLE panel shows debugging output and displays any values returned by the Lambda function.
When debugging SAM applications, the **AWS Toolkit** is selected as the Output channel in the **Output** panel.

**Note**
For Windows users, if you see a Docker mounting error during this process, you might need to refresh the credentials for your shared drives (in Docker Settings). A Docker mounting error looks like this:

```
Fetching lambci/lambda:nodejs10.x Docker container image......
2019-07-12 13:36:58 Mounting C:\Users\<username>\AppData\Local\Temp\ ... as /var/task:ro,delegated inside runtime container
Traceback (most recent call last):
...
requests.exceptions.HTTPError: 500 Server Error: Internal Server Error ...
```

**Running and debugging serverless functions directly from code**

When testing the AWS SAM application, you can choose to run and debug just the Lambda function and exclude other resources defined by the SAM template. This approach involves using the **CodeLens** feature to identify lambda function handlers in the source code that can be directly invoked.

The lambda handlers that are detected by CodeLens depend on the language and runtime you're using for your application:

<table>
<thead>
<tr>
<th>Language/runtime</th>
<th>Criteria for adding CodeLens indicators to a Lambda function</th>
</tr>
</thead>
</table>
| C# (dotnetcore2.1) | • Is a public function of a public class  
                     • Has one or two parameters. With two parameters, the second parameter must implement the ILambdaContext interface  
                     • Has a *.csproj file in its parent directory within the VS Code workspace folder |
| JavaScript (Node.js 10.x and 12.x): | • Is an exported function with up to three parameters  
                                     • Has a package.json in its parent directory within the VS Code workspace folder |
| Python (node.js 10.x and 12.x): | • Is a top-level function  
                                     • Has a requirements.txt file in its parent directory within the VS Code workspace folder |

1. To view your application files in VS Code’s File Explorer, choose **View, Explorer**.
2. From the application directory (*my-sample-app*, for example), expand the function directory (*hello-world*, in this case) and open the **app.js** file.
3. In the CodeLens indicator that identifies an eligible lambda handler function, click **Add Debug Configuration**.
4. In the **Command Palette**, select the runtime in which your SAM application will run.
5. Now, in the editor for the **launch.json** file, edit or confirm values for the following configuration properties:
6. If you're happy with the debug configuration, click the green play arrow beside RUN to start debugging.

7. When the debugging sessions starts, the DEBUG CONSOLE panel shows debugging output and displays any values returned by the Lambda function.

   Note
   When debugging SAM applications, AWS Toolkit is selected as the Output channel in the Output panel.

   Note
   If you see Docker mentioned in error messages, refer to this note.

Configuration options for debugging serverless applications

When you open the launch.json to edit debug configurations, you can use VS Code's IntelliSense feature to view and autocomplete valid properties. To trigger IntelliSense in the editor, press Ctrl+Space.

With IntelliSense, you can easily find and define properties for invoking Lambda functions directly or with the SAM template. You can also define properties for "lambda" (how the function runs), "sam" (how the AWS SAM CLI builds the application), and "aws" (how AWS connection information is provided).

AWS SAM: Direct Lambda handler invoke / Template-based Lambda invoke

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies which extension manages the launch configuration. Always set to aws-sam to use the AWS SAM CLI to build and debug locally.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>name</td>
<td>Specifies a reader-friendly name to appear in the Debug launch configuration drop-down.</td>
</tr>
<tr>
<td>request</td>
<td>Specifies the type of configuration to be performed by the designated extension (aws-sam). Always set to direct-invoke to start the Lambda function.</td>
</tr>
<tr>
<td>invokeTarget</td>
<td>Specifies the entry point for invoking the resource.</td>
</tr>
<tr>
<td></td>
<td>For invoking the Lambda function directly, set values for the following invokeTarget fields:</td>
</tr>
<tr>
<td></td>
<td>• target: code</td>
</tr>
<tr>
<td></td>
<td>• lambdaHandler: The name of the lambda function handler to invoke</td>
</tr>
<tr>
<td></td>
<td>• projectRoot: The path for the application file containing the lambda handler</td>
</tr>
<tr>
<td></td>
<td>For invoking the Lambda resources with the SAM template, set values for the following invokeTarget fields:</td>
</tr>
<tr>
<td></td>
<td>• target:template</td>
</tr>
<tr>
<td></td>
<td>• templatePath: The path to the SAM template file</td>
</tr>
<tr>
<td></td>
<td>• logicalId: The resource name of the AWS::Lambda::Function or AWS::Serverless::Function to invoke. The resource name can be found in the YAML-formatted SAM template.</td>
</tr>
</tbody>
</table>

### Lambda ("lambda") properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>environmentVariables</td>
<td>Passes operational parameters to your function. For example, if you're writing to an Amazon S3 bucket, instead of hard-coding the bucket name you're writing to, configure the bucket name as an environment variable.</td>
</tr>
<tr>
<td>payload</td>
<td>Describes in a JSON file format the event that triggers a Lambda function.</td>
</tr>
<tr>
<td></td>
<td>You can create an event payload by running the following command in the Terminal of VS Code: sam local generate-event apigateway aws-proxy</td>
</tr>
<tr>
<td>memoryMB</td>
<td>Specifies megabytes of memory provided for the execution of invoked Lambda function.</td>
</tr>
<tr>
<td>runtime</td>
<td>Specifies runtime used by the Lambda. For more information, see AWS Lambda runtimes.</td>
</tr>
<tr>
<td>timeoutSec</td>
<td>Sets the time allowed, in seconds, before the debug session times out.</td>
</tr>
</tbody>
</table>
AWS Toolkit for Visual Studio Code uses AWS SAM CLI to build and debug serverless applications locally. You can configure the behavior of AWS SAM CLI commands using properties of the "sam" configuration in the `launch.json` file.

### SAM CLI ("sam") properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buildArguments</code></td>
<td>Configures how the <code>sam build</code> command builds your Lambda source code. To view build options, see <code>sam build</code> in the <a href="https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/index.html">AWS Serverless Application Model Developer Guide</a>.</td>
<td>Empty string</td>
</tr>
<tr>
<td><code>containerBuild</code></td>
<td>Indicates whether to build your function inside an AWS Lambda-like Docker container.</td>
<td>false</td>
</tr>
<tr>
<td><code>dockerNetwork</code></td>
<td>Specifies the name or id of an existing Docker network to Lambda Docker containers should connect to, along with the default bridge network. If not specified, the Lambda containers will only connect to the default bridge Docker network.</td>
<td>Empty string</td>
</tr>
<tr>
<td><code>localArguments</code></td>
<td>Additional local invoke arguments.</td>
<td>Empty string</td>
</tr>
<tr>
<td><code>skipNewImageCheck</code></td>
<td>Specifies whether the command should skip pulling down the latest Docker image for Lambda runtime.</td>
<td>false</td>
</tr>
<tr>
<td><code>template</code></td>
<td>Customizes your SAM template by using parameters to input customer values to your template. For more information, see Parameters in the <a href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/glossary.html">AWS CloudFormation User Guide</a>.</td>
<td>&quot;parameters&quot;:{}</td>
</tr>
</tbody>
</table>

### AWS Connection ("aws") properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>credentials</code></td>
<td>Selects a specific profile (profile:default, for example) from your credential file to get AWS credentials.</td>
<td>The AWS credentials provided to the AWS Toolkit for Visual Studio Code by your existing shared AWS config file or shared AWS credentials file (p. 5).</td>
</tr>
<tr>
<td><code>region</code></td>
<td>Sets the AWS Region of the service (for example, us-east-1).</td>
<td>The default region associated with the active credentials profile.</td>
</tr>
</tbody>
</table>
Deploying serverless applications with the AWS Toolkit for Visual Studio Code

This example shows how to deploy the serverless application that was created in the previous topic (Creating serverless applications (p. 21)) to AWS using the AWS Toolkit for Visual Studio Code.

Prerequisites

- Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- Be sure to choose a globally unique Amazon S3 bucket name.
- Ensure that the credentials you configured in Establishing credentials (p. 3) include the appropriate read/write access to the following services: Amazon S3, AWS CloudFormation, AWS Lambda, and Amazon API Gateway.

Deploying serverless applications

1. To open the Command Palette, choose View, Command Palette, and then enter AWS.
2. Choose AWS: Deploy SAM Application.
3. Choose the template.yaml file to use for the deployment.
4. Choose the AWS Region to deploy to.
5. Enter the name of an Amazon S3 bucket this deployment can use. The bucket must be in the Region you're deploying to.

Warning
The Amazon S3 bucket name must be globally unique across all existing bucket names in Amazon S3. Therefore, you should add a unique identifier to the name given in the following example (or choose a different name).
6. Enter a name for the deployed stack, either a new stack name or an existing stack name.

7. Verify the success of the deployment on the **OUTPUT** tab of VS Code.

If an error occurs, a message pops up in the lower-right that is similar to the following:

Failed to deploy SAM application.

If this happens, check the text in the **OUTPUT** tab for details. The following is an example of error details.

```
S3 Bucket does not exist. Execute the command to create a new bucket
aws s3 mb s3://pbart-my-sam-app-bucket
An error occurred while deploying a SAM Application. Check the logs for more information by running the "View AWS Toolkit Logs" command from the Command Palette.
```

In this example, the error occurred because the Amazon S3 bucket did not exist.

When the deployment is complete, you'll see your application listed in the **AWS Explorer**. To learn how to invoke the Lambda function that was created as part of the application, see Interacting with Remote Lambda Functions (p. 32).

**Working with AWS CloudFormation stacks**

The AWS Toolkit for Visual Studio Code provides support for **AWS CloudFormation** stacks. Using the Toolkit for VS Code, you can perform certain tasks with AWS CloudFormation stacks, such as deleting them.

**Topics**

- Deleting an AWS CloudFormation stack (p. 31)
Deleting an AWS CloudFormation stack

You can use the AWS Toolkit for Visual Studio Code to delete AWS CloudFormation stacks.

**Prerequisites**

- Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- Ensure that the credentials you configured in Establishing credentials (p. 3) include appropriate read/write access to the AWS CloudFormation service. If in the AWS Explorer, under CloudFormation, you see a message similar to "Error loading CloudFormation resources", check the permissions attached to those credentials. Changes that you make to permissions will take a few minutes to affect the AWS Explorer in VS Code.

**Delete a CloudFormation stack**

1. In the AWS Explorer, open the context menu of the AWS CloudFormation stack you want to delete.

2. Choose Delete CloudFormation Stack.

3. In the message that appears, choose Yes to confirm the delete.

After the stack is deleted, it's no longer listed in the AWS Explorer.

Working with AWS Lambda Functions

The AWS Toolkit for Visual Studio Code provides support for AWS Lambda functions. Using the Toolkit for VS Code, you can author code for Lambda functions that are part of serverless applications. In addition, you can invoke Lambda functions either locally or on AWS.

Lambda is a fully managed compute service that runs your code in response to events generated by custom code or from various AWS services, such as Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Kinesis, Amazon Simple Notification Service (Amazon SNS), and Amazon Cognito.

**Topics**

- Interacting with Remote Lambda Functions (p. 32)
Interacting with Remote Lambda Functions

Using the Toolkit for VS Code, you can interact with AWS Lambda functions in various ways, as described later in this topic.

For more information about Lambda, see the AWS Lambda Developer Guide.

**Note**
If you have already created Lambda functions by using the AWS Management Console or in some other way, you can invoke them from the Toolkit. To create a new function (using VS Code) that you can deploy to AWS Lambda, you must first create a serverless application (p. 21).

**Prerequisites**

- Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- Ensure that the credentials you configured in Establishing credentials (p. 3) include appropriate read/write access to the AWS Lambda service. If in the AWS Explorer, under Lambda, you see a message similar to "Error loading Lambda resources", check the permissions attached to those credentials. Changes that you make to permissions will take a few minutes to affect the AWS Explorer in VS Code.

**Invoke a Lambda Function**

You can invoke a Lambda function on AWS from the Toolkit for VS Code.

1. In the AWS Explorer, choose the name of the Lambda function you want to invoke, and then open its context menu.

2. Choose **Invoke on AWS**.

3. In the invoke window that opens, enter the input that your Lambda function needs. The Lambda function might, for example, require a string as an input, as shown in the text box.
AWS Toolkit for VS Code User Guide
AWS Lambda Functions

You'll see the output of the Lambda function just like you would for any other project using VS Code.

Delete a Lambda Function

You can also delete a Lambda function using the same context menu.

**Warning**
Do not use this procedure to delete Lambda functions that are associated with AWS CloudFormation (for example, the Lambda function that was created when creating a serverless application (p. 21) earlier in this guide). These functions must be deleted through the AWS CloudFormation stack.
1. In the **AWS Explorer**, choose the name of the Lambda function you want to delete, and then open its context menu.

2. Choose **Delete**.
3. In the message that appears, choose **Yes** to confirm the delete.

After the function is deleted, it’s no longer listed in the **AWS Explorer**.

### Working with Amazon S3

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

**Topics**
- Working with Amazon S3 buckets (p. 34)
- Working with Amazon S3 objects (p. 36)

### Working with Amazon S3 buckets

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.
Creating an Amazon S3 bucket

1. In the AWS Explorer, right-click the S3 node, and then choose Create Bucket. Or choose the Create Bucket icon across from the S3 node.

2. In the Bucket Name field, enter a valid name for the bucket. Press Enter to confirm.

   The new bucket is displayed under the S3 node.

   **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.
   You can check the AWS Toolkit Logs in the Output tab if you can't create a bucket. For example, a BucketAlreadyExists error occurs if you use a bucket name already in use.
   For more information, see Bucket restrictions and limitations in the Amazon Simple Storage Service Developer Guide.

   After a bucket is created, you can copy its name and Amazon Resource Name (ARN) to the clipboard. Right-click the bucket entry and select the relevant option from the menu.

Adding a folder to an Amazon S3 bucket

You organize a bucket's contents by grouping objects in folders. You can also create folders within other folders.

1. In the AWS Explorer, choose the Amazon S3 node to view the list of buckets.
2. Right-click a bucket or a folder, and then choose Create Folder. Or choose the Create Folder icon across from the name of the bucket or folder.
3. Enter a Folder Name, and then press Enter.

Deleting an Amazon S3 bucket

When you delete a bucket you also delete the folders and objects that it contains. You're asked to confirm that you want to do this before the deletion is carried out.

**Note**
To delete only a folder, not the entire bucket, use the AWS Management Console.

1. In the AWS Explorer, choose the S3 node to expand the list of buckets.
2. Right-click the bucket to delete, and then choose Delete S3 Bucket.
3. Enter the bucket's name to confirm the deletion, and then press Enter.

   **Note**
   If the bucket contains objects, the bucket is emptied before deletion. This can take some time if it's necessary to delete every version of thousands of objects. A notification is displayed after the deletion is complete.
Configuring the display of Amazon S3 items

If you're working with a large number of Amazon S3 objects or folders, you can specify how many are displayed at one time. When the maximum number is displayed, you can choose Load More to display the next batch.

1. In the VS Code Activity Bar, choose Extensions.
2. For the AWS Toolkit extension, choose the settings icon, and then choose Extension Settings.
3. On the Settings page, scroll down to the AWS > S3: Max Items Per Page setting.
4. Change the default value to the number of S3 items you want displayed before choosing to load more.

   **Note**
   The range of accepted values is between 3 and 1000. This setting applies only to the number of objects or folders displayed at one time. All the buckets you've created are displayed at once. (By default, you can create up to 100 buckets in each of your AWS accounts.)
5. Close the Settings page to confirm your changes.

You can also update the settings in a JSON-formatted file by choosing the Open Settings (JSON) icon in the upper right of the Settings page.

Working with Amazon S3 objects

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

**Topics**
- Uploading a file to an Amazon S3 bucket (p. 36)
- Downloading an Amazon S3 object (p. 37)
- Deleting an Amazon S3 object (p. 37)

**Uploading a file to an Amazon S3 bucket**

This procedure uploads a file from a user's system and stores it as an S3 object in the AWS Cloud. You can upload a file to a bucket or to a folder that organizes that bucket's contents.

1. In the AWS Explorer, choose the Amazon S3 node to view the list of buckets.
2. Right-click a bucket or a folder in that bucket, and then choose Upload File. Or choose the Upload File icon across from the name of the bucket or folder.

   **Note**
   If you right-click an S3 object, you can choose Upload to Parent. This enables you to add a file to the folder or bucket that contains the selected file.
3. Using your system's file manager, select a file, and then choose Upload File.
The selected file is uploaded as an S3 object to the bucket or folder. Each object's entry describes the size of the stored object and how long ago it was uploaded. You can pause over the object's listing to view the path, size, and time when it was last modified.

### Downloading an Amazon S3 object

You can download objects in an Amazon S3 bucket from the AWS Cloud to your system.

1. In the **AWS Explorer**, choose the **S3** node to view the list of buckets.
2. In a bucket or in a folder in a bucket, right-click an object, and then choose **Download As**. Or choose the **Download As** icon across from the name of the object.
3. Using your system's file manager, select a destination folder, enter a file name, and then choose **Download**.

After a file is downloaded, you can open it in the IDE interface using the **AWS Toolkit Logs** on the **Output** tab. Depending on your operating system, press **Cmd** or **Ctrl** and click the link to the file's location.

### Deleting an Amazon S3 object

You can permanently delete an object if it's in a non-versioned bucket. But for versioning-enabled buckets, a delete request does not permanently delete that object. Instead, Amazon S3 inserts a delete marker in the bucket. For more information, see Deleting object versions in the [Amazon Simple Storage Service Developer Guide](https://docs.aws.amazon.com/AmazonS3/latest/dev/DeletingObjectVersions.html).

1. In the **AWS Explorer**, choose the **S3** node to view the list of buckets.
2. In a bucket or a folder in a bucket, right-click an object, and then choose **Delete**.
3. Choose **Delete** to confirm.

### Working with Amazon EventBridge

The AWS Toolkit for Visual Studio Code (VS Code) provides support for Amazon EventBridge. Using the Toolkit for VS Code, you can work with certain aspects of EventBridge, such as schemas.

**Topics**
- Working with Amazon EventBridge Schemas (p. 37)

#### Working with Amazon EventBridge Schemas

You can use the AWS Toolkit for Visual Studio Code (VS Code) to perform various operations on Amazon EventBridge schemas.

**Prerequisites**
- Be sure your system meets the the prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- The EventBridge schema you want to work with must be available in your AWS account. If it isn’t, create or upload it. See Amazon EventBridge Schemas in the Amazon EventBridge User Guide.
• Open the AWS Explorer (p. 15) side bar.

View an Available Schema

1. In the AWS Explorer, 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 a schema in the editor, open the context menu of the schema, and then choose View Schema.

Find an Available Schema

In the AWS Explorer, do one or more of the following:

• Begin typing the title of the schema you want to find. The AWS Explorer highlights the schema titles that contain a match. (A registry must be expanded for you to see the highlighted titles.)

• Open the context menu for Schemas, and then choose Search Schemas. Or expand Schemas, open the context menu for the registry that contains the schema you want to find, and then choose Search Schemas in Registry. In the EventBridge Schemas Search dialog box, begin typing the title of the schema you want to find. The dialog box displays the schema titles that contain a match.
To display the schema in the dialog box, select the title of the schema.

Generate Code for an Available Schema

1. In the **AWS Explorer**, 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 resulting wizard pages, choose the following:
   - The **Version** of the schema
   - The code binding language
   - The workspace folder where you want to store the generated code on your local development machine

**Working with AWS Step Functions**

The AWS Toolkit for Visual Studio Code (VS Code) provides support for **AWS Step Functions**. Using the Toolkit for VS Code, you can create, update and execute Step Functions state machines.

**Topics**
- Working with AWS Step Functions (p. 39)

**Working with AWS Step Functions**

You can use the AWS Toolkit for Visual Studio Code (VS Code) to perform various operations with **state machines**.

**Topics**
- Prerequisites (p. 40)
- Work with state machines in VS Code (p. 40)
- State machine templates (p. 43)
- State machine graph visualization (p. 43)
- Code snippets (p. 44)
• Code completion and validation (p. 45)

Prerequisites

• Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2), then install the toolkit.
• Ensure that you have configured your credentials before opening the AWS Explorer.

Work with state machines in VS Code

You can use VS Code to interact with remote state machines, and develop state machines locally. You can create or update state machines, list existing state machines, execute them, and download them. VS Code also lets you create new state machines from templates, see a visualization of your state machine, and provides code snippets, code completion, and code validation.

List existing state machines

If you've already created state machines, you can view a list of them:

1. Open the AWS Explorer.
2. Select Step Functions
3. Verify that it lists all the state machines in your account.

Download a state machine

To download a state machine:

1. In the AWS Explorer, right click the state machine that you want to download.
2. Select Download, then select the location where you want to download the state machine.
3. Verify that it downloaded correctly.
Create a state machine

You can create a new state machine yourself, or you can use a template. For more information on creating a state machine from a template, see the State Machine Templates section. To create a new state machine:

1. Create a new Amazon States Language (ASL) file with your state machine definition. Use the menu at the bottom right to set it as Amazon States Language.

2. Select Publish to Step Functions.

3. Select Quick Create, choose a role, and name your state machine.
Update a state machine

To update a state machine:

1. Edit the ASL file with your state machine definition.
2. Select Publish to Step Functions.
3. Select Quick Update, then select the state machine you want to update.

Execute a state machine

To execute a state machine:

1. In the AWS Explorer, right click the state machine that you want to execute.
2. Provide input for the execution. You can try both input from a file, and input in a text box.
3. Start the execution and verify that it runs successfully.
State machine templates

When you create a state machine, you have the option to create it from a template. The template contains a sample state machine definition with several commonly used states, and provides you with a starting point. To use state machine templates:

1. Open the Command Palette in VS Code.
2. Select AWS: Create a new Step Functions state machine.
3. Choose the template you want to use.

State machine graph visualization

Graph visualizations let you see what your state machine looks like in graphical format. When you create a graph visualization, another tab will open and display a visualization of the state machine JSON. You can then compare the state machine definition you are writing concurrently with its visualization. As you change your state machine definition, the visualization will be updated.

**Note**

To create a visualization of a state machine definition, the definition must be open in the active editor. If you close or rename the definition file, the visualization will close.

To create a state machine graph visualization:
1. Define your state machine.
2. Open the **Command Palette** in VS Code.
3. To create a visualization, use the visualization button in the upper right corner, or choose **AWS Preview state machine graph**.

### Code snippets

Code snippets let you insert short sections of code. To use code snippets:

1. Open a file and save it with the extension `.asl.json`.
2. Create a new state machine with the **States** property.
3. Place the cursor within **States**.
4. Use the key combination `Control + Space`, and select your preferred code snippet.
5. Use `Tab` to traverse the variable and parameters in the code snippet.
6. Test **Retry** and **Catch** snippets by placing the cursor within the related state.
Code completion and validation

To see how code completion works:

1. Create several states.
2. Place the cursor after a **Next**, **StartAt**, or **Default** property.
3. Use the key combination **Control + Space** to list available completions. Additional properties can be accessed using **Control + Space** again, and will be based on the **Type** of the **State**.
4. As you work, code validation will happen for:
   - Missing properties
   - Incorrect values
   - No terminal state
   - Nonexistent states that are pointed to
Working with Amazon Elastic Container Service

The AWS Toolkit for Visual Studio Code provides some support for Amazon Elastic Container Service (Amazon ECS). The Toolkit for VS Code assists you in certain Amazon ECS-related work, such as creating task definitions.

Topics

• Using IntelliSense for Amazon ECS task-definition files (p. 46)

Using IntelliSense for Amazon ECS task-definition files

One of the things that you might do when working with Amazon Elastic Container Service (Amazon ECS) is to create task definitions, as described in Creating a Task Definition from the Amazon Elastic Container Service Developer Guide. When you install the AWS Toolkit for Visual Studio Code, the installation includes IntelliSense functionality for Amazon ECS task-definition files.

Prerequisites

• Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2).

Use IntelliSense in Amazon ECS task-definition files

The following example shows you how you can take advantage of IntelliSense in Amazon ECS task-definition files.

1. Create a JSON file for your Amazon ECS task definition. The file's name must have `ecs-task-def.json` at the end, but can have additional characters at the beginning.

   For this example, create a file named `my-ecs-task-def.json`

2. Open the file in a VS Code editor and enter the initial curly braces.

3. Enter the letter "c" as if you wanted to add `cpu` to the definition. Observe the IntelliSense dialog that opens, which is similar to the following.
Working with the AWS CDK Explorer

The AWS CDK Explorer enables you to work with AWS Cloud Development Kit (AWS CDK) applications, or apps. You can find detailed information about the AWS CDK in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

AWS CDK apps are composed of building blocks known as constructs, which include definitions for your AWS CloudFormation stacks and the AWS resources within them. Using the AWS CDK Explorer, you can visualize the stacks and resources that are defined in AWS CDK constructs. This visualization is provided in a tree view in an Explorer pane within the Visual Studio Code (VS Code) editor. See a high-level view of the AWS CDK Explorer in the navigation (p. 16) topic.

This section provides information about how to access and use the AWS CDK Explorer in the VS Code editor. It assumes that you’ve already installed and configured (p. 2) the Toolkit for VS Code on your system.

Note
You can disable the AWS CDK Explorer so that it isn't displayed in the VS Code editor. In the File menu, choose Preferences, Settings. Then enter "cdk" into the Search box and clear the Enable the AWS CDK Explorer box.

Topics
- Working with AWS CDK applications (p. 47)

Working with AWS CDK applications

Use the AWS CDK Explorer in the AWS Toolkit for VS Code to visualize and work with AWS CDK applications.
Prerequisites

- Be sure your system meets the prerequisites specified in Installing the Toolkit for VS Code (p. 2).
- Install the AWS CDK command line interface, as described in the first few sections of Getting Started with the AWS CDK in the AWS Cloud Development Kit (AWS CDK) Developer Guide.

  **Important**
  The AWS CDK version must be 1.17.0 or later. Use `cdk --version` on the command line to see what version you’re running.

Visualize an AWS CDK application

Using the **AWS CDK Explorer**, you can create a CDK app or load an existing one. Then, you can visualize the stacks and resources that are defined in the CDK constructs of that app.

Perform the first several steps of the Hello World Tutorial in the AWS CDK Developer Guide. Stop when you get to the step called *Deploying the Stack*. You can run the commands provided in the tutorial, for example, `mkdir` and `cdk init`, on an operating system command line or in a Terminal window inside the VS Code editor.

After you complete that first part of the CDK tutorial, load the resulting folder into the VS Code editor if it isn’t already loaded. At the bottom of the **VS Code Explorer** side bar find **AWS CDK Explorer**, and then open the app’s tree view, as shown.
The tree view gives you a visual representation of the information in the `tree.json` file of the CDK app, which is created when you run the `cdk synth` command. The file is located in the app's `cdk.out` directory.

**Important notes**

- When you load CDK apps into the VS Code editor, you can load multiple folders at one time. Each folder can contain multiple CDK apps, as shown in the preceding image. The AWS CDK Explorer finds apps in the project root directory and its direct subdirectories.
• When you perform the first several steps of the tutorial, you might notice that the last command you execute is `cdk synth`, which generates the `tree.json` file. If you change aspects of a CDK app, for example, add more resources, you need to execute that command again to see the changes reflected in the tree view.

Perform other operations on an AWS CDK app

You can use the VS Code editor to perform other operations on a CDK app, just as you would by using the command line of your operating system or other tools. For example, you can update the code files in the editor and deploy the app by using a VS Code Terminal window.

To try out these types of actions, use the VS Code editor to continue the Hello World Tutorial in the AWS CDK Developer Guide. Be sure to perform the last step, Destroying the App's Resources, so that you don't incur unexpected costs to your AWS account.
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. 51)
• Identity and Access Management for this AWS Product or Service (p. 52)
• Logging and Monitoring in this AWS Product or Service (p. 52)
• Compliance Validation for this AWS Product or Service (p. 53)
• Resilience for this AWS Product or Service (p. 53)
• Infrastructure Security for this AWS Product or Service (p. 54)
• Configuration and vulnerability analysis in this AWS Product or Service (p. 54)

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.
Identity and Access Management

• 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.

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.

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.

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.

Logging and Monitoring in this AWS Product or Service

This AWS product or service provides status and results in the OUTPUT tab and the DEBUG CONSOLE tab. You can also view logs of this activity.

To view the logs of this AWS product or service

1. Open View, Command Palette.
2. Type "aws" into the find box and choose AWS: View AWS Toolkit Logs.
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 and AWS services that are in scope of AWS compliance efforts by compliance program.

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.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

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.

Configuration and vulnerability analysis in this AWS Product or Service

This AWS product or service is released to the Visual Studio Marketplace as new features or fixes are developed. These updates sometimes include security updates, so it's important to keep this AWS product or service up to date.

To verify that automatic updates for extensions are enabled

1. Open the Settings window by choosing File, Preferences, Settings.
2. Expand Features and choose Extensions.
3. Adjust the settings for your environment.

If you choose to disable automatic updates for extensions, be sure to check for updates to this AWS product or service at intervals that are appropriate for your environment.

Latest major documentation update: July 30, 2020

The following table describes important changes in each release of the AWS Toolkit for Visual Studio Code. For notification about updates to this documentation, you can subscribe to an RSS feed.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced local debugging of AWS Serverless applications (p. 23)</td>
<td>Added support for new local debugging features for AWS Serverless Applications.</td>
<td>July 30, 2020</td>
</tr>
<tr>
<td>Amazon S3 (p. 34)</td>
<td>Added support for Amazon S3.</td>
<td>July 30, 2020</td>
</tr>
<tr>
<td>AWS Step Functions support (p. 55)</td>
<td>Added support for AWS Step Functions.</td>
<td>March 31, 2020</td>
</tr>
<tr>
<td>Security Content (p. 51)</td>
<td>Added security content.</td>
<td>February 6, 2020</td>
</tr>
<tr>
<td>Working with Amazon EventBridge Schemas (p. 37)</td>
<td>Added support for Amazon EventBridge Schemas</td>
<td>December 1, 2019</td>
</tr>
<tr>
<td>AWS CDK Explorer (p. 47)</td>
<td>Preview release of the AWS CDK Explorer feature.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>Using an external credential process (p. 7)</td>
<td>Added information about using an external credential process to obtain AWS credentials.</td>
<td>September 25, 2019</td>
</tr>
<tr>
<td>Using IntelliSense for task-definition files (p. 46)</td>
<td>IntelliSense support was added for working with Amazon ECS task Definition files.</td>
<td>September 24, 2019</td>
</tr>
<tr>
<td>Installing the AWS Toolkit for VS Code (p. 2)</td>
<td>Added information about installing language SDKs to support various toolchains.</td>
<td>June 12, 2019</td>
</tr>
<tr>
<td>Configure your toolchain (p. 13)</td>
<td>Added information about configuring various toolchains.</td>
<td>June 12, 2019</td>
</tr>
</tbody>
</table>