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

What is the AWS Toolkit for Eclipse? ................................................................. 1
Additional documentation and resources .......................................................... 1
Getting Started .................................................................................................. 2
Set up the Toolkit ............................................................................................... 2
Prerequisites ...................................................................................................... 2
Install the AWS Toolkit for Eclipse ................................................................. 2
Upgrade the AWS Toolkit for Eclipse .............................................................. 3
Set up AWS Credentials .................................................................................. 3
Get your AWS access keys ............................................................................. 3
Add your AWS access keys to the AWS Toolkit for Eclipse ......................... 4
Using multiple AWS accounts with the AWS Toolkit for Eclipse ................. 5
Changing the AWS credentials file location .................................................... 5
Associate Private Keys with Your Amazon EC2 Key Pairs ......................... 6
AWS Toolkit for Eclipse Basics ........................................................................ 8
Building an AWS Java Application .................................................................. 8
Build and Run the Amazon Simple Queue Service Sample .......................... 8
Serverless Projects ........................................................................................ 10
Creating a Serverless Project ......................................................................... 10
Serverless Project Blueprints ......................................................................... 11
Serverless Project Structure .......................................................................... 12
Deploying a Serverless Project ....................................................................... 12
See Also .......................................................................................................... 13
Differentiating AWS Resources with Naming ................................................. 14
Working with AWS Services ............................................................................ 17
How to Access AWS Explorer ......................................................................... 17
Using Lambda with the AWS Toolkit for Eclipse ............................................ 19
Tutorial: How to Create, Upload, and Invoke an AWS Lambda Function ....... 20
AWS Lambda Interface Reference ................................................................. 29
The AWS CloudFormation Template Editor .................................................... 36
Adding and Accessing AWS CloudFormation Templates in Eclipse ............. 36
Deploying a AWS CloudFormation Template in Eclipse ............................... 38
Updating a AWS CloudFormation Template in Eclipse ............................... 41
Validating a AWS CloudFormation Template in Eclipse ............................... 43
Using DynamoDB with AWS Explorer ............................................................ 44
Creating an DynamoDB Table ......................................................................... 44
Viewing an DynamoDB Table as a Grid ....................................................... 45
Editing Attributes and Values ....................................................................... 46
Scanning an DynamoDB Table ....................................................................... 47
Launch an Amazon EC2 Instance from an Amazon Machine Image ............ 48
Managing Security Groups from AWS Explorer ............................................. 52
Creating a New Security Group ..................................................................... 52
Adding Permissions to Security Groups ......................................................... 53
Viewing and Adding Amazon SNS Notifications ......................................... 55
View an Amazon SNS Notification ............................................................... 55
Add an Amazon SNS Notification ................................................................ 56
Connecting to Amazon Relational Database Service (Amazon RDS) ............. 57
Identity and Access Management .................................................................... 59
About AWS Identity and Access Management .............................................. 59
Create an IAM User ....................................................................................... 60
Create an IAM Group .................................................................................... 61
Add an IAM User to an IAM Group ............................................................. 62
Manage Credentials for an IAM User ............................................................ 63
Create an IAM Role ....................................................................................... 67
Attach an IAM Policy to a User, Group, or Role ............................................ 70
What is the AWS Toolkit for Eclipse?

The **AWS Toolkit for Eclipse** is an open source plug-in for the Eclipse integrated development environment (IDE) that makes it easier for developers to develop, debug, and deploy Java applications that use Amazon Web Services. It enhances the Eclipse IDE with additional features:

- The AWS SDK for Java is included and managed by Maven when you create a new AWS project using the AWS Toolkit for Eclipse
- AWS Explorer, an interface to Amazon Web Services that allows you to manage your AWS resources from within the Eclipse environment.
- AWS Lambda Java project and Serverless Application Model (SAM) project blueprint creation, deployment and debugging
- AWS CodeCommit repository cloning
- Integration with AWS CodeStar
- AWS Elastic Beanstalk deployment and debugging
- An AWS CloudFormation template editor
- Support for multiple AWS accounts

**Important**

There is no charge for using the AWS Toolkit for Eclipse, however you may incur AWS charges for creating or using AWS chargeable resources, such as running Amazon EC2 instances or using Amazon S3 storage. You can use the [AWS Pricing Calculator](https://aws.amazon.com/pricing/) to estimate charges for the use of various AWS resources.

Additional documentation and resources

In addition to this guide, there are a number of other resources available for AWS Toolkit for Eclipse users:

- AWS SDK for Java Developer Guide
- AWS SDK for Java API Reference
- Java developer blog
- Java developer forums
- GitHub:
  - documentation source
  - documentation issues
  - toolkit source
  - toolkit issues
- @awsforjava (Twitter)
- Toolkit license
- Toolkit FAQ
- Getting Started with the AWS SDK for Java
- Using AWS Elastic Beanstalk with the AWS Toolkit for Eclipse (video)
- AWS Toolkit for Eclipse: Amazon EC2 Management (video)
Getting Started

This section provides information for those getting started with the AWS Toolkit for Eclipse, including information about how to install and configure the AWS Toolkit for Eclipse.

Topics

- Set up the Toolkit (p. 2)
- Set up AWS Credentials (p. 3)
- Associate Private Keys with Your Amazon EC2 Key Pairs (p. 6)

Set up the Toolkit

This section describes how to install or upgrade the AWS Toolkit for Eclipse.

Prerequisites

The AWS Toolkit for Eclipse has the following prerequisites:

- An Amazon Web Services account—To obtain an AWS account, go to the AWS home page and click Sign Up Now. Signing up will enable you to use all of the services offered by AWS.
- A supported operating system—The AWS Toolkit for Eclipse is supported on Windows, Linux, macOS, or Unix.
- Java 1.8
- Eclipse IDE for Java Developers 4.2 or later—We attempt to keep the AWS Toolkit for Eclipse current with the default version available on the Eclipse download page.

  Note
  Eclipse provides a number of different downloads. We recommend installing the Eclipse IDE for Enterprise Java Developers, which includes the Eclipse Web Tools Platform required by Elastic Beanstalk, the Eclipse Data Tools Platform required for Amazon SimpleDB features, the Eclipse EGit, and the M2Eclipse. If you install another version of Eclipse, make sure that you have (or that you install, using the provided links) support for these features.

- (Optional) Google Android Development Tools (ADT)—if you want AWS Toolkit for Eclipse support for the AWS Mobile SDK for Android, you must install the ADT first.

Install the AWS Toolkit for Eclipse

To install the AWS Toolkit for Eclipse

1. Within Eclipse, click Help and then click Install New Software.
2. In the Work with box, type https://aws.amazon.com/eclipse and then press Enter.
3. Choose the components of the AWS Toolkit for Eclipse that you want to install. Click Select All to install all components at once.

  Note
  - AWS Toolkit for Eclipse Core (in the AWS Core Management Tools section) is required; all other components are optional.
  - Support for the AWS Mobile SDK for Android requires that you have the Google Android Developer Tools (ADT) for Eclipse installed first. If you have not yet installed the ADT, make sure that AWS SDK for Android is unchecked, or installation will fail.
• Support for the Amazon RDS or Amazon SimpleDB managers requires that the *Eclipse Data Tools Platform* (DTP) is installed. The DTP is installed by default with the "Java EE Developers" version of Eclipse, or can be installed separately.

4. Once you have made your selections, click **Next** (or **Finish**) to complete installation.

Once you have set up the AWS Toolkit for Eclipse you should **configure your AWS Credentials** (p. 3).

**Note**
Depending on the options selected, and on factors such as network speed, server latency and system capabilities, it may take up to 30 minutes for the installation to complete.

### Upgrade the AWS Toolkit for Eclipse

To upgrade or reinstall the AWS Toolkit for Eclipse, use the same instructions for **installing the toolkit** (p. 2).

Some versions of Eclipse, (notably **Mars** and **Neon**), may fail to fetch the latest artifacts due to a bug in old versions of the *Oomph plugin*. To work around this issue:

2. Delete the `~/.eclipse/org.eclipse.oomph.p2/cache/` directory to remove cached content.
3. Install the latest version of *Oomph* (Eclipse Installer).

### Set up AWS Credentials

To access Amazon Web Services with the AWS Toolkit for Eclipse, you must configure the AWS Toolkit for Eclipse with AWS account credentials.

### Get your AWS access keys

Access keys consist of an **access key ID** and **secret access key**, which are 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 IAM access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account.

**Note**
To create access keys, you must have permissions to perform the required IAM actions. For more information, see [Granting IAM User Permission to Manage Password Policy and Credentials](#) in the IAM User Guide.

### To get your access key ID and secret access key

1. Open the **IAM console**.
2. On the navigation menu, choose **Users**.
3. Choose your IAM user name (not the check box).
4. Open the **Security credentials** tab, and then choose **Create access key**.
5. To see the new access key, choose **Show**. Your credentials resemble the following:
   - Access key ID: AKIAIOSFODNN7EXAMPLE
   - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
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 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.*

Related topics


### Add your AWS access keys to the AWS Toolkit for Eclipse

The AWS Toolkit for Eclipse uses the same system for locating and using AWS access keys as that used by the AWS CLI and AWS Java SDK. Access keys entered in the Eclipse IDE are saved to a *shared AWS credentials file* (called *credentials*) in the `.aws` sub-directory within your home directory.

**Note**
The location of the credential file can be modified. For information about setting the location of this file, see Changing the AWS credentials file location (p. 5).

If you have already set your AWS credentials using the AWS CLI, then the AWS Toolkit for Eclipse will automatically detect and use those credentials. For more information about using the AWS CLI, see the AWS CLI User Guide.

**To add your access keys to the AWS Toolkit for Eclipse**

1. Open Eclipse’s **Preferences** dialog box and click **AWS Toolkit** in the sidebar.
2. Type or paste your AWS access key ID in the **Access Key ID** box.
3. Type or paste your AWS secret access key in the **Secret Access Key** box.
4. Click **Apply** or **OK** to store your access key information.

Here’s an example of a configured set of default credentials:
Using multiple AWS accounts with the AWS Toolkit for Eclipse

The Preferences dialog box allows you to add information for more than one AWS account. Multiple accounts can be useful, for example, to provide developers and administrators with separate resources for development and for release/publication.

Separate sets of AWS credentials are stored as profiles within the shared AWS credentials file described in Add your AWS access keys to the AWS Toolkit for Eclipse (p. 4). All of the configured profiles can be seen in the drop-down box at the top of the AWS Toolkit Preferences Global Configuration screen, labeled Default Profile.

To add a new set of access keys

1. On the AWS Toolkit Preferences screen in Eclipse's Preferences dialog box, click Add profile.
2. Add your new account information to the Profile Details section.
   - Choose a descriptive name for the Profile Name, and enter your access key information in the Access Key ID and Secret Access Key boxes.
3. Click Apply or OK to store your access key information.

You can repeat this procedure for as many sets of AWS account information that you need.

When you have entered all of your AWS account information, select the default account by choosing one of the accounts from the Default Profile drop-down. AWS Explorer displays resources associated with the default account, and when you create a new application through the AWS Toolkit for Eclipse, the application uses the credentials for the configured default account.

Note
For an alternative approach to separate your AWS resources, see Differentiating AWS Resources with Naming (p. 14).

Changing the AWS credentials file location

Using the AWS Toolkit for Eclipse Preferences screen, you can change the location used by the Toolkit to store and load credentials.

To set the AWS credentials file location

- In the AWS Toolkit Preferences dialog, locate the Credentials file location section, and enter the pathname of the file where you would like your AWS credentials stored.
Important
It is strongly recommended that you don’t store your AWS credential information within any network-shared directory or within any source-control-managed projects. Always retain strict control of your AWS access keys!

Associate Private Keys with Your Amazon EC2 Key Pairs

The AWS Toolkit for Eclipse can obtain your Amazon EC2 key pairs from AWS. However, you will need to associate private keys to use them with the AWS Toolkit for Eclipse.

To view your Amazon EC2 key pairs in the AWS Toolkit for Eclipse and associate private keys with them

1. Open Eclipse’s Preferences dialog box and click the triangle next to AWS Toolkit in the sidebar to show additional categories of AWS Toolkit for Eclipse settings.

2. Select Key Pairs.

Eclipse displays a scrollable list of your key pairs. If a key pair has a red X next to it, you will need to associate a private key with the key pair to use it.

3. Right-click the key pair and, from the context menu, select Select Private Key File...
4. Navigate to the private key file and select it to associate it with your key pair.
AWS Toolkit for Eclipse Basics

This section provides information about how to accomplish common development tasks with the AWS Toolkit for Eclipse.

Topics
- Building an AWS Java Application (p. 8)
- Serverless Projects (p. 10)
- Differentiating AWS Resources with Naming (p. 14)

Building an AWS Java Application

In this section, we’ll use the AWS Toolkit for Eclipse to build and run a local Java application that accesses AWS resources.

The AWS Toolkit for Eclipse includes the AWS SDK for Java and a number of Java sample programs. The AWS Toolkit for Eclipse makes it easy to build and run any of these samples. To demonstrate how the AWS Toolkit for Eclipse can help you build and run AWS applications in Java, we’ll use the AmazonSimpleQueueService sample as an example. The AWS Explorer that is provided with the AWS Toolkit for Eclipse can be used to view the running Amazon SQS queue.

Note
The AWS SDK for Java samples are provided in the samples directory in the SDK download, and can also be viewed on GitHub. For more information about the AWS SDK for Java itself, view the AWS SDK for Java Developer Guide.

Build and Run the Amazon Simple Queue Service Sample

To build and run the Amazon Simple Queue Service sample

1. Click the AWS icon on the Eclipse toolbar, and then click New AWS Java Project.
2. In the dialog box that appears, type a name for the project in the Project name box and select Amazon Simple Queue Service Sample.
3. Click Finish.

4. The sample application appears in Project Explorer. Expand the tree view for this project.

5. Beneath the src node, double-click the SimpleQueueService.java source file to open it in the editor pane. Locate the following line:

   System.out.println("Receiving messages from MyQueue.\n");

6. Right-click in the left margin of the editor pane, and select Toggle Breakpoint.
7. Right-click the project node in Project Explorer—in our example, this would be the node named myJavaSqsApp—then click Debug As > Java Application.

8. In the Select Java Application dialog box, select the SQS application and then click OK.

9. When the application stops at the breakpoint, Eclipse will ask if it should switch to the Debug perspective. Click No (the Debug perspective does not include AWS Explorer).

10. Go to AWS Explorer and expand the Amazon SQS node.

11. Double-click MyQueue and view the contents of the queue that was created by the Java client application.

12. Press F8. The Java client application will continue running and terminate normally.

13. Refresh the view in AWS Explorer. You will see that the MyQueue queue is no longer present; the application deletes the queue before the application exits.

Note
If you run this sample application repeatedly, you should wait at least 60 seconds between subsequent runs. Amazon SQS requires that at least 60 seconds elapse after deleting a queue before creating a queue with the same name.

Serverless Projects

The AWS Toolkit for Eclipse includes a project creation wizard that you can use to quickly configure and create serverless projects that deploy on AWS CloudFormation and run Lambda functions in response to RESTful web requests.

Creating a Serverless Project

To create a serverless project

1. Select the AWS icon in the toolbar, and choose New AWS serverless project… from the menu that appears.
2. Enter a Project name.
3. Enter a Package namespace for your project. This will be used as the prefix for the source namespaces created for your project.
4. Choose either to Select a blueprint or to Select a serverless template file:

   Select a Blueprint
   Choose a pre-defined project blueprint (p. 11) to use for your serverless project.
Select a Serverless Template File

Choose a JSON-formatted Serverless Application Model (SAM) `.template` file on your filesystem to fully customize your serverless project.

**Note**
For information about the structure and contents of a `.template` file, view the current version of the specification on GitHub.

5. Press the **Finish** button to create your new serverless project.
article

This blueprint creates a S3 Bucket for storing article content, and a DynamoDB Table for article metadata. It contains Lambda functions for retrieving (GetArticle) and storing (PutArticle) articles, which are triggered by API Gateway events.

hello-world

A simple blueprint that creates a Lambda function which takes a single string. Its output is Hello, value, where value is the string that was passed in, or World if no string is passed to the function.

Serverless Project Structure

The serverless project wizard will create a new Eclipse project for you, consisting of the following parts:

- The src directory contains two sub-directories, each prefaced with your chosen Package namespace:
  - mynamespace.function
    Contains class files for the Lambda functions that are defined by your serverless template.
  - mynamespace.model
    Contains generic ServerlessInput and ServerlessOutput classes that define the input and output model for your Lambda functions.

**Note**
For more information about the input and output formats used in the model classes, see the Configure Proxy Integration for a Proxy Resource page in the API Gateway Developer Guide.

- The serverless.template file defines the AWS resources and Lambda functions (a resource of type "AWS::Serverless:Function") used by your project.

Deploying a Serverless Project

**To deploy your serverless project**

1. In Eclipse's Project Explorer window, select your project and open the context menu (right-click or long press).
2. Choose Amazon Web Services • Deploy Serverless Project... on the context menu. This will bring up the Deploy Serverless to AWS CloudFormation dialog.
3. Select the AWS Regions to use. This determines where the AWS CloudFormation stack that you deploy is located.
4. Choose an S3 Bucket to use to store your Lambda function code, or select the Create button to create a new S3 bucket to store your code.
5. Choose a name for your AWS CloudFormation stack.
6. Press the Finish button to upload your Lambda functions to Amazon S3 and deploy your project template to AWS CloudFormation.
When your project is deployed, a AWS CloudFormation stack detail window will appear that provides information about your deployment and its current status. It will initially show its status as CREATE_IN_PROGRESS. When the status is CREATE_COMPLETE, your deployment is active.

To return to this window at any time, open the AWS Explorer, select the AWS CloudFormation node, and then select the name of the AWS CloudFormation stack you specified.

Note
If there was an error during deployment, your stack may be rolled back. See Troubleshooting in the AWS CloudFormation User Guide for information about how to diagnose stack deployment errors.

See Also

- AWS Serverless Application Model (GitHub)
- The AWS CloudFormation Template Editor (p. 36)
- Using Lambda with the AWS Toolkit for Eclipse (p. 19)
Differentiating AWS Resources with Naming

During development of new products or features, it is useful to keep AWS resources that are used for development separate from resources that are used for production. One approach to maintaining this separation was discussed in the Set up AWS Credentials (p. 3), that is, to use different accounts for development and production resources. That approach works especially well when using AWS Explorer, because AWS Explorer displays resources based on account credentials. This section will discuss an alternative approach in which a naming convention is used to differentiate between development and production resources—and in which support for the naming convention is implemented in code.

The basic idea is to distinguish your AWS resources, such as Amazon Simple Storage Service (Amazon S3) buckets or Amazon SimpleDB domains, by adding a designated string value to the resource name. For example, instead of naming your Amazon SimpleDB domain “customers”, you would name it “customers-dev” for development use or “customer-prod” for production use. However, an issue arises if you need to move development code into production. At that point, you would need to change all these strings, perhaps with a number of global search and replace operations; that could be tedious or error prone. A more efficient method would be to add support for the naming convention in the code.
The `StageUtils` class exposes the following method.

```java
public static String getResourceSuffixForCurrentStage()
```

The `getResourceSuffixForCurrentStage` method returns a string that corresponds to the "stage" in the software life cycle for which the resource is used, such as "dev" or "beta" or "prod". This string can then be appended to resource identifiers used in code. You can use `getResourceSuffixForCurrentStage` to construct resource names. For example, the following method, `getTopicName`, returns a unique name for an Amazon SNS topic. Notice how it embeds the return value from `getResourceSuffixForCurrentStage` in this name.

```java
private String getTopicName (Entry entry) {
    return "entry" + StageUtils.getResourceSuffixForCurrentStage() + "-" + entry.getId();
}
```

The value returned by `getResourceSuffixForCurrentStage` is retrieved from the Java system property, "application.stage". You can specify this value by setting the system property in the container configuration for AWS Elastic Beanstalk.

**Note**

In the AWS Toolkit for Eclipse, your AWS Elastic Beanstalk application needs to be up and running in order for you to access the container configuration. Changing and saving the configuration causes the application to automatically restart with the new configuration.

**To access the Container/JVM Options panel for your AWS Elastic Beanstalk application**

1. In **AWS Explorer**, expand the **AWS Elastic Beanstalk** node and your application node.
2. Beneath the application node, double-click your AWS Elastic Beanstalk environment.
3. At the bottom of the **Overview** pane, click the **Configuration** tab.
4. In the **Container** area, configure the container options.
5. In the **Additional Tomcat JVM command line options** box, specify the value for the application.stage system property by adding a `-D` command line option. For example, you could use the following syntax to specify that the string value should be "-beta".

```
-Dapplication.stage=beta
```

**Note** that `getResourceSuffixForCurrentStage` automatically prepends a hyphen character to whatever string value you specify.
6. After you have added the system property value, click the **File** menu, and then click **Save**. Eclipse will save the new configuration. The application should restart automatically. You can check the **Events** tab—at the bottom of the Eclipse editor—for the event that indicates that the new configuration was successfully deployed to the environment.

7. After the application restarts, expand the **Amazon SimpleDB** node in **AWS Explorer**. You should now see a new set of domains that use the string value that you specified.

![AWS Explorer](image)

**Note**
For more information about configuring the container, see *Creating and Deploying Java Applications on AWS Elastic Beanstalk* in the AWS Elastic Beanstalk Developer Guide.
Working with AWS Services

AWS Explorer gives you a view of, and allows you to manipulate, multiple Amazon Web Services simultaneously. This section provides information about how to access and use the AWS Explorer view in Eclipse.

It assumes that you've already installed (p. 2) the AWS Toolkit for Eclipse on your system.

Topics

- How to Access AWS Explorer (p. 17)
- Using Lambda with the AWS Toolkit for Eclipse (p. 19)
- The AWS CloudFormation Template Editor (p. 36)
- Using DynamoDB with AWS Explorer (p. 44)
- Launch an Amazon EC2 Instance from an Amazon Machine Image (p. 48)
- Managing Security Groups from AWS Explorer (p. 52)
- Viewing and Adding Amazon SNS Notifications (p. 55)
- Connecting to Amazon Relational Database Service (Amazon RDS) (p. 57)
- Identity and Access Management (p. 59)
- Debug Serverless Applications Using AWS SAM Local (p. 74)

How to Access AWS Explorer

To display AWS Explorer, click the AWS icon on the toolbar, and select Show AWS Explorer View.
AWS Icon Menu

**Note**  
If the AWS icon is not visible on the toolbar, click the **Window** menu, and then click **Open Perspective | Other**. Click **AWS Management** from the list of Eclipse perspectives.

You can expand each node in AWS Explorer to view resources on AWS that are associated with your account. For example, if you click the white triangle to the left of the **Amazon EC2** node, it will expand and display Amazon EC2 resources associated with your AWS account. The AWS Toolkit for Eclipse uses the AWS account that you configured in the Set up AWS Credentials (p. 3) to determine which resources to display.
If you select any of the subnodes to Amazon EC2, Eclipse will open a view with detailed information about those resources. For example, double-clicking **Instances** opens a view that lists information about each of your Amazon EC2 instances such as its public DNS name, availability zone, and launch time.

### Using Lambda with the AWS Toolkit for Eclipse

The AWS Toolkit for Eclipse provides support for authoring code for AWS Lambda. 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 S3, DynamoDB, Kinesis, Amazon SNS, and Amazon Cognito. For more information about Lambda, see the [AWS Lambda Developer Guide](#).

This section of the AWS Toolkit for Eclipse User Guide focuses on how you can use features of the AWS Toolkit for Eclipse to create, deploy and execute Lambda functions.

#### Topics
- Tutorial: How to Create, Upload, and Invoke an AWS Lambda Function (p. 20)
- AWS Lambda Interface Reference (p. 29)
Tutorial: How to Create, Upload, and Invoke an AWS Lambda Function

This tutorial guides you through the process of a typical AWS Lambda workflow, and provides you with first-hand experience using Lambda with the AWS Toolkit for Eclipse.

Important

The tutorial assumes that you have an AWS account, have already installed the AWS Toolkit for Eclipse (p. 2), and that you understand the basic concepts and features of Lambda. If you're unfamiliar with Lambda, learn more at the Lambda home page and in the AWS Lambda Developer Guide.

Create an AWS Lambda Project

To begin a Lambda project, you first implement the code as a method in a handler class. The AWS Toolkit for Eclipse provides a new project wizard to help you create a new handler class. The Lambda project is a Maven project that uses a POM.xml file to manage package dependencies. You can use the Maven command line tool for building, testing, and deploying your application. For more information about Maven, see the Maven project documentation.

To create an AWS Lambda project

1. On the Eclipse toolbar, open the Amazon Web Services menu (identified by the AWS homepage icon), and then choose New AWS Lambda Java project. Or on the Eclipse menu bar, choose File, New, AWS Lambda Java Project.

2. Add a Project name, Group ID, Artifact ID, and class name in the associated input boxes. The Group ID and Artifact ID are the IDs that identify a Maven build artifact. This tutorial uses the following example values:

   - Project name: HelloLambda
   - Group ID: com.example.lambda
   - Artifact ID: demo
   - Class name: Hello

   The Package Name field is the package namespace for the AWS Lambda handler class. The default value for this field is a concatenation of the Group ID and Artifact ID, following Maven project conventions. This field is automatically updated when the Group ID and Artifact ID fields are updated.

3. For Input Type, choose Custom. For information about each of the available input types, see New AWS Lambda Java Project Dialog (p. 29).

4. Verify that your entries look like the following screenshot (modify them if they are not), and then choose Finish.
As you type, the code in the **Source preview** changes to reflect the changes you make in the dialog box.

5. **After you choose Finish**, your project's directory and source files are generated in your Eclipse workspace. A new web browser window opens, displaying `README.html` (which was created for you in your project's root directory). `README.html` provides instructions to guide you through the next steps of implementing, testing, uploading, and invoking your new Lambda function. Read through it to gain some familiarity with the steps that are described here.
Next, you implement the function in the HelloLambda Java project that was just created for you in Eclipse.

**Implement the Handler Method**

You use the **Create New Project** dialog box to create a skeleton project. Now fill in the code that will be run when your Lambda function is invoked. (In this case, by a custom event that sends a String to your function, as you specified when setting your method’s input parameter.)

**To implement your Lambda handler method**

1. In the Eclipse **Project Explorer**, open Hello.java in the HelloLambda project. It will contain code similar to the following.

```java
package com.example.lambda.demo;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Hello implements RequestHandler<Object, String> {
    @Override
    public String handleRequest(Object input, Context context) {
        context.getLogger().log("Input: " + input);
        // TODO: implement your handler
        return "Hello from Lambda";
    }
}
```

2. Replace the contents of the `handleRequest` function with the following code.

```java
@Override
public String handleRequest(String input, Context context) {
    context.getLogger().log("Input: " + input);
    String output = "Hello, " + input + "!";
    return output;
}
```

**Allow Lambda to Assume an IAM Role**

For Lambda to be able to access your Lambda function, you have to create an IAM role that gives it access to your AWS resources. You can create the role in two ways, either through the AWS Management Console or by using the AWS Toolkit for Eclipse. This section describes how to create the IAM role in the console. See **Upload the Code (p. 23)** to create one using the AWS Toolkit for Eclipse.

**To create an IAM role for Lambda**

1. Sign in to the AWS Management Console.
2. From the **Services** menu, open the IAM console.
3. In the Navigation pane, choose **Roles**, and then choose **Create role**.
4. For **Select type of trusted entity**, choose **AWS service**, and then choose **Lambda** for the service that will use this role. Then choose **Next: Permissions**.
5. For **Attach permissions policy**, choose **AWSLambdaBasicExecutionRole**. This allows Lambda to write to your CloudWatch Logs resources. Then choose **Next: Review**.
6. Add a name for your role, such as hello-lambda-role, and a description for the role. Then choose Create role to finish creating the IAM role.

Create an Amazon S3 Bucket for Your Lambda Code

AWS Lambda requires an Amazon S3 bucket to store your Java project when you upload it. You can either use a bucket that already exists in the AWS Region in which you'll run your code, or you can create a new one specifically for Lambda to use (recommended).

You can create an Amazon S3 bucket in two ways, either through the AWS Management Console or by using the AWS Toolkit for Eclipse. This section describes how to create an Amazon S3 bucket in the console. See Upload the Code (p. 23) to create one using the AWS Toolkit for Eclipse.

To create an Amazon S3 bucket for use with Lambda

1. Sign in to the AWS Management Console.
2. From the Services menu, open the S3 console.
3. Choose Create bucket.
4. Enter a bucket name, and then choose a region for your bucket. This region should be the same one in which you intend to run your Lambda function. For a list of regions supported by Lambda see Regions and Endpoints in the Amazon Web Services General Reference.
5. Choose Create to finish creating your bucket.

Upload the Code

Next, you upload your code to AWS Lambda in preparation for invoking it using the AWS Management Console.

To upload your function to Lambda

1. Right-click in your Eclipse code window, choose AWS Lambda, and then choose Upload function to AWS Lambda.
2. On the Select Target Lambda Function page, choose the AWS Region to use. This should be the same region that you chose for your Amazon S3 bucket (p. 23).
3. Choose **Create a new Lambda function**, and then type a name for your function (for example, HelloFunction).

4. Choose **Next**.

5. On the **Function Configuration** page, enter a description for your target Lambda function, and then choose the IAM role and Amazon S3 bucket that your function will use.
6. On the **Function Configuration** page, choose **Create** in **Function Role** if you want to create a new IAM role for your Lambda function. Enter a role name in the dialogue box the **Create Role** dialogue box.

For more information about the available options, see **Upload Function to AWS Lambda Dialog Box** (p. 31).
7. On the **Function Configuration** page, choose **Publish new version** if you want the upload to create a new version of the Lambda function. To learn more about versioning and aliases in Lambda, see [AWS Lambda Function Versioning and Aliases](https://docs.aws.amazon.com/lambda/latest/dg/versioning-aliases.html) in the AWS Lambda Developer Guide.

8. If you chose to publish a new version, the **Provide an alias to this new version** option is enabled. Choose this option if you want to associate an alias with this version of the Lambda function.

9. On the **Function Configuration** page, choose **Create** in the **S3 Bucket for Function Code** section if you want to create a new Amazon S3 bucket for your Lambda function. Enter a bucket name in the **Create Bucket** dialogue box.

10. In the **S3 Bucket for Function Code** section, you can also choose to encrypt the uploaded code. For this example, leave **None** selected. To learn more about Amazon S3 encryption, see [Protecting Data Using Server-Side Encryption](https://docs.aws.amazon.com/AmazonS3/latest/dev/server-side-encryption.html) in the Amazon S3 Developer Guide.

11. Leave the **Advanced Settings** options as they are. The AWS Toolkit for Eclipse selects default values for you. Choose **Finish** to upload your Lambda function to AWS.

If the upload succeeds, you will see the Lambda function name that you chose appear next to your Java handler class in the **Project Explorer** view.
Invoke the Lambda Function

You can now invoke the function on AWS Lambda.

To invoke your Lambda function

1. Right-click in the Eclipse code window, choose **AWS Lambda**, and then choose **Run Function on AWS Lambda**.
2. Choose the handler class you want to invoke.
3. In the input box, type a valid JSON string, such as “AWS Lambda”.

If you don’t see this happen, open the Eclipse **Error Log** view. Lambda writes information about failures to upload or run your function to this error log so you can debug them.
Note
You can add new JSON input files to your project, and they will show up in this dialog box if the file name ends with .json. You can use this feature to provide standard input files for your Lambda functions.

4. The **Show Live Log** box is checked by default. This displays the logs from the Lambda function output in the Eclipse **Console**.

5. Choose **Invoke** to send your input data to your Lambda function. If you have set up everything correctly, you should see the return value of your function printed out in the Eclipse **Console** view (which automatically appears if it isn't already shown).

Congratulations, you've just run your first Lambda function directly from the Eclipse IDE!

**Next Steps**

Now that you've uploaded and deployed your function, try changing the code and rerunning the function. Lambda automatically reuploads and invokes the function for you, and prints output to the Eclipse **Console**.

**More Info**

For more information about each of the pages that were covered in this tutorial, as well as a full description of each option, see the AWS Lambda Interface Reference (p. 29).
For more information about Lambda and about writing Java code for Lambda, see Authoring Lambda Functions in Java in the AWS Lambda Developer Guide.

AWS Lambda Interface Reference

This section provides detailed information about each of the user interface elements added to Eclipse by the AWS Toolkit for Eclipse for AWS Lambda.

Topics
- New AWS Lambda Java Project Dialog (p. 29)
- Upload Function to AWS Lambda Dialog Box (p. 31)
- Run AWS Lambda Function Dialog (p. 35)

New AWS Lambda Java Project Dialog

The New Lambda Java Project dialog helps you to create and configure a new Java project that you can use to author a Lambda function.

Launching the dialog

The New Lambda Java Project dialog can be launched in the following ways:

- by opening the AWS menu in the Eclipse toolbar and selecting New AWS Lambda Java project....
- by selecting File • New • Other... in the Eclipse menu, and then choosing AWS • AWS Lambda Java Project in the resulting dialog.
Create Project Dialog user interface

**Project name**

*Required.* You must provide a name for your project.

**Package name**

An optional name for your Java package. It must be a valid Java package name, such as “com.mycompany.myproject”. When you enter the package name in the text entry field, it will be added to the contents of the **Source Preview** window.

*Default:* None, this parameter is optional.

**Class name**

*Required.* The name that identifies the Java class that contains your Lambda code. It must be a valid Java class name. The default value is generic; you can specify your own name here or change the **Package name** to avoid conflicts with similarly-named classes.

*Default:* `LambdaFunctionHandler`
Input type

Required. The type of input that will be used to call your Lambda function. You can select a category from the drop-down list:

- **S3 Event**– receives an event from Amazon S3 event.
- **SNS Event**– receives an event from Amazon SNS.
- **Kinesis Event**– receives an event from an Amazon Kinesis stream.
- **Cognito Event**– receives an event from Amazon Cognito.
- **Custom**– receives an event from custom code. If you set the input type to **Custom**, then you can also set the name of the custom input type in the box next to the type selection. By default, the generic **Object** type is used.

**Important**
The custom input type must be a valid Java class name, and not a primitive type such as `int`, `float`, and so on. You can use Java's standard boxed types (`Integer`, `Float`, ...) for these cases.

Use the **Custom** input type for setting up event sources such as the following:

- user applications
- mobile applications
- The AWS Management Console.
- The AWS CLI invoke command.

*Default: S3 Event*

Output type

The output type. This must be a valid Java object.

*Default: Object*

Upload Function to AWS Lambda Dialog Box

You use the **Upload Function to AWS Lambda** dialog box to create a Lambda function, and upload your code to run when the Lambda function is invoked.

Launching the Dialog Box

You can launch the **Upload Function to AWS Lambda** dialog box in two ways:

- Open the context menu for your AWS Lambda Java Project in the Eclipse **Project Explorer** view, and then choose **Amazon Web Services, Upload function to AWS Lambda**.
- Open the context menu in the code window for your Java class, and then choose **AWS Lambda, Upload function to AWS Lambda**.

The **Upload Function to AWS Lambda** dialog box has two pages:

- Select Target Lambda Function (p. 32)
- Function Configuration (p. 33)
Select Target Lambda Function Options

Select the Handler

(Required) The handler class that contains the Lambda function code you want to upload.

(Default) The most recently uploaded handler or the first one found if none were previously uploaded.

Select the AWS Region

(Required) The region in which you want to create your Lambda function.

(Default) The default AWS Management Console region for your AWS account.

Select or Create a Lambda Function

(Required) You must select whether to use an existing Lambda function from the drop-down list, or to create a new one by entering its name.

(Default) Create a new Lambda function

When you choose Next, the Function Configuration page opens.
Function Configuration Options

The page is divided into five sections, each with its own settings.

**Basic Settings**

This section shows the function name and enables you to add a text description.

**Name**

(Immutable) The name is determined by the name you chose on the **Select Target Lambda Function** page. You can’t modify it here, however, you can choose **Back** to re-enter it on the previous page.
Description

(Optional) A text description of the function.

(Default) The description is empty.

Function Role

In this section, you can select the IAM role to apply to the function. You can also create a new IAM role with the Create button. The IAM role you create through the AWS Toolkit for Eclipse is a basic role that provides access to Amazon S3. If you need more access to AWS resources, you must provide access to each of the services used in the AWS Management Console.

IAM Role

(Required) The role that Lambda uses to access your AWS resources during the execution of your function.

(Default) The first IAM role from your AWS account.

Function Versioning and Alias

In this section, you can publish a new version of your Lambda function and specify an alias for that version. To learn more about Lambda versioning and aliasing, see AWS Lambda Function Versioning and Aliases in the AWS Lambda Developer Guide.

Publish new version

(Default) Not selected. If you select this option, the upload creates a new version of the Lambda function instead of replacing it.

Provide an alias to this new version

(Default) Not selected. If you select this option, you can type in a new alias or use an existing one.

S3 Bucket for Function Code

In this section, you can set an Amazon S3 bucket for your Lambda function to use. You can also create a new bucket with the Create button and select settings to encrypt your Lambda function when it uploads to Amazon S3.

S3 Bucket

(Required) An Amazon S3 bucket that your function's code can use. Only buckets that are in the same region in which you will run the function are displayed here.

(Default) The first bucket in your list or the last bucket you uploaded your Lambda function to.

Encryption setting

(Default) None is selected. To learn more about Amazon S3 encryption, see Protecting Data Using Server-Side Encryption in the Amazon S3 Developer Guide.

Advanced Settings

This section contains settings that you might use less often. They can provide you with more control over your function's execution environment than the settings in the Function Execution section.

Memory (MB)

(Required) The number of megabytes of memory available to your Lambda function.
(Default) 512 MB.

**Timeout (s)**

(Required) The timeout, in seconds, after which the function is considered to have failed if it has finished execution.

(Default) 15 s.

**Run AWS Lambda Function Dialog**

The **Run Lambda Function** dialog provides a way for you to invoke a Lambda function directly from the Eclipse user interface.

**Launching the dialog**

The **Run Lambda Function** dialog can be launched in the following ways:

- by opening the context menu for your AWS Lambda Java Project in Eclipse's *Project Explorer* view, and selecting **Amazon Web Services > Run function on AWS Lambda**.
- by opening the context menu in the code window for your Java class and selecting **AWS Lambda > Run function on AWS Lambda**.

The Invoke Function dialog appears like this:

![Lambda Function Input](image)

**Options**

There are two ways to provide data to your function. Either one or the other is required.

- **Select one of the JSON files as input**– If you have any `.json` files attached to your project, you can select one of them from the list provided. Otherwise, this option will be greyed out.
• **Or enter the JSON input for your function**—You can directly enter valid JSON input for your function here. The type of data that you enter must match the input parameter of the Java method in your handler class.

Once you've made a selection and have provided your input data, you can click **Finish** to invoke your Lambda function, or click **Cancel** to exit the dialog without running anything.

# The AWS CloudFormation Template Editor

The AWS Toolkit for Eclipse includes a built-in AWS CloudFormation template editor. Among the features supported:

- The ability to create and update stacks directly from the Eclipse IDE from the currently-edited template.
- A JSON validator to help ensure that your template complies with JSON formatting and content rules.

**Topics**

- Adding and Accessing AWS CloudFormation Templates in Eclipse (p. 36)
- Deploying a AWS CloudFormation Template in Eclipse (p. 38)
- Updating a AWS CloudFormation Template in Eclipse (p. 41)
- Validating a AWS CloudFormation Template in Eclipse (p. 43)

# Adding and Accessing AWS CloudFormation Templates in Eclipse

**To add a CloudFormation template to your Eclipse project**

1. Locate the template you'd like to add to your project in your system's file manager, and drag the file into your project's **Package Explorer** window.
2. Choose how you would like to add the file to your project, and click OK.

To access a CloudFormation template in your Eclipse project
• Double-click the template name in Package Explorer to begin editing the file.

Note
Files that end with .template or .json will automatically use the AWS CloudFormation template editor. If your file is not automatically recognized as an AWS CloudFormation template, you can select the editor by right-clicking the file name in Package Explorer or by right-clicking in the editor window with the file loaded, selecting Open With, then CloudFormation Template Editor

Deploying a AWS CloudFormation Template in Eclipse

Note
Only files that end in .template can be launched from the Eclipse IDE. If your file ends with another extension, such as .json, you will need to rename it first with a .template extension to use this feature.

To deploy an CloudFormation template from Eclipse

1. With your AWS CloudFormation.template file open in the AWS CloudFormation template editor (see Adding and Accessing AWS CloudFormation Templates in Eclipse (p. 36) for more information), right-click on the open template and select Run on AWS, then Create Stack on the context menu.
2. In the Create New CloudFormation Stack dialog, enter your stack name in the Stack Name field. Your template file should be automatically chosen in the Template File field.

3. Choose any (or none) of the following options:

**SNS Topic**—choose an existing SNS topic from the list to receive notifications about the stack's progress, or create a new one by typing an email address in the box and clicking Create New Topic.

**Creation Timeout**—choose how long AWS CloudFormation should allow for the stack to be created before it is declared failed (and rolled back, unless the Rollback on failure option is unchecked).

**Rollback on failure**—if you want the stack to rollback (delete itself) on failure, check this option. Leave it unchecked if you would like the stack to remain active, for debugging purposes, even if it has failed to complete launching.

4. Click Next to continue with entering parameter values.

5. If your stack has parameters, you will enter values for them next. For parameters with a predefined list of possible responses, you can choose a value from the list provided.
6. Click **Finish** to begin launching your stack.

While your stack is being launched, you can view its status by double-clicking the stack name beneath the **CloudFormation** node in the **AWS Explorer** view, or by right-clicking the stack name and selecting **Open in Stack Editor** on the context menu.

**Note**
If you cannot see the stack you launched in **AWS Explorer**, you may need to manually refresh the view by clicking the **Refresh AWS Explorer** icon at the top of the **AWS Explorer** view.
Updating a AWS CloudFormation Template in Eclipse

To update an CloudFormation template from Eclipse

1. With your AWS CloudFormation .template file open in the AWS CloudFormation template editor (see Adding and Accessing AWS CloudFormation Templates in Eclipse (p. 36) for more information), right-click on the open template and select Run on AWS, then Update Stack on the context menu.

2. In the Update CloudFormation Stack dialog, select your stack name in the Stack Name field if it has not been automatically selected for you. Your template file should also be automatically chosen in the Template File field.

3. Choose any (or none) of the following options:
SNS Topic— choose an existing SNS topic from the list to receive notifications about the stack's progress, or create a new one by typing an email address in the box and clicking Create New Topic.

Creation Timeout— choose how long AWS CloudFormation should allow for the stack to be created before it is declared failed (and rolled back, unless the Rollback on failure option is unchecked.

Rollback on failure— if you want the stack to rollback (delete itself) on failure, check this option. Leave it unchecked if you would like the stack to remain active, for debugging purposes, even if it has failed to complete launching.

4. Click Next to continue with entering parameter values.

5. If your stack has parameters, you will enter values for them next. For parameters with a predefined list of possible responses, you can choose a value from the list provided.

6. Click Finish to begin updating your stack.

While your stack is being updated, you can view its status by double-clicking the stack name beneath the CloudFormation node in the AWS Explorer view, or by right-clicking the stack name and selecting Open in Stack Editor on the context menu.
Validating a AWS CloudFormation Template in Eclipse

To validate an CloudFormation template in Eclipse

- Perform either one of the following actions:
  - Right-click the template name in the Package Explorer view and click Validate on the context menu.
  - Right-click the template that you are editing in the editor pane and click Validate on the context menu.
Important
Your template will be validated for JSON correctness only; it will not be validated for CloudFormation correctness. A stack template validated in this way can still fail to launch or update.

Using DynamoDB with AWS Explorer

Amazon DynamoDB is a fast, highly scalable, highly available, cost-effective, non-relational database service. DynamoDB removes traditional scalability limitations on data storage while maintaining low latency and predictable performance. The AWS Toolkit for Eclipse provides functionality for working with DynamoDB in a development context. For more information, see DynamoDB on the AWS website.

In the AWS Toolkit for Eclipse, AWS Explorer displays all the DynamoDB tables associated with the active AWS account.

Creating an DynamoDB Table

Using the AWS Toolkit for Eclipse, you can create a new DynamoDB table.

To create a new table in AWS Explorer

1. In AWS Explorer, right-click Amazon DynamoDB, and then click Create Table. The Create New DynamoDB Table wizard appears.
2. Enter a table name in the Table name box.
3. Enter a primary hash key attribute in the Hash key attribute box, and select the hash key type from the Hash key type drop-down list. DynamoDB builds an unordered hash index using the primary key attribute and an optional sorted range index using the range primary key attribute. For more information about the primary hash key attribute, see Partitions and Data Distribution in the Amazon DynamoDB Developer Guide.
4. Optionally, specify a range primary key by selecting Use a range key. Enter a range key attribute in the Range key attribute box, and select a range key type from the Range key type drop-down list.
5. Specify the number of read capacity units in the Read capacity units box, and specify the number of write capacity units in the Write capacity units box. You must specify a minimum of 3 read capacity
units and 5 write capacity units. For more information about read and write capacity units, see Provisioned Throughput in the Amazon DynamoDB Developer Guide.

6. Click Finish to create the table. Click the refresh button in AWS Explorer to view your new table in the table list.

Creating a table

Viewing an DynamoDB Table as a Grid

To open a grid view of one of your DynamoDB tables, double-click the subnode in AWS Explorer that corresponds to the table. From the grid view, you can view the items, attributes, and values stored in the table. Each row corresponds to an item in the table. The table columns correspond to attributes. Each cell of the table holds the values associated with that attribute for that item.

An attribute can have a value that is a string or a number. Some attributes have a value that consists of a set of strings or numbers. Set values are displayed as a comma-separated list enclosed by square brackets.
Amazon DynamoDB Grid View

Editing Attributes and Values

The table grid view is editable; by double-clicking a cell, you can edit the values for the item’s corresponding attribute. For set-value attributes, you can also add or delete individual values from the set.

<table>
<thead>
<tr>
<th>Brand</th>
<th>BicycleType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand-Company C</td>
<td>Hybrid</td>
<td>205 Desc...</td>
</tr>
<tr>
<td>Brand-Company B</td>
<td>Road</td>
<td>203 Desc...</td>
</tr>
<tr>
<td>Brand-Company A</td>
<td>Road</td>
<td>202 Desc...</td>
</tr>
<tr>
<td>Mountain A</td>
<td>Road</td>
<td>201 Desc...</td>
</tr>
<tr>
<td>Brand-Company B</td>
<td>Mountain</td>
<td>204 Desc...</td>
</tr>
</tbody>
</table>

Cell editing in Amazon DynamoDB Grid View

The editing UI enables you not only to change the value of an attribute, but also to change the format of the value for the attribute—with some limitations. For example, any number value can be converted into a string value. If you have a string value, the content of which is a number, such as “125”, the editing UI enables you to convert the format of the value from string to number. Also, the editing UI enables you to convert a single-value to a set-value. However, you cannot generally convert from a set-value to a single-value; an exception is when the set-value has, in fact, only one element in the set.
Editing set values in Amazon DynamoDB Grid View

The **Edit Values** dialog box opens when you are editing a set of values. After editing the attribute value, click **Save set** to confirm your changes. If you want to discard your changes, click **Cancel**.

After confirming your changes, the attribute value is displayed in red. This indicates that the attribute has been updated, but that the new value has not been written back to the Amazon DynamoDB database. To write your changes back to DynamoDB, click **File**, and then click **Save**, or press from the keyboard. To discard your changes, click **Scan Table**, and when the Toolkit asks if you would like to commit your changes before the Scan, click **No**.

Scanning an DynamoDB Table

From the Toolkit, you can perform Scans on your DynamoDB tables. In a Scan, you define a set of criteria and the Scan returns all items from the table that match your criteria. Scans are expensive operations and should be used with care to avoid disrupting higher-priority production traffic on the table. For more recommendations on safely using the Scan operation, go to the **Amazon DynamoDB Developer Guide**.

**To perform a Scan on an Amazon DynamoDB table from AWS Explorer**

1. In the grid view, click **Add scan condition**. A UI appears that enables you to edit a new Scan clause.
2. In the Scan clause editor, specify the attribute to match against, how it should be matched (Begins With, Contains, etc.), what literal value it should match, and if the value is a string or a number.
3. Add more Scan clauses as needed for your search. The Scan will return only those items that match the criteria from all of your Scan clauses. Note that the Scan will perform a case-sensitive comparison when matching against string values.
4. On the button bar at the top of the grid view, click the green play button to run the scan.

To remove a Scan clause, click the red X to the left of each clause.
Launch an Amazon EC2 Instance from an Amazon Machine Image

Before launching an EC2 instance, you should create a security group that will permit network traffic that is appropriate to your application to connect to the instance. At a minimum, the security group should enable access on port 22, so that you can SSH into the EC2 instance. You may also want to create a keypair, although you can also create the keypair while going through the launch wizard. Finally, you should think about which instance type is appropriate to your application; the price for an EC2 instance is typically higher for the more powerful types of instances. You can find a list of instance types and pricing information on the EC2 Pricing page.

To launch an Amazon EC2 instance

1. In AWS Explorer, expand the Amazon EC2 node. Right-click the Amazon Machine Images (AMIs) subnode and select Open EC2 AMIs View.
2. Configure the AMIs view to show the AMI that we'll use in this example. In the filter box, type `start ebs`. This filters the list of AMIs to show only those AMIs with names that contains both "start" and "ebs".

Right-click the `amazon/getting-started-with-ebs` AMI and select **Launch** from the context menu.

3. In the **Launch EC2 Instance** dialog box, configure the AMI for your application.

   **Number of Hosts**
   
   Set this value to the number of EC2 instances to launch.

   **Instance Type**
   
   Select the type of the EC2 instance to launch. You can find a list of instance types and pricing information on the **EC2 Pricing** page.
**Availability Zone**

Select an availability zone (AZ) in which to launch the instance. Note that not all AZs are available in all regions. If the AZ that you select is not available, the Toolkit will generate a message saying that you need to select a different AZ. For more information about AZs, go to the Region and Availability Zone FAQ in the Amazon EC2 User Guide for Linux Instances.

**Key Pair**

A key pair is a set of public/private encryption keys that are used to authenticate you when you connect to the EC2 instance using SSH. Select a keypair for which you have access to the private key.

**Security Group**

The security group controls what type of network traffic the EC2 instance will accept. You should select a security group that will allow incoming traffic on port 22, i.e. the port that is used by SSH, so that you can connect to the EC2 instance. For information about how to create security groups using the Toolkit, see Managing Security Groups from AWS Explorer (p. 52)

**Instance Profile**

The instance profile is a logical container for an IAM role. When you select an instance profile, you associate the corresponding IAM role with the EC2 instance. IAM roles are configured with policies that specify access to particular AWS services and account resources. When an EC2 instance is associated with an IAM role, application software that runs on the instance runs with the permissions specified by the IAM role. This enables the application software to run without having to specify any AWS credentials of its own, which makes the software more secure. For in-depth information about IAM roles, go to Working with Roles in the IAM User Guide.

**User Data**

The user data is data that you provide to the application software that runs on your EC2 instance. The application software can access this data through the Instance Meta Data Service (IMDS).
Launching an AMI from AWS Explorer
Managing Security Groups from AWS Explorer

The AWS Toolkit for Eclipse enables you to create and configure security groups to use with Amazon Elastic Compute Cloud (Amazon EC2) instances. When you launch an Amazon EC2 instance, you need to specify an associated security group.

A security group acts like a firewall on incoming network traffic. The security group specifies what types of network traffic an Amazon EC2 instance will allow to be received. It can also specify that incoming traffic will be accepted only from certain IP addresses or only from other specified security groups.

Creating a New Security Group

In this section, we’ll create a new security group. Initially after creation, the security group will not have any permissions configured. Configuring permissions is handled through an additional operation.

To create a new security group

1. In AWS Explorer, beneath the Amazon EC2 node, right-click Security Groups, and then click Open EC2 Security Groups View.
2. Right-click in the left pane of the EC2 Security Groups tab, and then click New Group.
Adding Permissions to Security Groups

In this section, we'll add permissions to the new security group to allow other computers to connect to our Amazon EC2 instance using Secure Shell (SSH) protocol.

To add permissions to a security group

1. Right-click in the right pane of the EC2 Security Groups tab, and then click Add Permissions.

3. In the dialog box, enter a name and description for the new security group. Click OK.
Invoke Add Permissions UI

2. In the dialog box, select Protocol, port and network. Click TCP from the Protocol drop-down menu. Enter 22 for Port or Port Range. Port 22 is the standard port for SSH. The Network Mask box specifies the allowed source IP addresses in CIDR format; it defaults to 0.0.0.0/0, which specifies that the security group will allow a TCP connection to port 22 (SSH) from any external IP address.

You could also, for example, specify that connections should be allowed only from computers in your local computer's subnet. In this case, you would specify your local computer's IP address followed by a “/10”. For example, “xxx.xxx.xxx.xxx/10” where the “xxx” correspond to the distinct octet values that make up your local computer's IP address.

Click OK.

You could also set permissions to the security group by specifying a UserID and security group name. In this case, Amazon EC2 instances in this security group would accept all incoming network traffic from Amazon EC2 instances in the specified security group. It is necessary to also specify the UserID as a way
to disambiguate the security group name; security group names are not required to be unique across all
of AWS. For more information about security groups, see Network and Security in the Amazon EC2 User
Guide for Linux Instances.

Viewing and Adding Amazon SNS Notifications

You can use the AWS Toolkit for Eclipse to view Amazon Simple Notification Service (Amazon SNS)
topics associated with your application. Amazon SNS is a service that enables your application to send
notifications, using a protocol such as email, when specified events occur. To learn more about Amazon
SNS, see the Amazon SNS Developer Guide.

View an Amazon SNS Notification

The following process illustrates how to view an Amazon SNS notification.

To view a notification

1. In AWS Explorer, click the triangle to the left of the Amazon SNS node to expand it and see the
Amazon SNS topics it contains.

2. Double-click this SNS topic to open a detail view in the Eclipse editor pane. In this example, the
Subscription ARN column says that the topic is pending confirmation. Amazon SNS requires
a confirmation from the individual specified by the email address before SNS will send email
notifications to that individual.
Add an Amazon SNS Notification

You can add new Amazon SNS notifications through AWS Explorer.

**To add a new notification**

1. In **AWS Explorer**, right-click **Amazon SNS**, and then click **Create New Topic**. Enter a name for the new topic and click **OK**.

2. Double-click the new topic to display the detail view for the topic. Right-click in the **Subscriptions** area, and then click **Create Subscription**. Leave the **Subscription Protocol** box as **Email (plain text)** and enter an email address for the endpoint. Click **OK**. The detail view for the notification will now include this subscription.
Connecting to Amazon Relational Database Service (Amazon RDS)

In this section, we'll use the AWS Toolkit for Eclipse to connect to a database instance on the Amazon Relational Database Service (Amazon RDS). Before stepping through the process described below, you will need to have an RDS database instance associated with your AWS account. You can create a database instance on RDS using the AWS Management Console. When you create a database instance, set the TCP port that the database uses to receive connections to a value that is accessible from your location. For example, if you are behind a firewall, choose a TCP port to which your firewall allows connections. For more information, see the Amazon RDS User Guide.

1. In AWS Explorer, expand the Amazon RDS node. You should see a list of the database instances that are associated with your AWS account. Right-click one of these instances, and then click Connect.
2. The AWS Toolkit for Eclipse displays an authentication dialog box. Enter the master password that you specified when you created the database instance. Click **Finish**.

3. The AWS Toolkit for Eclipse brings up the connection to the database instance in the Eclipse Data Source Explorer. From here, you can inspect the structure and data in the database.
Identity and Access Management

AWS Identity and Access Management (IAM) lets you control who can access your AWS resources and what they can do with them. The AWS Explorer lets you create and manage IAM users, groups, and roles. You can also set a password policy for users, which lets you specify password requirements like minimum length, and lets you specify whether users are allowed to change their own passwords.

Note
It is a best practice for all users, even the account owner, to access AWS resources as IAM users. This ensures that if the credentials for one of the IAM users are compromised, the affected credentials can be revoked without needing to change the root credentials for the account.

About AWS Identity and Access Management

Instead of sharing the password and security credentials for your account (the access key ID and secret access key), you can create IAM users that can each have their own password and security credentials. You can then attach policies to users; in the policies you specify permissions that determine what actions a user can take and what resources the user is allowed access to.

For convenience, instead of adding policies to individual users, you can create IAM groups (for example, Admins and Developers) attach policies to them, and then add users to those groups. You can also create roles that have policies with permissions. Roles can be assumed by users who are in other accounts, by services, and by users who do not have an IAM identity. For more information about IAM, see the IAM User Guide.
Create an IAM User

You create IAM users so that others in your organization can have their own AWS identity. You can assign permissions to an IAM user either by attaching an IAM policy to the user or by assigning the user to a group. IAM users that are assigned to a group derive their permissions from the policies that are attached to the group. For more information, see Create an IAM Group (p. 61) and Add an IAM User to an IAM Group (p. 62).

Using the Toolkit, you can also generate AWS credentials (access key ID and secret access key) for the IAM user. For more information, see Manage Credentials for an IAM User (p. 63).

To create an IAM User

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Users node, and then select Create New Users.

2. In the Create New Users dialog box, enter up to five names for new IAM users, and then click Finish. For information about constraints on names for IAM users, see Limitations on IAM Entities in the IAM User Guide.
Create an IAM Group

You can add IAM users to groups in order to make it easier to manage permissions. Any permissions that are attached to the group apply to any users in that group. For more information about IAM groups, see Working with Users and Groups in the IAM User Guide.

When you create a group, you can create a policy that includes the permissions that members of the group will have.

To create an IAM group

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Groups node, and then select Create New Group.

2. Enter a name for the new IAM group and then click Next.

3. Enter a name for the policy that establishes what members of the group are allowed to do. Enter the policy as a JSON document, and then click OK.
Add an IAM User to an IAM Group

If an IAM user is added to a group, any policies that are attached to the group are also in effect for the user. For more information about IAM users, see Users and Groups in the IAM User Guide.

To add an IAM user to a IAM group

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Groups node, and then select Open Groups Editor. Note that you add IAM users to IAM groups from the Groups node in AWS Explorer rather than from the Users node.
2. In the Groups editor, select the group you want to add users to, and then click the Users tab.
3. On the right-hand side of the bottom pane, click the **Add Users** button.

4. In the **Add Users to Group** dialog box, select the users you want to add, and then click **OK**.

---

**Manage Credentials for an IAM User**

For each user, you can add a password. IAM users use a password to work with AWS resources in the AWS Management Console.
To create a password for an IAM user

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Users node, and then select Open Users Editor.
2. In the users listing, select the user you want to create a password for, and then click the Summary tab.
3. On the right-hand side of the bottom pane, click the Update Password button.
4. In the Update User Password dialog box, enter a password and then click OK.
   Note
   The new password will overwrite any existing password.

For each user you can also generate a set of access keys (an access key ID and a secret access key). These keys can be used to represent the user for programmatic access to AWS—for example, to use the AWS command-line interface (CLI), to sign programmatic requests using the SDK, or to access AWS services through the Toolkit. (For information about how to specify credentials for use with the Toolkit, see Set up AWS Credentials (p. 3).)

To generate access keys for an IAM user

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Users node, and then select Open Users Editor.
2. In the users listing, select the user you want to generate keys for, and then click the **Summary** tab.

![User management screenshot](image1.png)

3. Click the **Manage Access Keys** button.

   ![Manage Access Keys](image2.png)

   A window is displayed where you can manage access keys for the user.

4. Click the **Create Access Key** button.

   ![Create Access Key](image3.png)

   The **Manage Access Key** dialog box is displayed.
5. Click the **Download** button to download a comma-separated value (CSV) file that contains the credentials that were generated.

**Note**
This will be your only opportunity to view and download these access keys. If you lose these keys, you must delete them and create a new set of access keys.

You can generate only two sets of credentials per IAM user. If you already have two sets of credentials and you need to create an additional set, you must delete one of the existing sets first.

You can also deactivate credentials. In that case, the credentials still exist, but any requests to AWS that are made using those credentials will fail. This is useful if you want to temporarily disable access to AWS for that set of credentials. You can reactivate credentials that were previously deactivated.

**To delete, deactivate, or reactivate access keys for an IAM user**

1. In **AWS Explorer**, expand the **AWS Identity and Access Management** node, right-click the **Users** node, and then select **Open Users Editor**.
2. In the users listing, select the user you want to manage access keys for, click the **Summary** tab, and then click the **Manage Access Keys** button.
3. In the window that lists the access keys for that user, right-click the credentials you want to manage and then choose one of the following:
   - **Delete Access Key**
   - **Make Inactive**
   - **Make Active**
Create an IAM Role

Using the AWS Toolkit, you can create IAM roles. The role can then be assumed by entities that you want to allow access to your AWS resources. Policies that you attach to the role determine who can assume the role (the trusted entity or principal) and what those entities are allowed to do.

In the Toolkit, you can specify the following trusted entities:

- An AWS service. For example, you can specify that an Amazon EC2 can call other AWS services or that AWS Data Pipeline is allowed to manage Amazon EC2 instances. This is known as a service role.
- A different account that you own. If you have multiple AWS accounts, you might need to let users in one account use a role to get permissions to access resources that are in another account of yours.
- A third-party account. You might let a third-party vendor manage your AWS resources. In that case, you can create a role in which the trusted entity is the third party's AWS account.

After you specify who the trusted entity is, you can specify a policy that determines what the role is allowed to do.

For example, you could create a role and attach a policy to that role that limits access to only one of your Amazon S3 buckets. You can then associate the role with an Amazon EC2 instance. When an application runs on the Amazon EC2 instance, the application can access only the Amazon S3 bucket that you allowed access to in the role's policy.

For more information about IAM roles, see IAM Roles in the IAM User Guide.

To create an IAM role

1. In AWS Explorer, expand the AWS Identity and Access Management node, right-click the Roles node, and then select Create New Role.
2. Enter a name for the IAM role and then click **Next**.

3. Select the trusted entity for the role. To create a service role, select **AWS Service Roles** and then select a service role from the drop-down list.
To provide access for a user that's defined in a different AWS account that you own, select **Account ID** and enter the AWS account number of the other account.

To provide access for a third-party account, select **Account ID** and enter the third party's AWS account number. If the third party has provided you with an **external ID**, enter that as well.

4. Click **Next**.
5. Enter a name for the policy that establishes what the role is allowed to do. Then enter the policy as a JSON document, and click **OK**.
Attach an IAM Policy to a User, Group, or Role

The policy name must be unique within your account. The JSON that you enter for the policy must validate, or you will not be able to save the policy. For information about how to create a policy, see Overview of Policies in the Using IAM guide.

6. Click Finish.

The new IAM role appears in the Roles editor.

For examples that show how to access AWS using the IAM role associated with an Amazon EC2 instance, see Using IAM Roles to Grant Access to AWS Resources on Amazon EC2 in the AWS SDK for Java Developer Guide.

Attach an IAM Policy to a User, Group, or Role

Policies are documents that define permissions. For example, a policy that’s attached to a user can specify what AWS actions the user is allowed to call and what resources the user is allowed to perform the actions on. If the policy is attached to a group, the permissions apply to users in the group. If the policy is attached to a role, the permissions apply to whoever assumes the role.

The process for attaching a policy to a user or group is similar. For roles, you can attach a policy that specifies what the role is allowed to do. You use a separate process to attach or edit the policy that determines who is allowed to assume the role (that is, to manage the trust relationship.)
To create an IAM policy for a user, group, or role

1. In AWS Explorer, expand the AWS Identity and Access Management node and then double-click the Groups node, the Users node, or the Roles node.
2. Select the group, user, or role you want to attach the policy to, and then click the Permissions tab.
3. On the right-hand side of the bottom pane, click the Attach Policy button.
4. In the Manage Group Policy, Manage User Policy, or Manage Role Permissions dialog box, enter a name for the policy. Then enter the policy as a JSON document, and click OK.
Attach an IAM Policy to a User, Group, or Role

The policy name must be unique within your account. The JSON that you enter for the policy must validate, or you will not be able to save the policy. For information about how to create a policy, see Overview of IAM Policies in the IAM User Guide.

To create or manage a trust relationship for a role

1. In AWS Explorer, expand the AWS Identity and Access Management node and then double-click the Roles node.
2. In the Roles editor, select the role you want to manage, and then click the Trust Relationships tab.
3. On the right-hand side of the bottom pane, click the **Edit Trust Relationship** button.

4. In the **Edit Trust Relationship** dialog box, edit the JSON policy document and then click **OK**.

---

**Set Password Policy**

In the AWS Toolkit for Eclipse you can set a password policy for your account. This lets you make sure that passwords that are created for IAM users follow certain guidelines for length and complexity. It also lets you specify whether users are allowed to change their own passwords. For more information, see [Managing an IAM Password Policy](#) in the IAM User Guide.

**To create an IAM policy for a user or group**

1. In **AWS Explorer**, under **Identity and Access Management**, double-click the **Password Policy** node.
2. In the **Password Policy** pane, specify the policy options that you want for your AWS account, and then click **Apply Password Policy**.
Debug Serverless Applications Using AWS SAM Local

This tutorial guides you through debugging a serverless application project with the AWS Toolkit for Eclipse using AWS SAM Local. SAM Local is the AWS CLI tool for managing serverless applications written with the AWS Serverless Application Model (AWS SAM). See the SAM Local README for more information.

Prerequisites

To use this tutorial, you must have the AWS Toolkit for Eclipse, Docker, and AWS SAM Local installed. See the AWS SAM Local README for Docker and SAM Local installation instructions. See the Getting Started (p. 2) topic for instructions on installing and setting up the AWS Toolkit for Eclipse.

Note

To use the AWS SAM Local feature of the AWS Toolkit for Eclipse, your project must be a valid Maven Project with a valid pom.xml file.

After you install the required tools, open the Eclipse Preferences dialog box from the Eclipse menu. Configure the SAM Local Executable path, as shown. This enables the AWS Toolkit for Eclipse to know where to find your SAM Local installation.
Import the SAM Application from AWS CodeStar

For this tutorial, you need a sample project in AWS CodeStar. See the Creating a Serverless Project in AWS CodeStar tutorial in the AWS CodeStar User Guide to create a sample project.

To import SAM app from AWS CodeStar

1. On the Eclipse toolbar, open the Amazon Web Services menu (identified by the AWS homepage icon), and then choose Import AWS CodeStar Project. Or, on the Eclipse menu bar, choose File, Import, AWS, AWS CodeStar Project.
2. Choose the region that the sample application was created in.
3. Choose your sample project from the Project Name list.
4. Add in your Git credentials. See the AWS CodeCommit User Guide to learn how to get Git credentials for CodeCommit.
5. Choose **Next**.
6. Choose **Next** on the **Branch Selection** page.
7. Choose **Finish** on the **Local Destination** page.

Next, you can debug this serverless application locally using SAM Local within Eclipse.

**Debug Lambda Function Locally**

Create a debug configuration for your serverless application and use SAM Local to run the application locally.
To debug the Lambda function locally

1. In the Eclipse Project Explorer, open HelloWorldHandler.java.
2. Right-click in your Eclipse code window, choose Debug As, and then choose AWS SAM Local.

![Debug Configuration](image-url)
3. For this example, leave the Project and Template as they are.
4. Choose Lambda Function in the Run as field.
6. For this example, we will provide an Amazon S3 event. Choose Generate next to the Event input box.

7. Choose a region that has your Amazon S3 bucket.
8. Enter a valid Amazon S3 bucket name.
9. Enter a valid Amazon S3 object key, and then choose OK.
10. On the Save As page, select the current project and enter a name for the event file. In this example, we used s3-event.json.
11. Choose OK to save the event file and get back to the main dialog box.
12. Leave the advanced settings as they are. See Advanced Settings (p. 81) to learn more about those fields.
13. Choose Apply, and then choose Debug.

This runs the Lambda function locally. You can set breakpoints as you would for other applications to debug the code.

**Test API Gateway Locally**

You can also test the HTTP request/response functionality with SAM Local.

**To test API Gateway locally**

1. Right-click in your Eclipse code window, choose Debug As, Debug Configuration.
2. Create a new Debug Configuration for this run and name it something different.

3. Choose API Gateway in the Run as field.

4. Leaving all other fields as they are, your configuration should look similar to the following.

5. Choose Apply, and then choose Debug.
This spawns a local API gateway that you can use to test your application. The debug output will contain HTTP links that can be used to verify the request/response functionality of your code.

Advanced Settings

This section describes the advanced options available on the SAM Local Debug configurations page.
AWS Configuration

Select profile

(Required) The profile to use for AWS credentials.
( default) The default profile

Select region

(Required) The region that the application is deployed to.
( Default) US East (Virginia)

SAM Local Configuration

Maven goals

(Required) Maven goals to execute when building the application. You must customize these goals if the default does not generate a Jar file with all the dependencies included (fat Jar). See Maven Shade Plugin in Maven Project to learn how to use the plugin to create a fat Jar.

( Default) clean package

SAM runtime

(Required) Path to the SAM executable.

( Default) /usr/local/bin/sam

Debug port

(Required) Port that the Eclipse debugger uses to connect to SAM Local.

( Default) 5858

Env vars

(Optional) Path to a JSON file that contains values for environment variables used by Lambda functions. See Environment variable files in the SAM Local user guide to learn the required syntax for this file.

( Default) Empty

Lambda Function Configuration

Code URI

(Optional) Path to the code archive file. For the example on this page, it would be the path to the .jar file.

( Default) Path in the template.yml file

Timeout

(Required) Lambda function runtime timeout.

( Default) 300
More Info

For more information about AWS SAM Local, see the AWS SAM Local user guide in GitHub. For more information about the AWS Serverless Application Model (SAM), see the AWS SAM project in GitHub.
Trouble Shooting

AWS CodeCommit plugin - Eclipse was unable to write to the secure store.

*Problem:* when checking out or checking in an AWS CodeCommit repository, I got an error saying *Writing to secure store failed, No password provided.*

Document History

The following table describes the important changes since the last release of the AWS Toolkit for Eclipse User Guide.

API version: 2010-12-01

Last documentation update: Apr 28, 2020

Dec 01, 2016

Added a new section that provides detail about the new serverless project wizard (p. 10).

Dec 22, 2015

Removed the Additional Resources topic—the information from that page is now available on the first page of the guide, under the heading Additional documentation and resources (p. 1).

Oct 22, 2015

• The guide has been renamed from “Getting Started Guide” to “User Guide”, to better represent its function.
• Installation instructions have been updated to compensate for changes in the way you select components of the toolkit to install.

June 16, 2014

The AWS Toolkit for Eclipse now provides support for authoring AWS Lambda functions with Java. For more information, see Using Lambda with the AWS Toolkit for Eclipse (p. 19).

September 27, 2013

• The AWS Toolkit for Eclipse now uses the same system for storing and accessing AWS credentials as the AWS CLI and AWS SDKs, which includes the ability to use multiple profiles to store more than one set of credentials. For information, see the newly-updated topic: Set up AWS Credentials (p. 3).
• The AWS Toolkit for Eclipse Getting Started Guide has been restructured in alignment with other AWS SDK Documentation (most notably, the AWS Java SDK upon which the AWS Toolkit for Eclipse depends). Much of the restructuring should be logical and self-evident, but a description of each of the guide’s major sections is provided in What is the AWS Toolkit for Eclipse? (p. 1).
• Getting Started (p. 2) has been updated for Eclipse 4.3 (“Kepler”).

September 9, 2013

This topic tracks recent changes to the AWS Toolkit for Eclipse User Guide. It is intended as a companion to the release notes history.