AWS Toolkit for Eclipse User Guide

Release 1.0

Amazon Web Services

Jul 19, 2016

Contents

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 installed as part of 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 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 Simple Monthly Calculator to estimate charges for the use of various AWS resources.

1.1 Additional documentation and resources

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

- AWS Java Developer Guide

- AWS SDK for Java Reference

- Java developer blog

- Java developer forums

- GitHub:

    - documentation source

    - documentation issues

- – toolkit source

- – toolkit issues

- @awsforjava (Twitter)

- license

- Toolkit FAQ

- Getting Started with the AWS SDK for Java

- Installing the AWS Toolkit for Eclipse (video)

- Using AWS Elastic Beanstalk with the AWS Toolkit for Eclipse (video)

- Amazon SimpleDB Management in 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.

2.1 Set up the Toolkit

- *Prerequisites*
- *Install the AWS Toolkit for Eclipse*

2.1.1 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, OS X or unix.

- *Java 1.6 or later*

- *Eclipse IDE for Java Developers 3.6 or later* – We attempt to keep the AWS Toolkit for Eclipse current with the default version available on the Eclipse download page.

Tip: Eclipse provides a number of different downloads. We recommend installing the *Eclipse IDE for Java EE Developers*, which includes the Eclipse Web Tools Platform required by Elastic Beanstalk, and the Eclipse Data Tools Platform required for Amazon SimpleDB features. 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 SDK for Android, you must install the ADT first.

2.1.2 Install the AWS Toolkit for Eclipse

To install the AWS Toolkit for Eclipse

1. Start Eclipse.

2. Click *Help* and then click *Install New Software*.

3. In the *Work with* box, type `https://aws.amazon.com/eclipse` and then press `Enter`.

4. Choose the components of the AWS Toolkit for Eclipse that you want to install. If you want to install *all* components, click *Select All*. Also note:

   - *AWS Toolkit for Eclipse Core* (in the *AWS Core Management Tools* section) is **required**; all other components are optional.

   - Support for the AWS 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.

5. 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*.

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.

2.2 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.

Contents

- *Get your AWS access keys*
- *Add your AWS access keys to the AWS Toolkit for Eclipse*
- *Using multiple AWS accounts with the AWS Toolkit for Eclipse*
- *Changing the AWS credentials file location*

2.2.1 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. From the navigation menu, click *Users*.

3. Select your IAM user name.

4. Click *User Actions*, and then click *Manage Access Keys*.

5. Click *Create Access Key*.

    Your keys will look something like this:

    - Access key ID example: `AKIAIOSFODNN7EXAMPLE`

    - Secret access key example:
      `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

6. Click *Download Credentials*, and store the keys in a secure location.

Important: Your secret key will no longer be available through the AWS Management Console; you will have the only copy. Keep it confidential in order to protect your account, and never email it. Do not share it 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

- What Is IAM? in IAM User Guide.

- AWS Security Credentials in Amazon Web Services General Reference.

2.2.2 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*.

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:
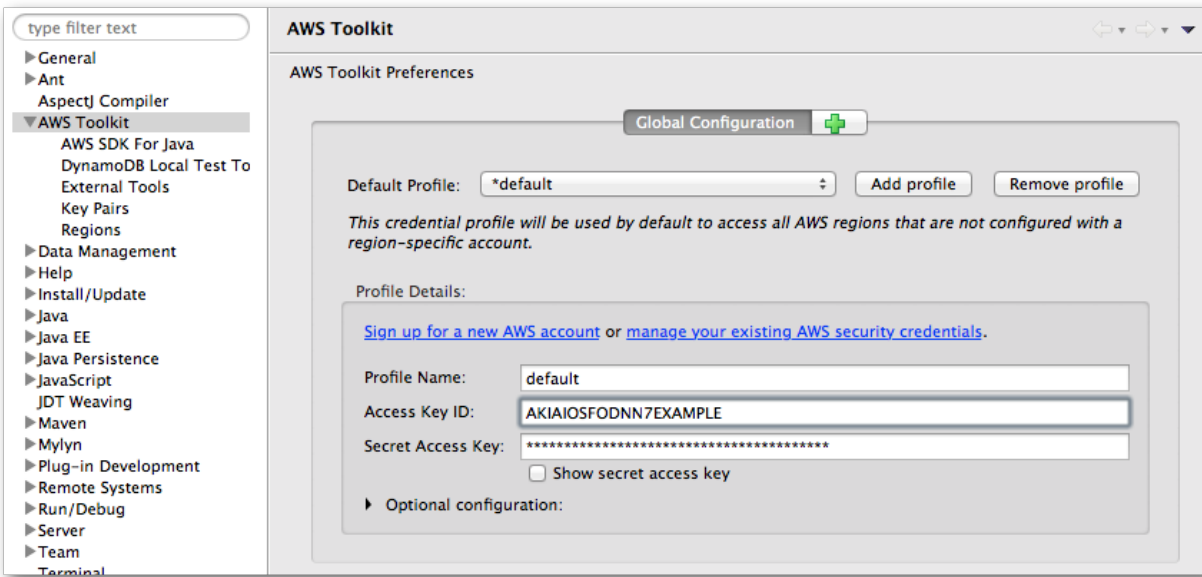
2.2.3 Using multiple AWS accounts with the AWS Toolkit for Eclipse

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*. 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.
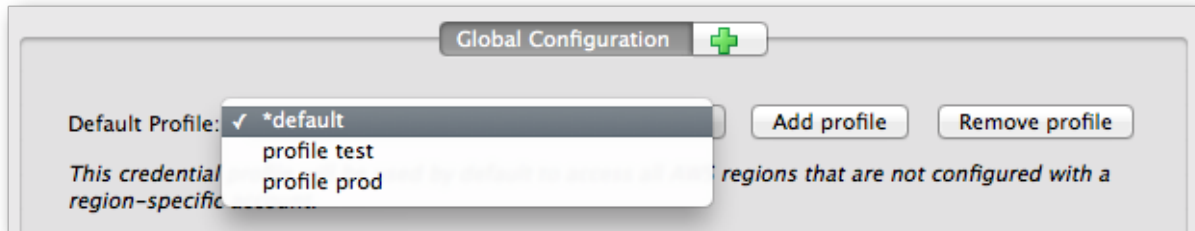
Note: For an alternative approach to separate your AWS resources, see *Differentiating AWS Resources with Naming*.

2.2.4 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.
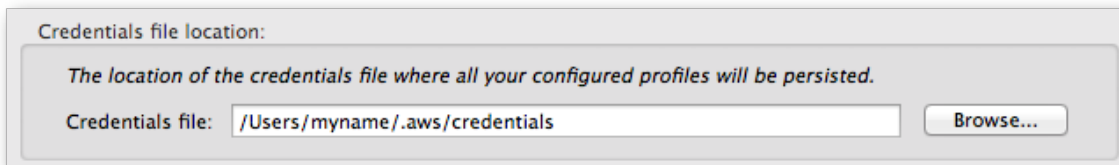
Important: It is *strongly recommended* that you don't store your AWS credential information within any network-shared directory or within any source-control-managed projects. Always retain strict control of your AWS access keys!

2.3 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.
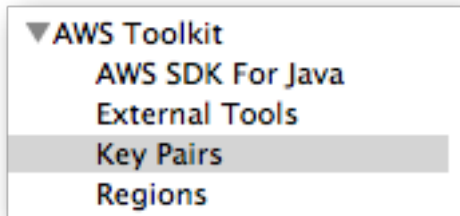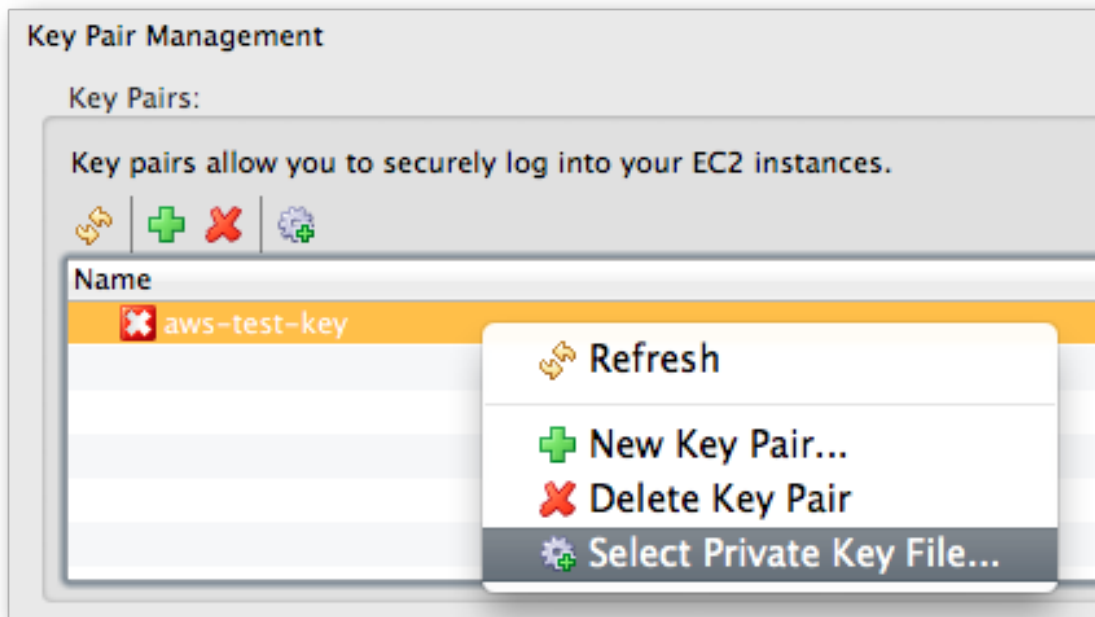

2. Select *Key Pairs*.

   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.
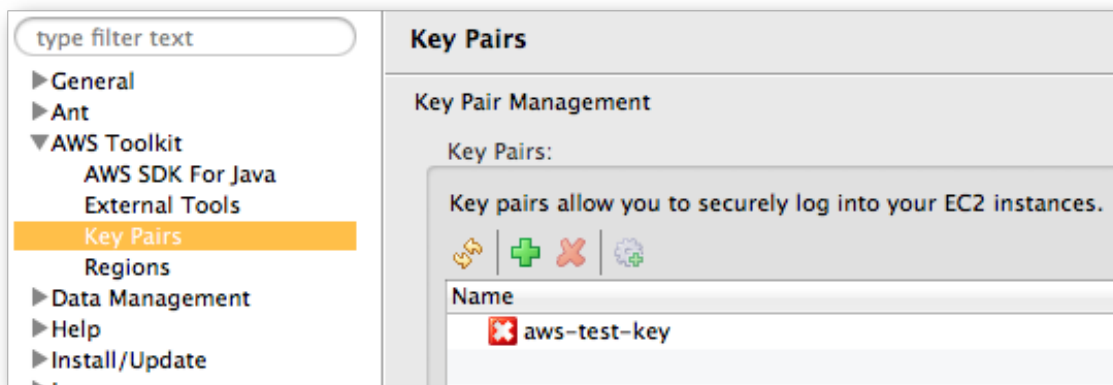
3. Right-click the key pair and, from the context menu, select *Select Private Key File...*

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.

3.1 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 Java Developer Guide.

3.1.1 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*.
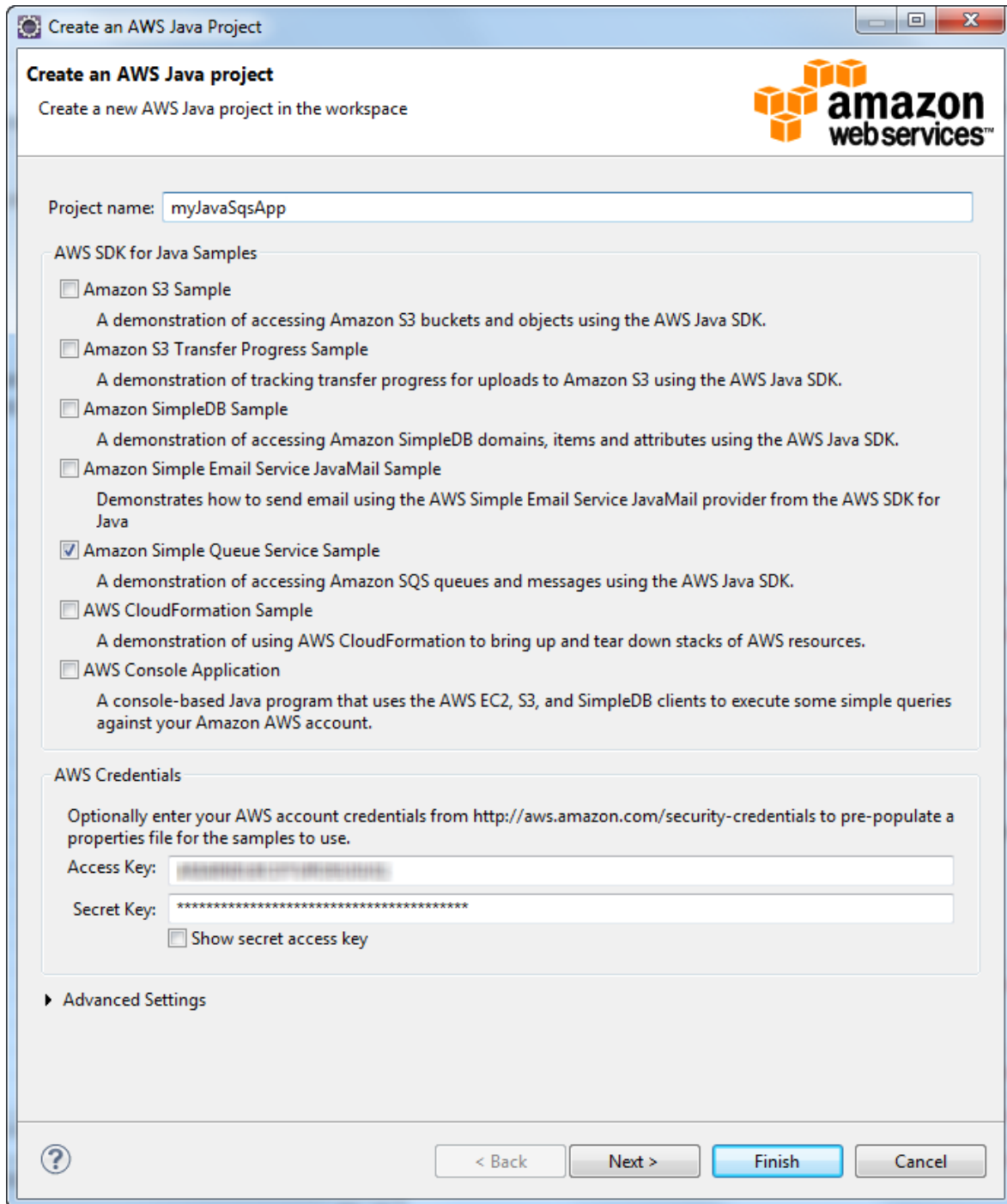
3. Click *Finish*.

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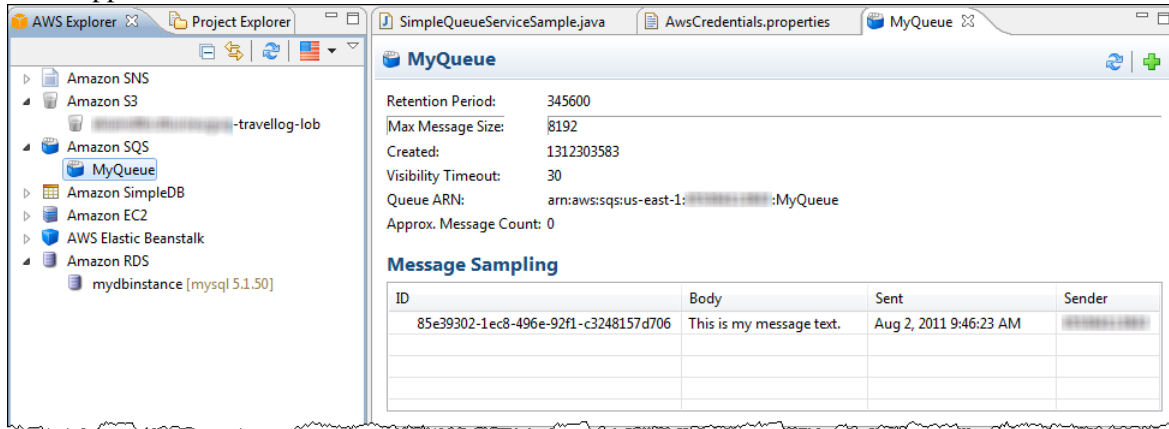
12. Press `F8`. The Java client application will continue running and terminate normally.

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.

3.2 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*, 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.
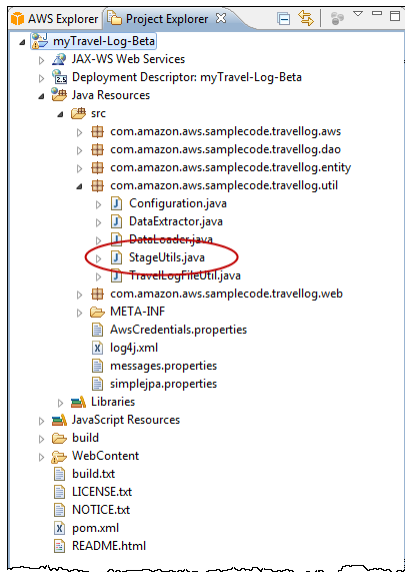
The `StageUtils` class exposes the following method.

```
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.

```
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.
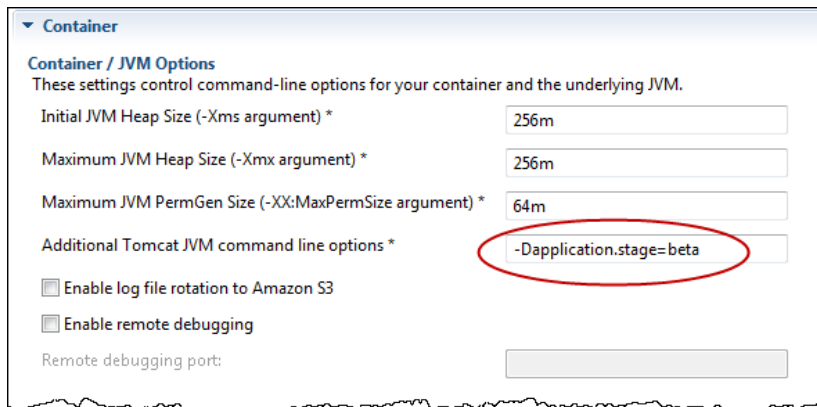
6. After you have added the system property value, click the *File* menu, and then click *Save*. Eclipse will save the new configuration. The application should restart automatically. You can check the *Events* tab—at the bottom of the Eclipse editor—for the event that indicates that the new configuration 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.
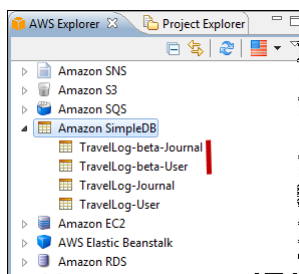
Note: For more information about configuring the container, see Creating and Deploying Java Applications on AWS Elastic Beanstalk in the Elastic Beanstalk Developer Guide.

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* the AWS Toolkit for Eclipse on your system.

4.1 How to Access AWS Explorer

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

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* to determine which resources to display.
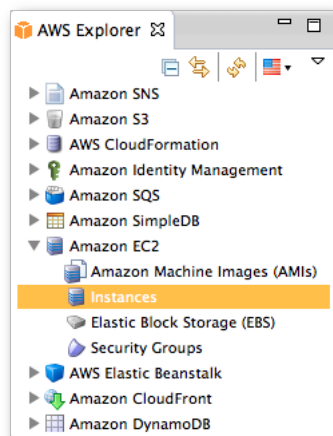
If you select any of the subnodes to Amazon EC2, Eclipse will open a view with detailed information about

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.

4.2 Using AWS 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, Amazon Kinesis, Amazon SNS, and Amazon Cognito. For more information about Lambda, see the 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.

4.2.1 Tutorial: How to Create, Upload and Invoke an AWS Lambda Function

This tutorial will guide you through the process of a typical AWS Lambda workflow and provide you with first-hand experience in 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* and that you understand the basic concepts and features of Lambda. If you are unfamiliar with Lambda, you can find out more at the AWS Lambda home page and in the Lambda Developer Guide.

Contents

- *Create a Lambda handler class*
- *Implement the handler method*
- *Allow Lambda to assume an IAM role*
- *Create an Amazon S3 bucket for your Lambda code*
- *Upload the code*
- *Invoke the Lambda function*
- *Where to go from here*

Create a Lambda handler class

First, you will 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.

To create an Lambda handler class

1. On the Eclipse toolbar, open the drop-down Amazon Web Services menu (identified by the AWS icon) and select *New AWS Lambda Java project...*

2. Add your Java *project name*, *package name*, and *class name* in the associated input boxes. You can choose any valid names that you want. This tutorial will use the following sample values:

- *Project name*: HelloLambda

- *Package name*: *example*

- *Class name*: *Hello*

While you type, the code in the *Source preview* will change to reflect the changes you make in the dialog.

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

4. The second *Input Type* field is name of the actual Java type that will be returned, which must be a Java class (not a primitive type such as `float`, `int` or `boolean`). It defaults to `Object`. Change this value to `String`.

5. Change the value of *Output Type* to `String`, as well.

6. Verify that your entries look like the following screenshot (modify them if they are not), and then click *Finish*.

7. Once you click *Finish*, your project's directory and source files will be generated in your eclipse workspace, and a new web browser window will appear, displaying `README.html` (which was created for you in your project's root directory). `README.html` provides instructions to help 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 next steps that will be covered here.

Next, we'll implement the function in the `HelloLambda` Java project that was just created for you in Eclipse.

Implement the handler method

The *Create New Project* dialog creates a skeleton project for you, but it's up to you to 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. Using Eclipse's *Project Explorer*, open `Hello.java` in the *HelloLambda* project. It will contain code similar to the following:

```java
package example;

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

public class Hello implements RequestHandler<String, String> {

    @Override
    public String handleRequest(String input, Context context) {
        context.getLogger().log("Input: " + input);

        // TODO: implement your handler
        return null;
    }

}
```

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

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

Your `Hello.java` file should now contain:

```java
package example;

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

public class Hello implements RequestHandler<String, String> {
    @Override
    public String handleRequest(String input, Context context) {
            String output = "Hello, " + input + "!";
            return output;
    }
}
```

Allow Lambda to assume an IAM role

In order for Lambda to access your Lambda function, you will need to create an IAM role that gives it access to your AWS resources. The easiest way to do this is with the AWS Management Console.

To create an IAM role for Lambda

1. Sign in to the AWS Management Console.

---

2. Open the IAM console.

3. Select *Roles* on the sidebar, then *Create New Role*.

4. Add a name for your role, such as `hello-lambda-role`, and click *Next Step*.

5. On the *Select Role Type* page, select *AWS Lambda* within the *AWS Service Roles* list.

6. For *Attach Policy*, check *AmazonS3FullAccess*, which allows Lambda to access your Amazon S3 resources, and then click *Next Step* to continue.

Note: Amazon S3 is required because Lambda will upload your code to an Amazon S3 bucket when you deploy and run your Lambda function. You can use a bucket that you create (this is covered in the next section) or use an existing bucket.

---

7. Review your role parameters, then click *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 use by Lambda (recommended).

To create an Amazon S3 bucket for use with Lambda

1. Log in to AWS and go to the S3 console.

2. Click *Create Bucket*.

3. Enter a bucket name and select an AWS 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 the Regions and Endpoints topic in the *AWS General Reference*

4. Click *Create* to finish creating your bucket.

Upload the code

Next, we'll 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 code window and select *AWS Lambda*, then *Upload function to AWS Lambda....*

2. In the *Select Target Lambda Function* dialog that appears, select the AWS region to use. This should be the same region that you chose for your *Amazon S3 bucket*.
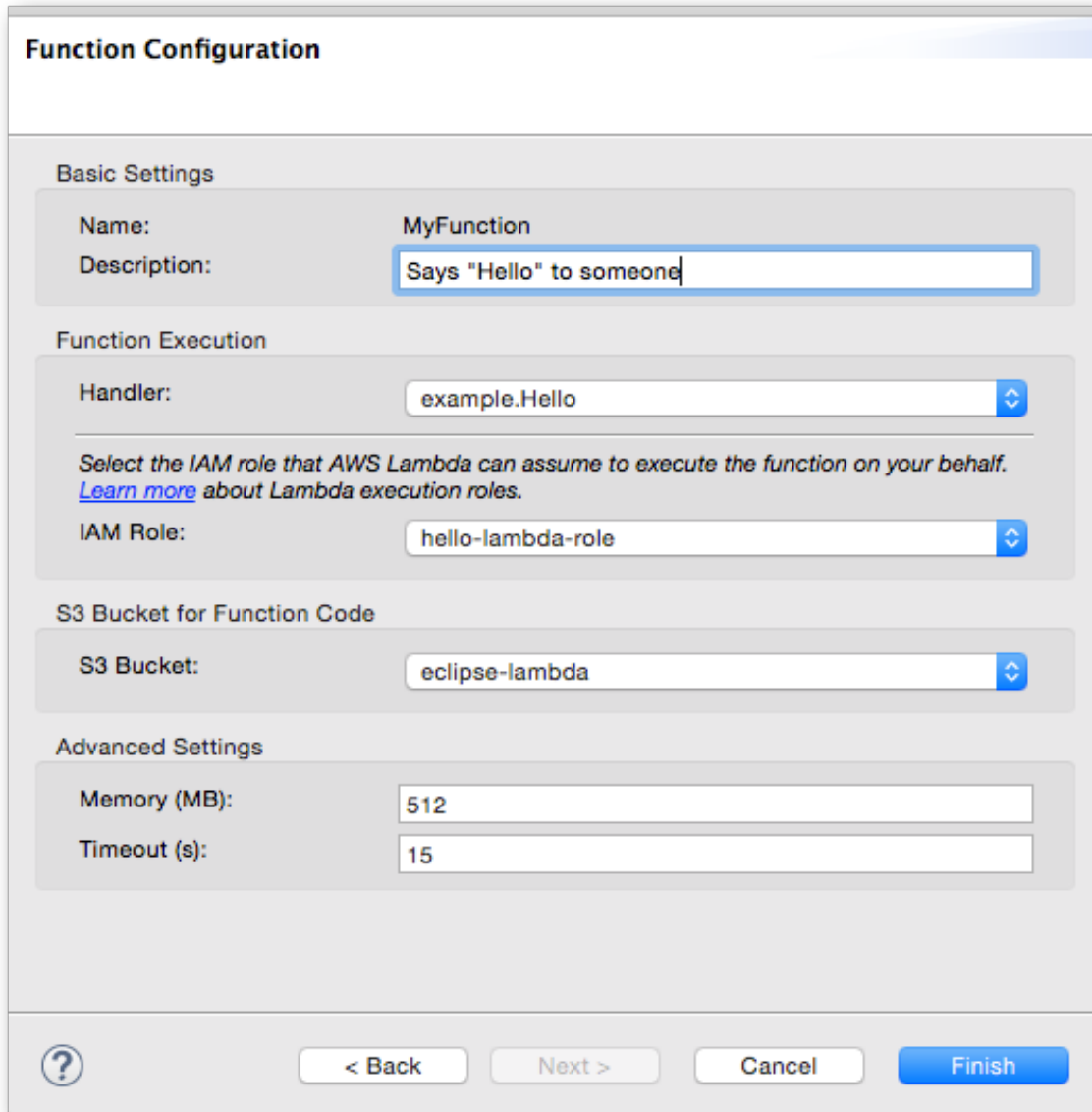
3. Select *Create a new Lambda function* and enter the name of your function (such as `HelloFunction`).

4. Click *Next* to proceed to *Function Configuration*.

5. Enter a description for your target Lambda function. You can leave the rest of the options as they are; the AWS Toolkit for Eclipse chooses default values for you.

For more information about the available options, see *Upload AWS Lambda Function Dialog*.

6. Click *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 Eclipse project name in the *Project Explorer* view:
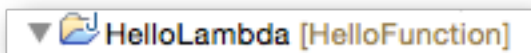
If you don't see this happen, you should open Eclipse's *Error Log* view. Lambda will write information about failures to upload or run your function to the error log for further debugging.

Invoke the Lambda function

You can now invoke the function on AWS Lambda.

To invoke your Lambda function

1. Right-click in your code window and select *AWS Lambda*, then *Run on AWS Lambda*.

2. In the input box, type a valid JSON string, such as "AWS Lambda".
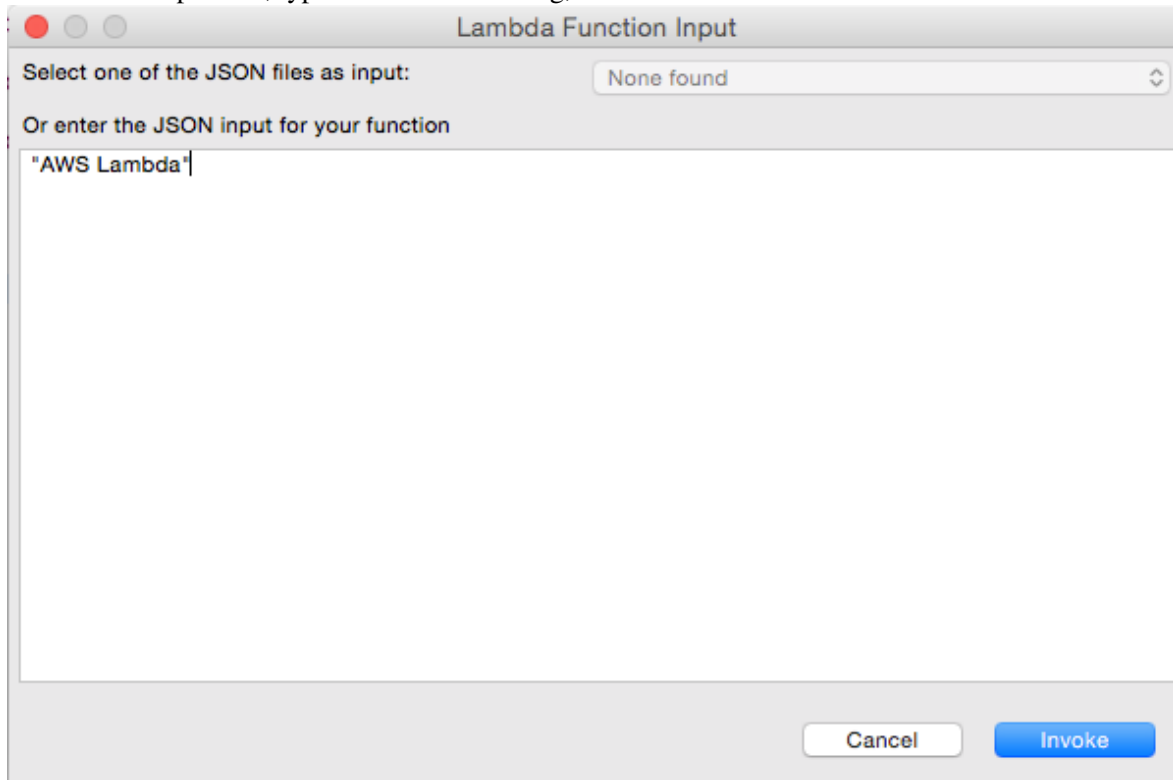
Tip: You can add new JSON input files in your project, and they will show up in this dialog as long as the file name ends with ".json". You can use this feature to provide standard input files for your Lambda functions.

---

3. Click *Invoke* and it will 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 will automatically appear if it isn't already shown).
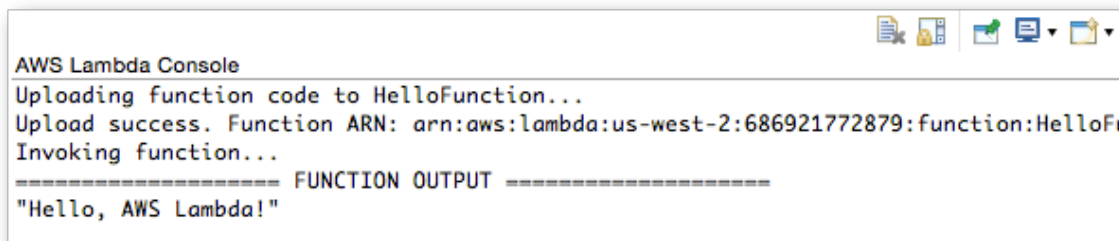
Congratulations, you've just run your first Lambda function directly from the Eclipse IDE!

Where to go from here

Now that you've uploaded and deployed your function, try changing the code and re-running the function. Lambda will automatically re-upload and invoke the function for you, and print output to the console.

For more information about each of the screens that were covered in this tutorial, as well as a full description of each option, see the *AWS Lambda Interface Reference*.

For more information about Lambda itself, and about writing Java code for Lambda, see Authoring Lambda Functions in Java in the *AWS Lambda Developer Guide*.

4.2.2 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.

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.

Contents

- *Launching the dialog*
- *Create Project Dialog user interface*

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 AWS Lambda Function Dialog

The *Upload Lambda Function* dialog will create a Lambda function and upload your code to be run when the Lambda function is invoked.

Contents

- *Launching the dialog*
- *Select Target Lambda Function user interface*
- *Configure Function user interface*

Launching the dialog

The *Upload Lambda Function* dialog can be launched in the following ways:
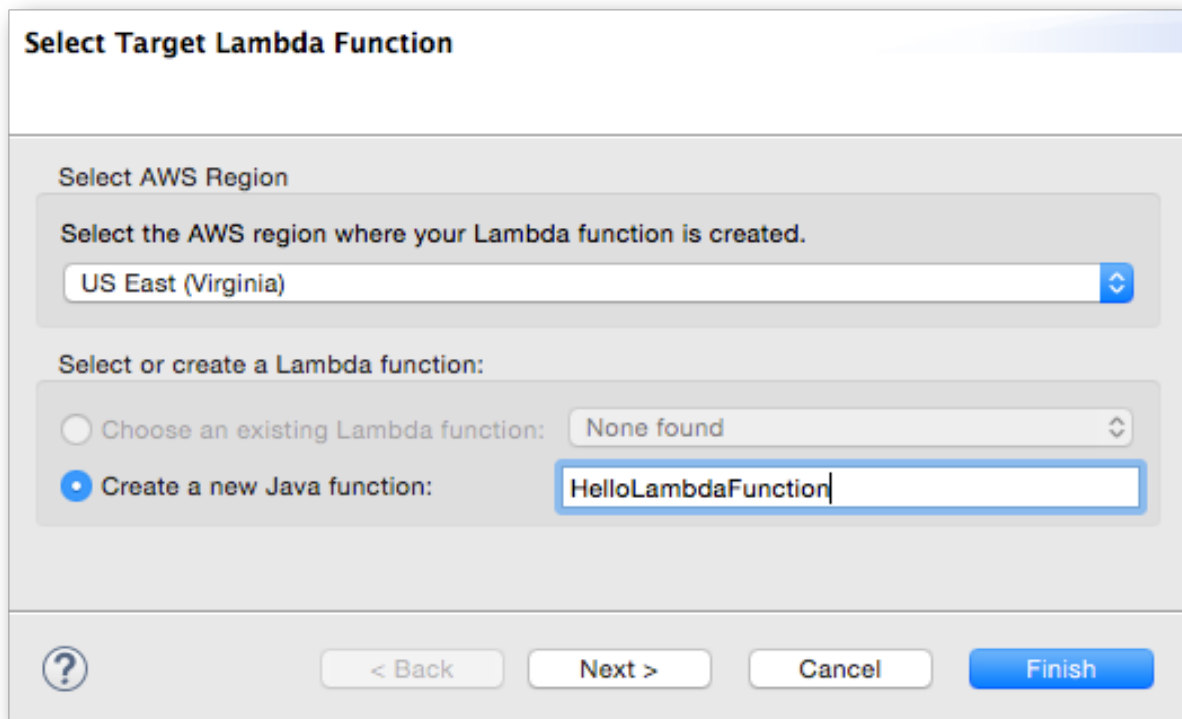
- by opening the context menu for your AWS Lambda Java Project in Eclipse's *Project Explorer* view, and selecting *Amazon Web Services > Upload function to AWS Lambda....*

- by opening the context menu in the code window for your Java class and selecting *AWS Lambda > Upload function to AWS Lambda....*

The Upload Function dialog contains two screens:

- *Select Target Lambda Function*

- *Function Configuration*

Select Target Lambda Function user interface

Select AWS Region *Required*. Select the region where your Lambda function will be created.
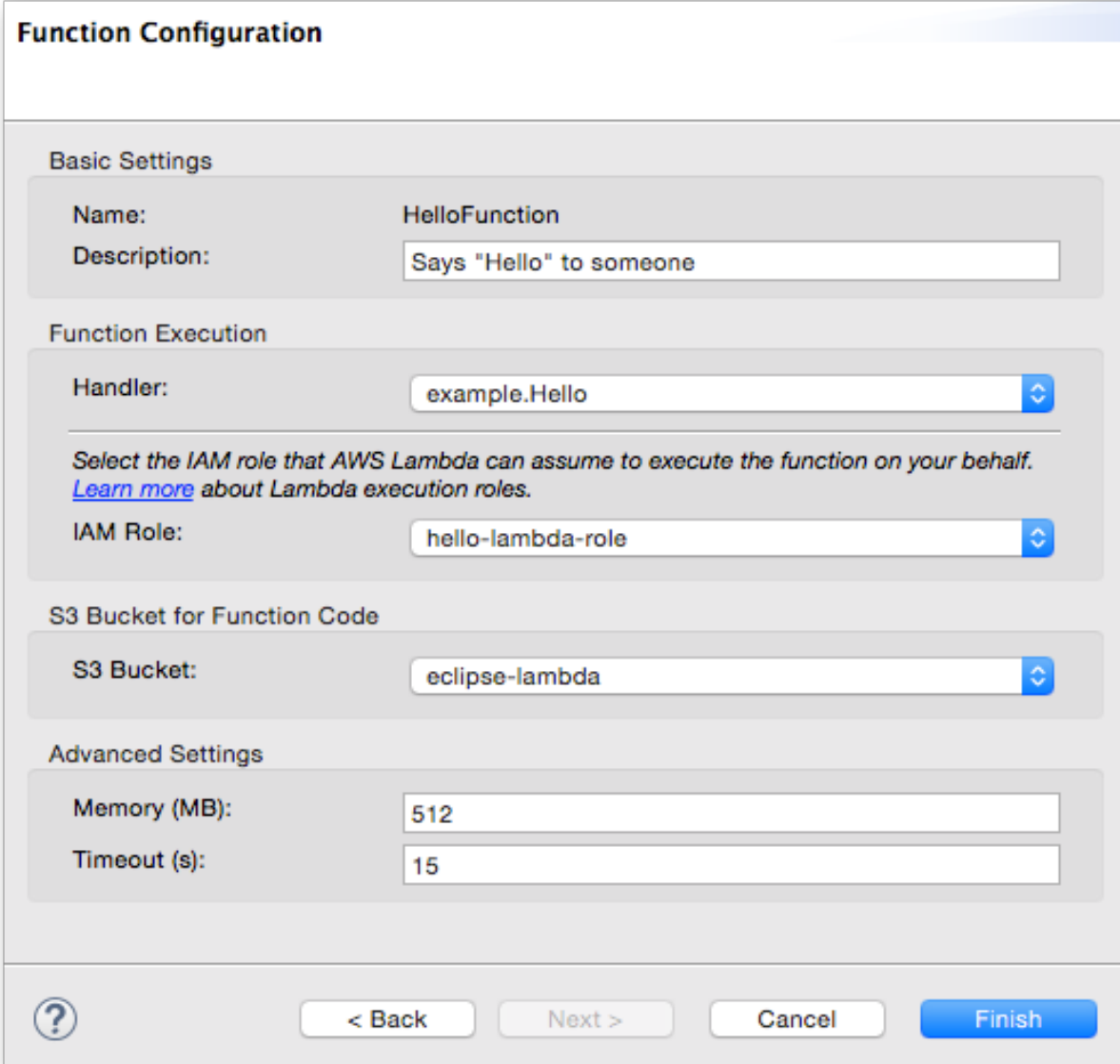
> *Default*: the default AWS Management Console region for your AWS account.

Select or create a Lambda function *Required*. You must choose 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 Java function*

When you click *Finish*, control proceeds to the *Configure Function* screen.

Configure Function user interface

The screen is divided into four sections, each with its own settings.

Basic Settings   This section shows the function name and allows you to add a text description.

Name *Immutable*. The name is determined by the name chosen during the *Select Target Lambda Function* screen. It can not be modified here, though you can click *Back* to re-enter it on the previous screen.

Description *Optional*. A text description of the function.

> *Default*: the description is empty.

Function Execution   This section allows you to modify the execution environment for the function.

Handler *Required.* The Java class that contains your Lambda function code.

> *Default*: the package and class name from your project are automatically selected for you.

IAM Role *Required.* The role that Lambda will use to access your AWS resources during the execution of your function. You must make sure that this role provides access for AWS Lambda to at least Amazon S3, but if your function uses any other AWS resources, you must also provide access to each of the services used.

> *Default*: the first IAM role from your AWS account.

S3 Bucket for Function Code This section allows you to set an S3 bucket used by your Lambda function.

S3 Bucket *Required.* An Amazon S3 bucket that can be used by your function's code. Only buckets that are in the same region that you will run the function in will be displayed here.

Advanced Settings This section contains settings that are less often used, they provide further 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 will be 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.

> **Contents**
> - *Launching the dialog*
> - *Options*

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:
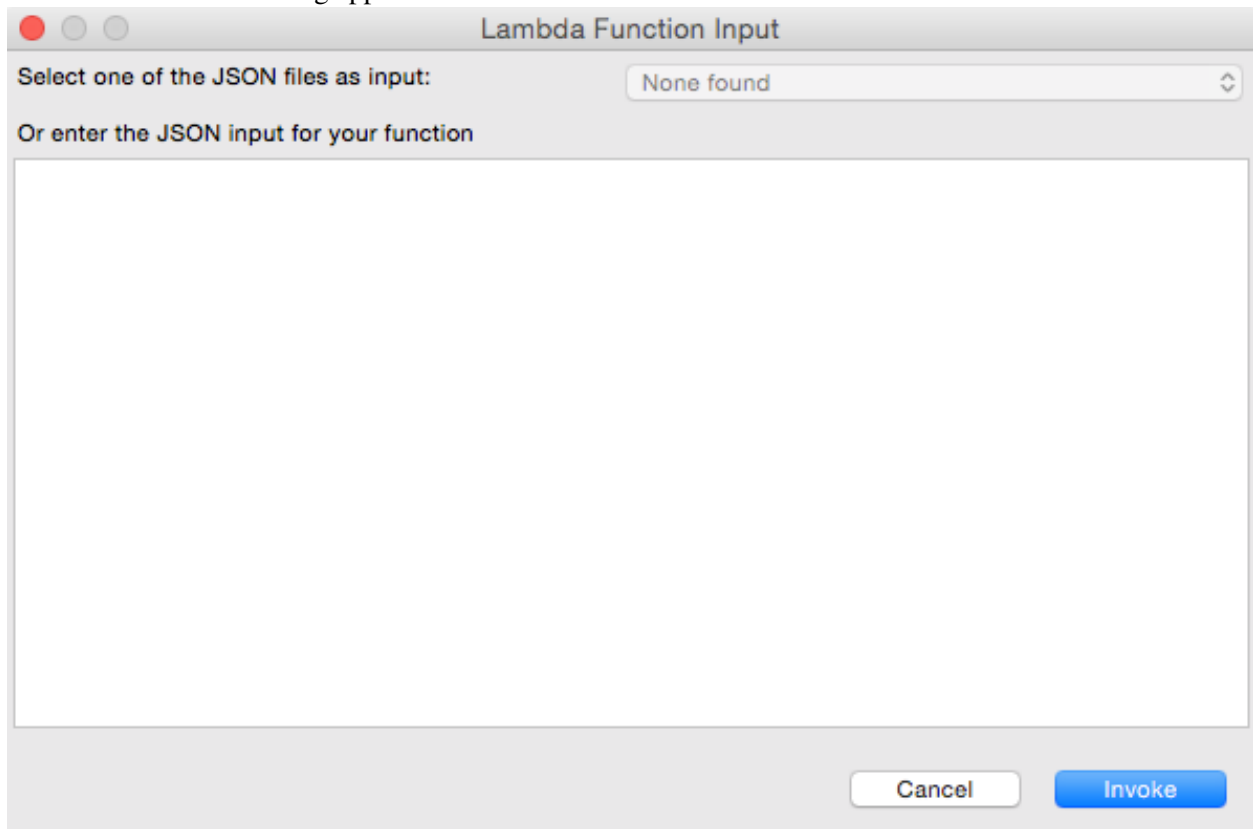
Options

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.

4.3 Using the CloudFormation Template Editor for Eclipse

The AWS Toolkit for Eclipse now includes a built-in 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.

4.3.1 Adding and Accessing CloudFormation Templates in Your Eclipse Project

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.
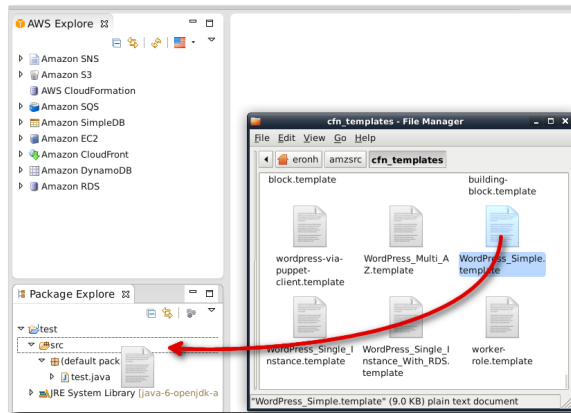
2. Choose how you would like to add the file to your project, and click *OK*.
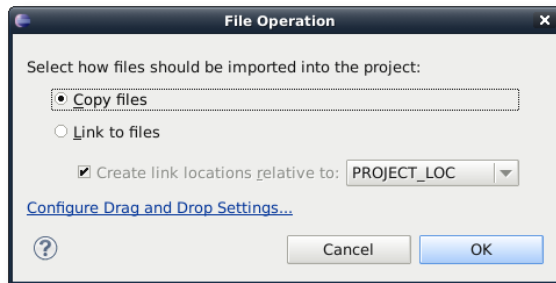
To access a CloudFormation template in your Eclipse project

- Double-click the template name in *Package Explorer* to begin editing the file.
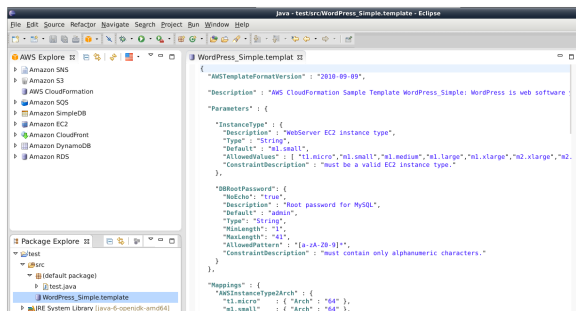
Note: Files that end with .template or .json will automatically use the CloudFormation template

---

editor. If your file is not automatically recognized as an 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*
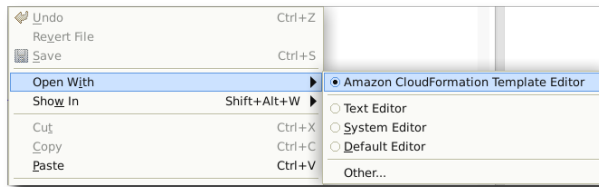
4.3.2 Deploying a CloudFormation Template in Eclipse

Note: At this time, 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 CloudFormation `.template` file open in the CloudFormation template editor (see *Adding and Accessing CloudFormation Templates in Your Eclipse Project* for more information), right-click on the open template and select *Run on AWS*, then *Create Stack* on the context menu.
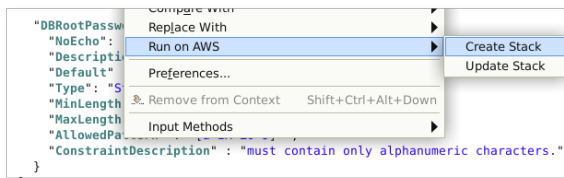
2. In the *Create New CloudFormation Stack* dialog, enter your stack name in the *Stack Name* field. Your template file should be automatically chosen in the *Template File* field.
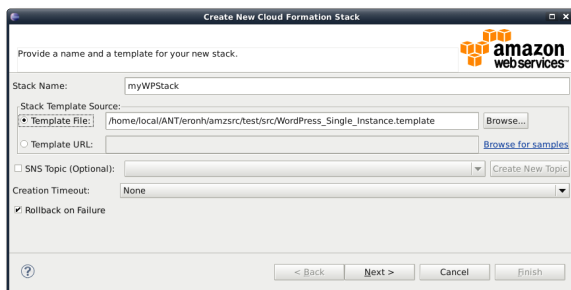
3. Choose any (or none) of the following options:

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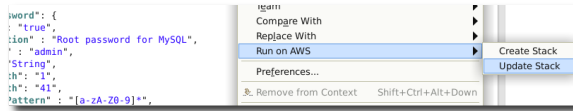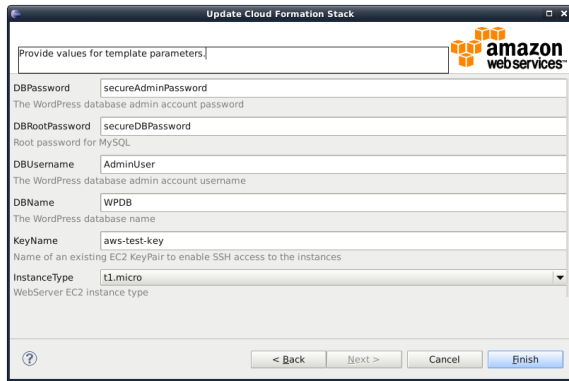
Creation Timeout – choose how long CloudFormation should allow for the stack to be created before it is declared failed (and rolled back, unless the *Rollback on failure* option is unchecked.

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.

4.3.3 Updating an CloudFormation Template in Eclipse

To update an CloudFormation template from Eclipse

1. With your CloudFormation `.template` file open in the CloudFormation template editor (see
   Adding and Accessing CloudFormation Templates in Your Eclipse Project 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 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.
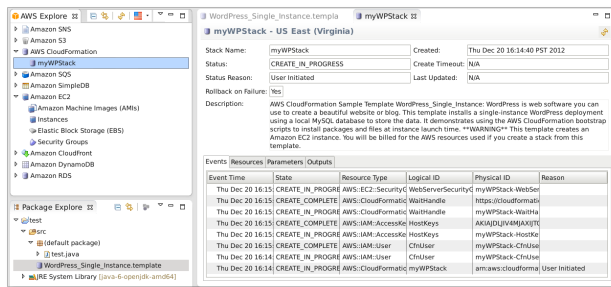
4.3.4 Validating an CloudFormation Template in Eclipse

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.
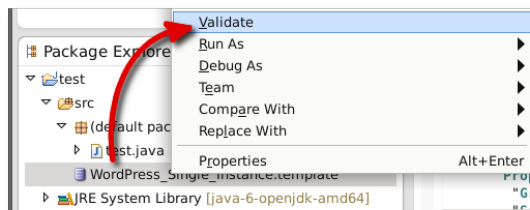
    – Right-click the template that you are editing in the editor pane and click *Validate* on the context menu.
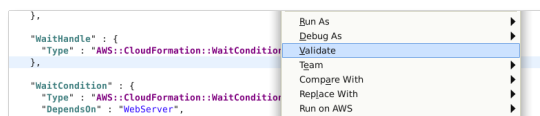
Important: Your template will be validated for *JSON correctness* only; it will not be validated for *CloudFormation correctness*. A stack template validated in this way can still fail to launch or update.

4.4 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.

4.4.1 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 Primary Key in the 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 Amazon DynamoDB in the 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.

4.4.2 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.
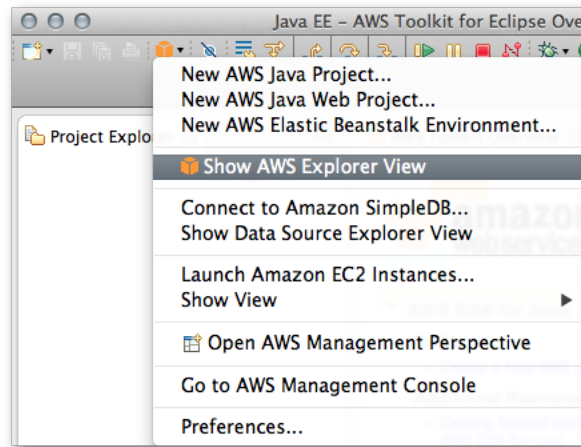
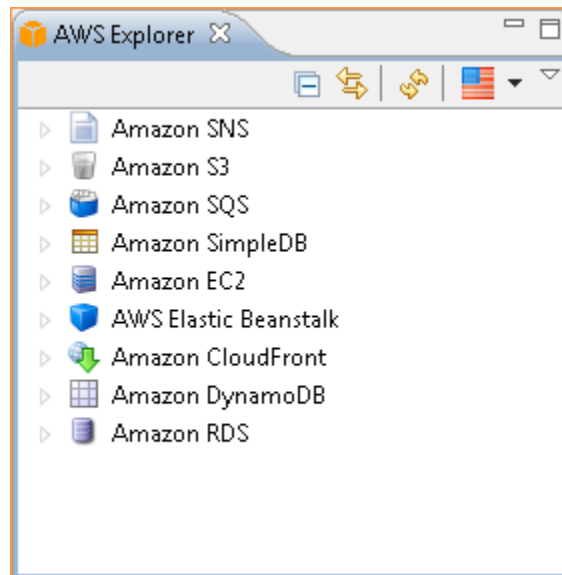---

Fig. 4.1: AWS Icon Menu

Fig. 4.2: Display Amazon DynamoDB tables in AWS Explorer.
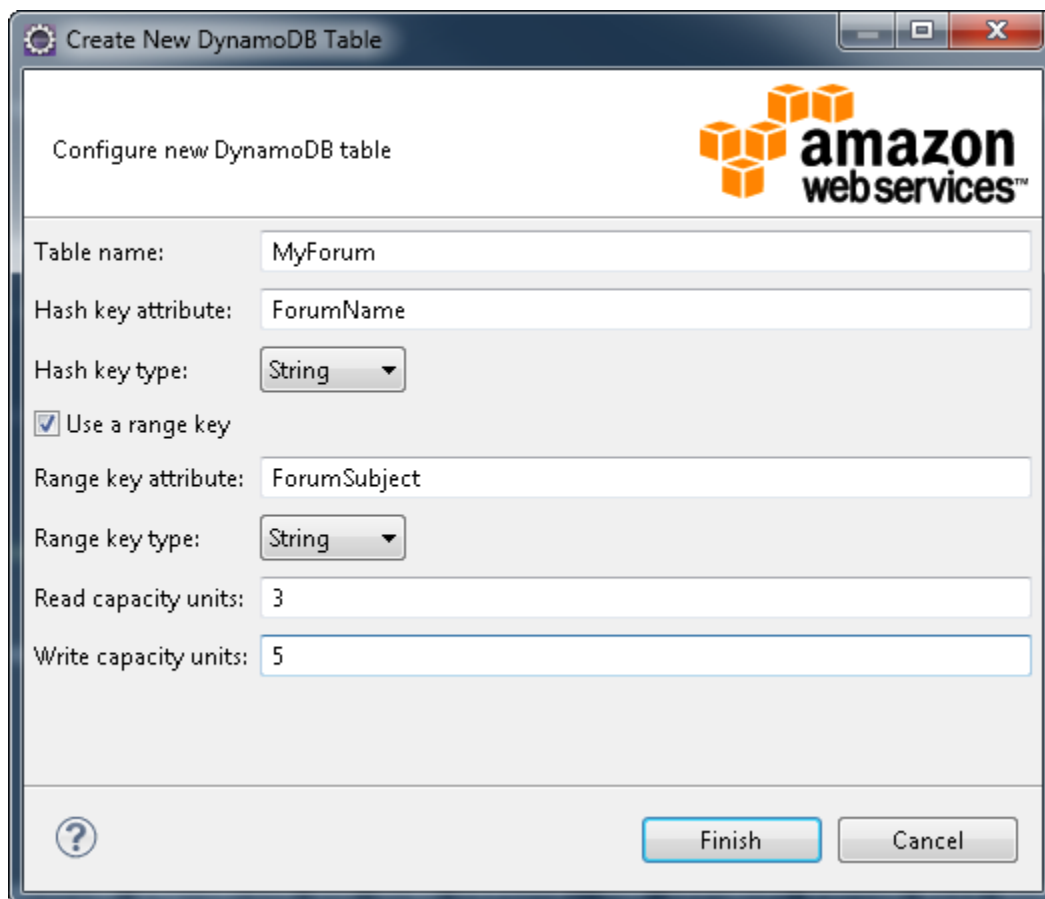
Fig. 4.3: Creating a table

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.

4.4.3 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.

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.

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*.

4.4.4 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*.

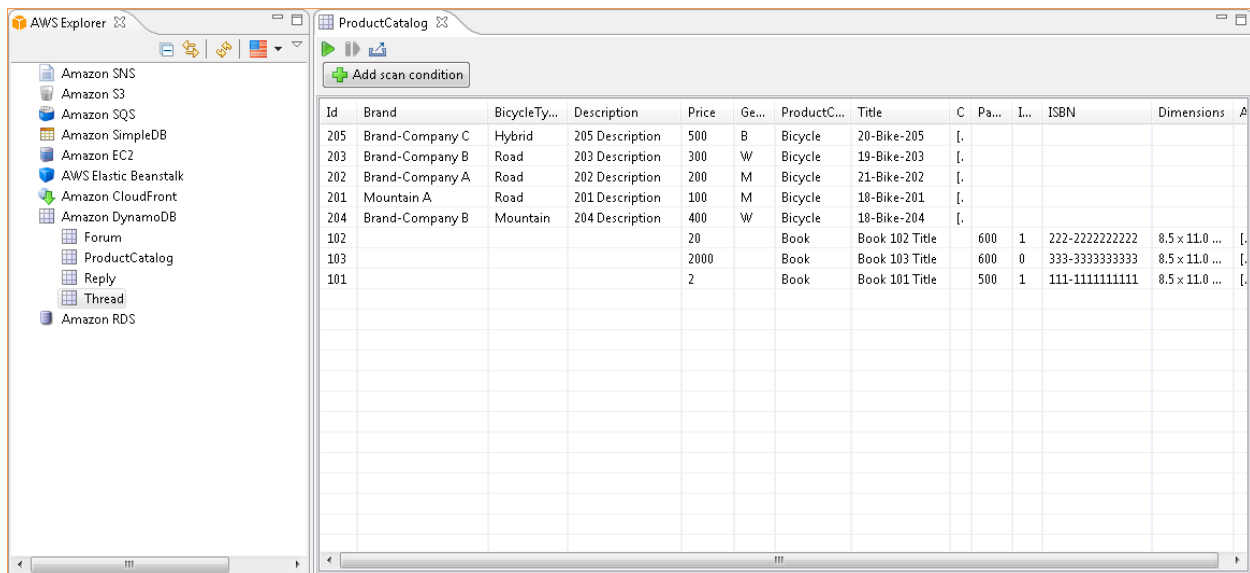**To perform a Scan on an Amazon DynamoDB table from AWS Explorer**

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.

To return to the view of the table that includes all items, double-click *Amazon DynamoDB* in *AWS Explorer*.

Paginating Scan Results

At the top of the view are three buttons.

Fig. 4.4: Amazon DynamoDB Grid View

Fig. 4.5: Cell editing in Amazon DynamoDB Grid View
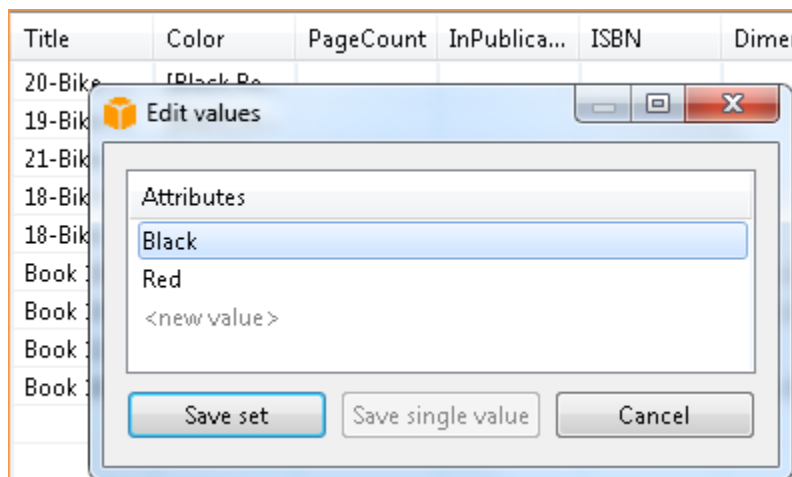

Fig. 4.6: Editing set values in Amazon DynamoDB Grid View
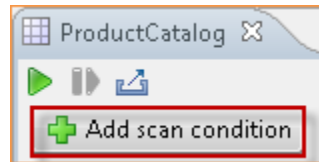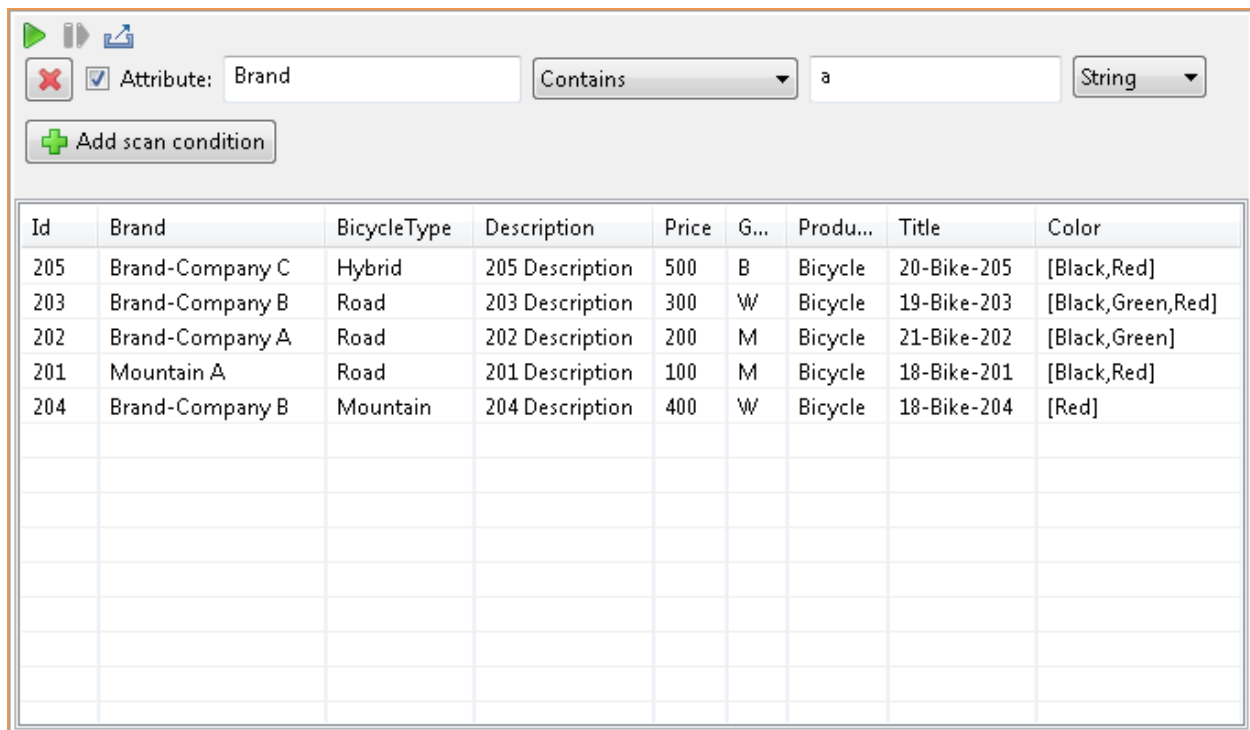
Fig. 4.7: Scan button

Fig. 4.8: Scan button

Fig. 4.9: Paginate and export buttons

The *second* button provides pagination for Scan results. Clicking the *rightmost* button exports the results from the current scan into a CSV file.

4.5 Launch an Amazon EC2 Instance from an Amazon Machine Image (AMI)

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*

   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).

4. Click *Finish*.

5. In AWS Explorer, under the *Amazon EC2* node, right-click the *Instances* subnode and select *Open EC2 Instances View*.

   Your EC2 instance should appear in the *EC2 Instances* view. It may take a few minutes for the instance to transition into the *running* state. Once the instance is running, you can right-click the instance to bring up a context menu of operations that you can perform on the instance. For example, you can terminate the instance from this menu. You can also copy the instance's public DNS address. You would use this address to connect to the instance using SSH.

4.6 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.

4.6.1 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*.

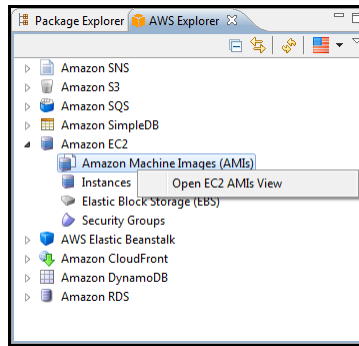2. Right-click in the left pane of the *EC2 Security Groups* tab, and then click *New Group*.

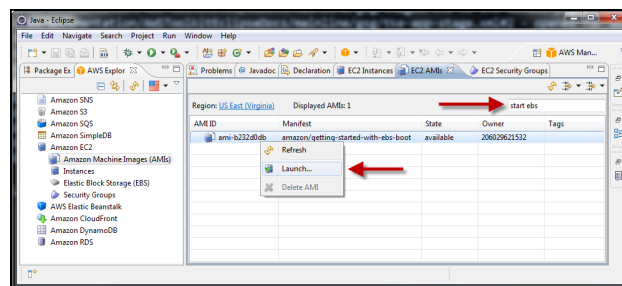Fig. 4.10: AMI configuration dialog box

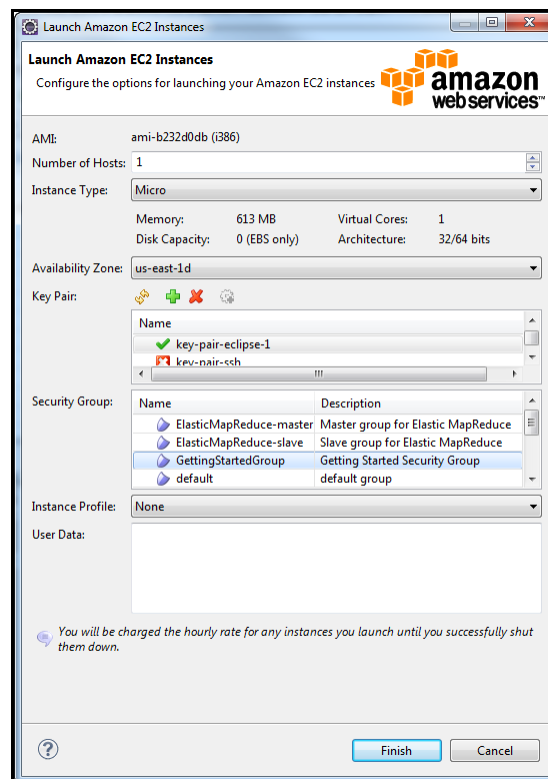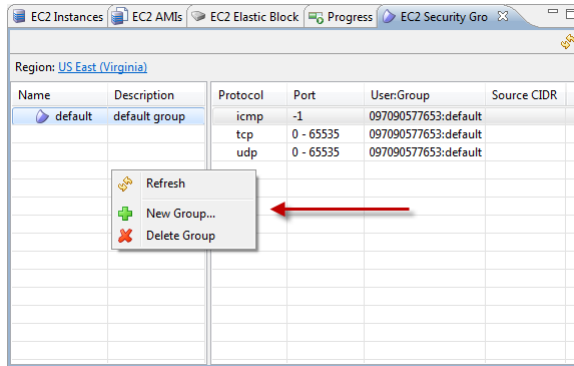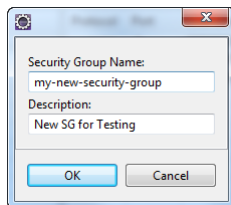Fig. 4.11: Select the Getting Started with EBS AMI

Fig. 4.12: Launching an AMI from AWS Explorer

3. In the dialog box, enter a name and description for the new security group. Click *OK*.

4.6.2 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*.
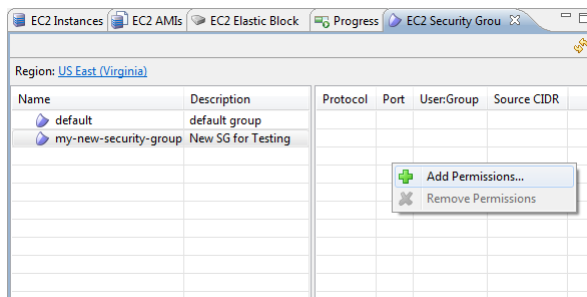
   Invoke Add Permissions UI

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*.
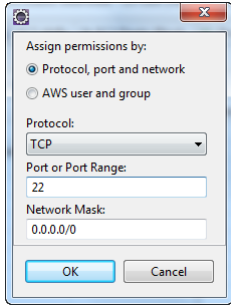
You could also set permissions to the security group by specifying a UserID and security group name. In this case, Amazon EC2 instances in this security group would accept all incoming network traffic from Amazon EC2 instances in the specified security group. It is necessary to also specify the UserID 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 Amazon EC2 Security Groups.

4.7 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.

4.7.1 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.
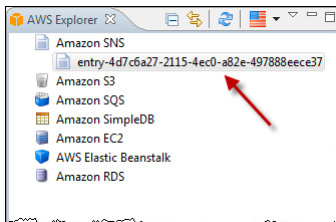
2. Double-click this SNS topic to open a detail view in the Eclipse editor pane. In this example, the *Subscription ARN* column says that the topic is pending confirmation. Amazon SNS requires a confirmation from the individual specified by the email address before SNS will send email notifications to that individual.
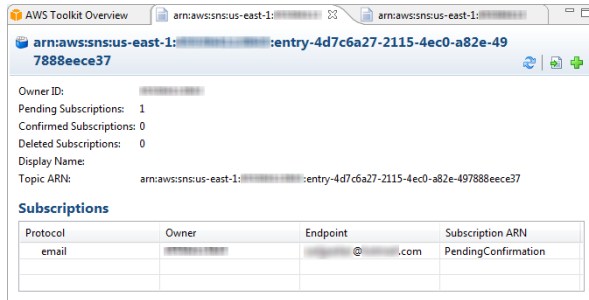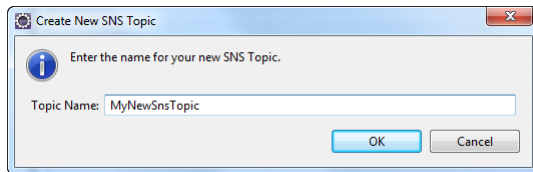
4.7.2 Add an Amazon SNS Notification

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.

3. To delete the subscription, right-click the entry in the *Protocol* column for the subscription and click *Delete Subscription*.

Note: The creation of the subscription will cause a verification email to be sent to the individual specified by the subscription "endpoint" email address. This email address will be used by AWS only to send notifications. It will not be used for any other purpose by AWS or Amazon.com.

4.8 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.

4.9 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.

Contents

- *About AWS Identity and Access Management*
- *Create an IAM User*
- *Create an IAM Group*
- *Add an IAM User to an IAM Group*
- *Manage Credentials for an IAM User*
- *Create an IAM Role*
- *Attach an IAM Policy to a User, Group, or Role*
- *Set Password Policy*

4.9.1 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.
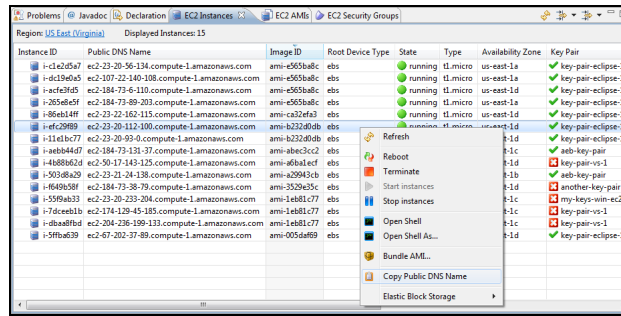
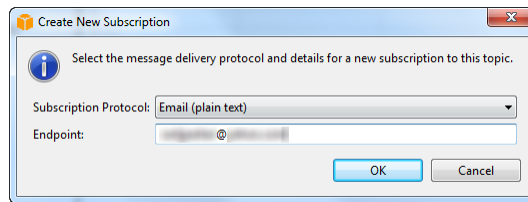Fig. 4.13: List of Amazon EC2 instances

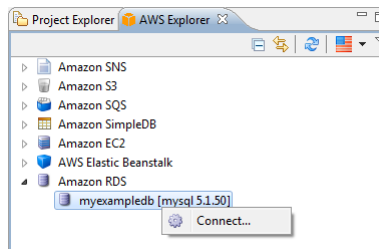

Fig. 4.14: Select the notification protocol and endpoint.
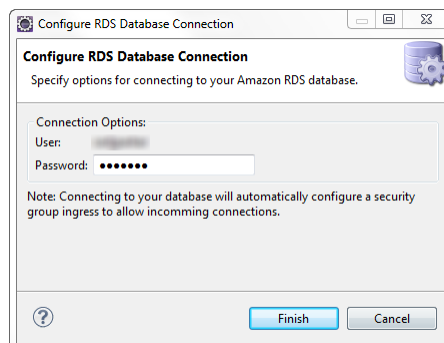

Fig. 4.15: Connect in context menu in AWS Explorer

Fig. 4.16: Authenticate against the database instance

4.9.2 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* and *Add an IAM User to an IAM Group*.

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*.

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*.
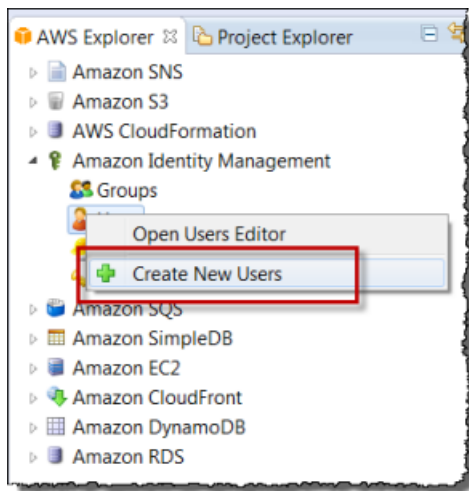
2. In the *Create New Users* dialog box, enter up to five names for new IAM users, and then click *Finish*. For information about constraints on names for IAM users, see Limitations on IAM Entities in the IAM User Guide.
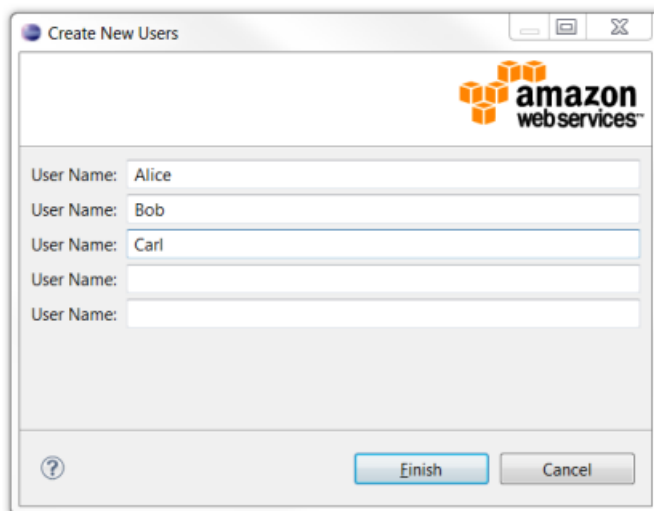
For information about adding a user to a group, see *Add an IAM User to an IAM Group*. For information about how to create a policy and attach it to the user, see *Attach an IAM Policy to a User, Group, or Role*.

4.9.3 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.
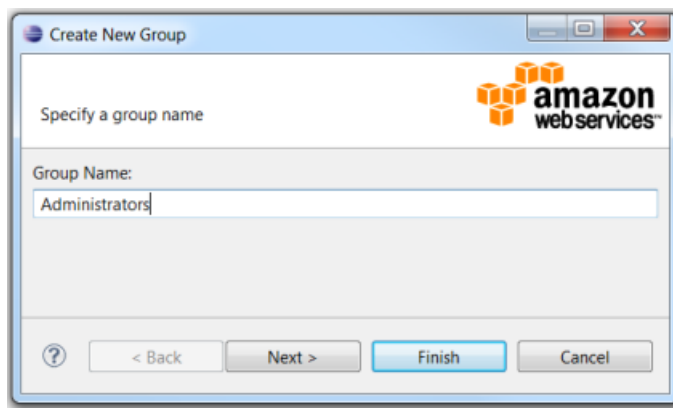
When you create a group, you can create a policy that includes the permissions that members of the group will have.
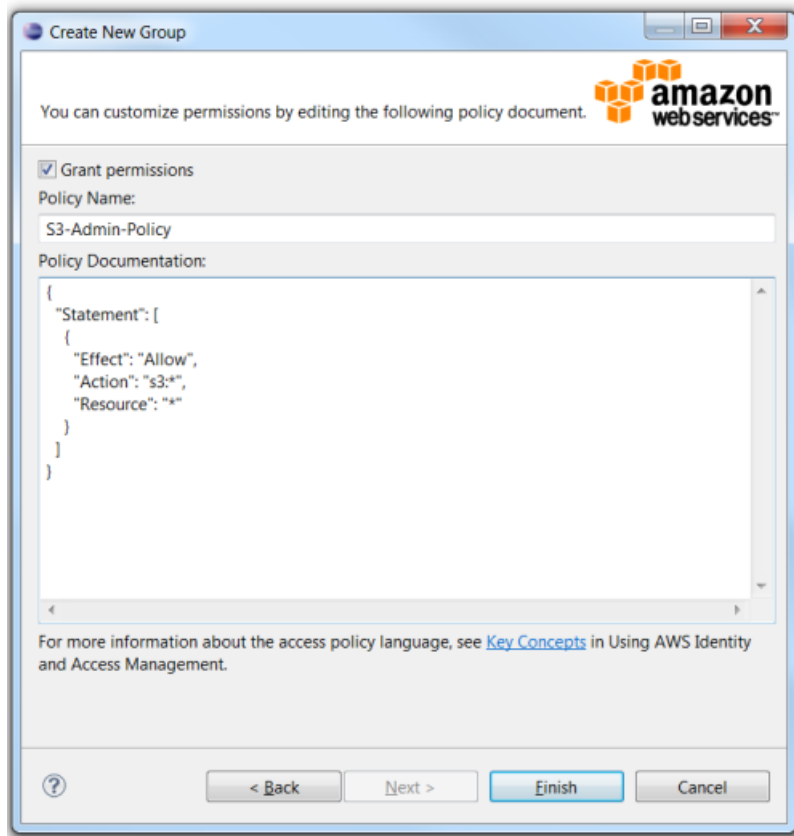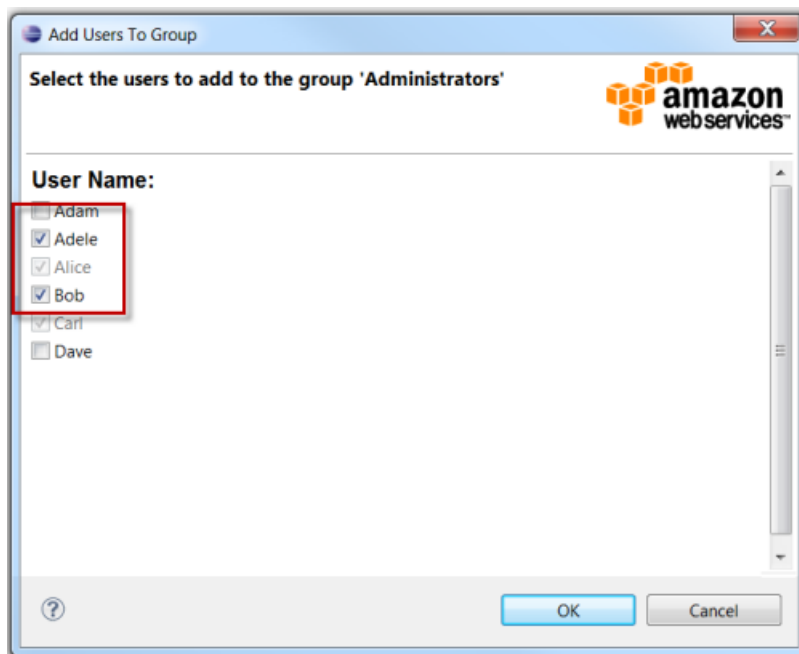
To create an IAM group

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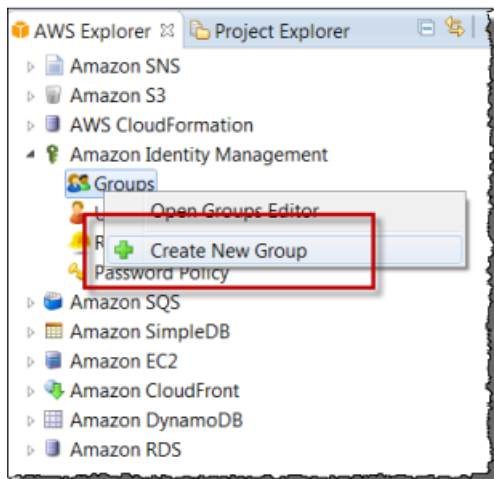
2. Enter a name for the new IAM group and then click *Next*.

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*.

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 IAM User Guide.

4. Click *Finish*.

For information about attaching additional policies to the IAM group, see *Attach an IAM Policy to a User, Group, or Role*.

4.9.4 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.
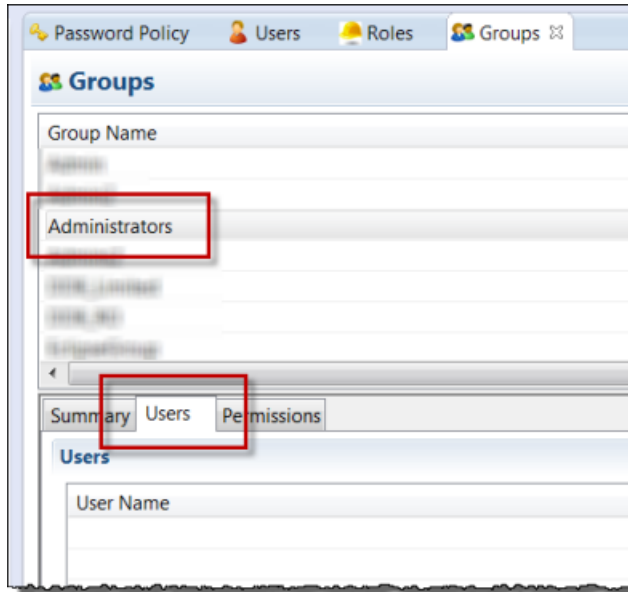
3. On the right-hand side of the bottom pane, click the *Add Users* button.

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

### 4.9.5 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.
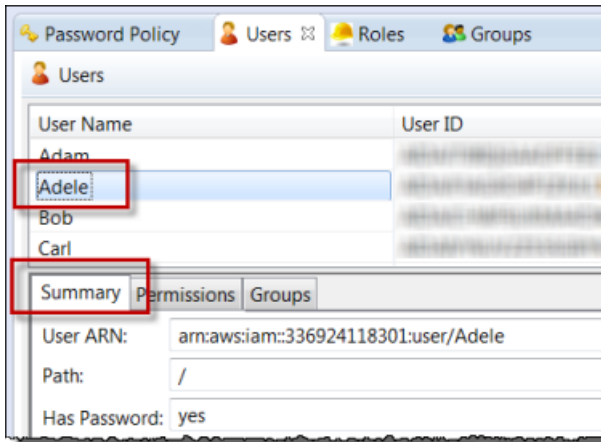
3. On the right-hand side of the bottom pane, click the *Update Password* button.
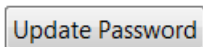
4. In the *Update User Password* dialog box, enter a password and then click *OK*.

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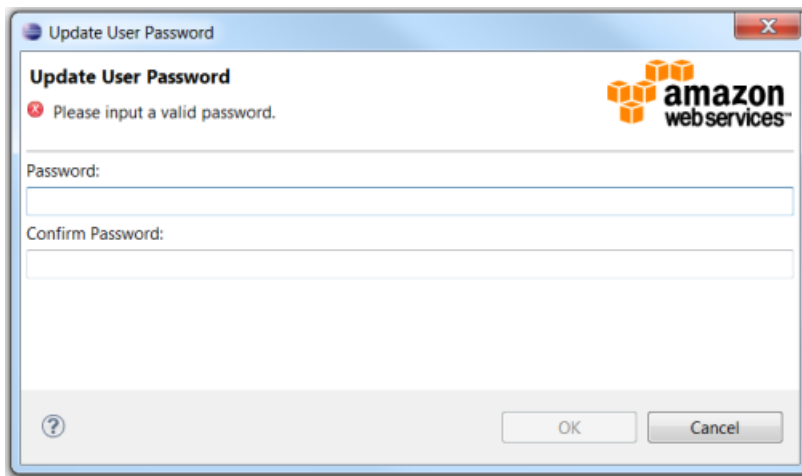
For each user you can also generate a set of access keys (an access key ID and a secret access key). These keys can be used to represent the user for programmatic access to AWS—for example, to use the AWS command-line interface (CLI), to sign programmatic requests using the SDK, or to access AWS services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

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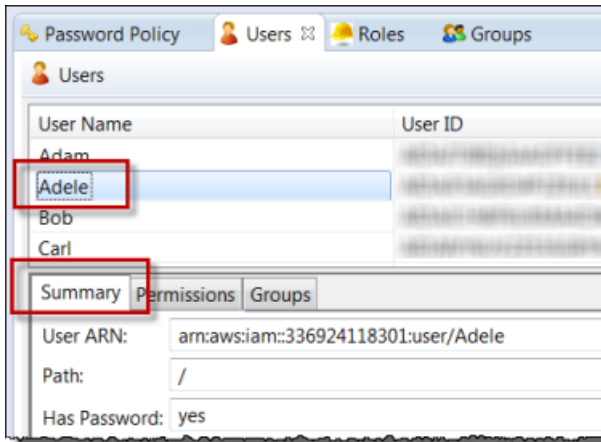
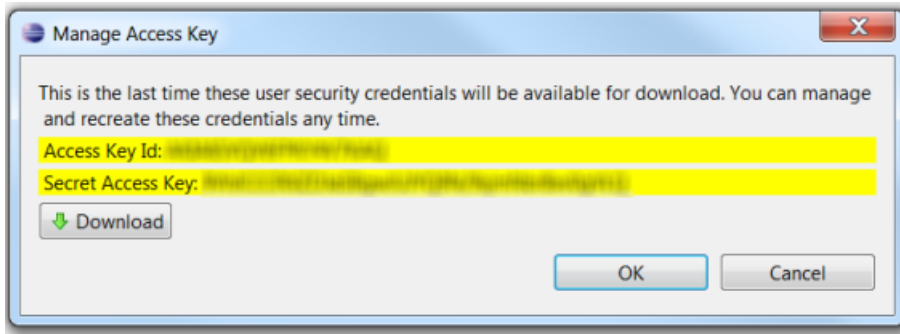3. Click the *Manage Access Keys* button.

A window is displayed where you can manage access keys for the user.
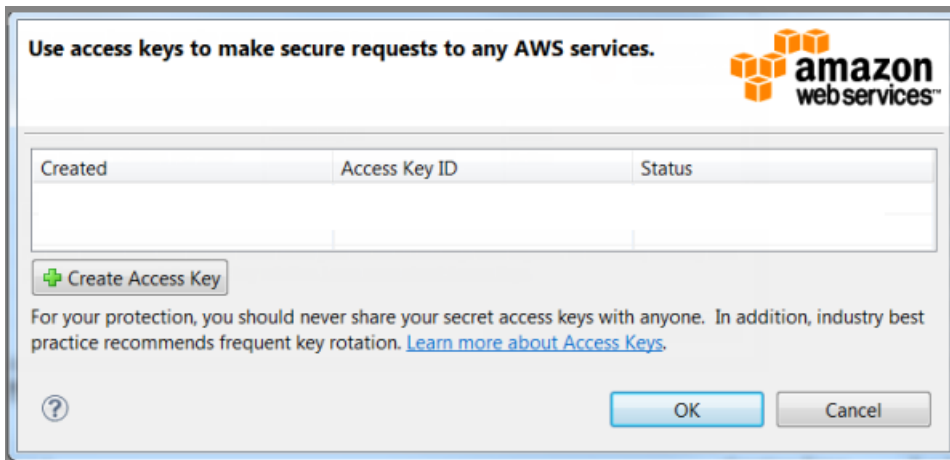
4. Click the *Create Access Key* button.

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*
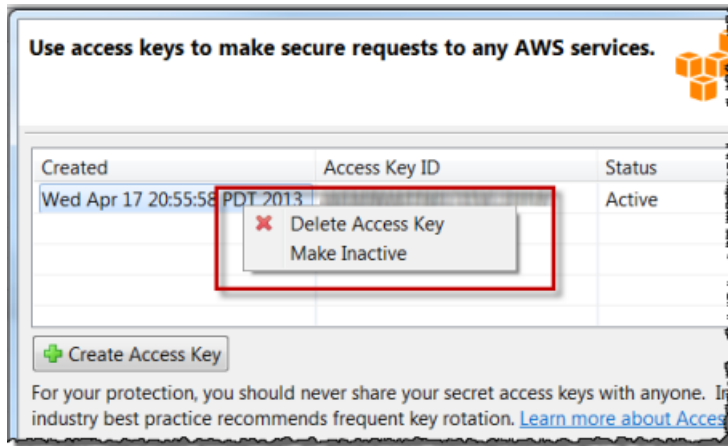
4.9.6 Create an IAM Role

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.
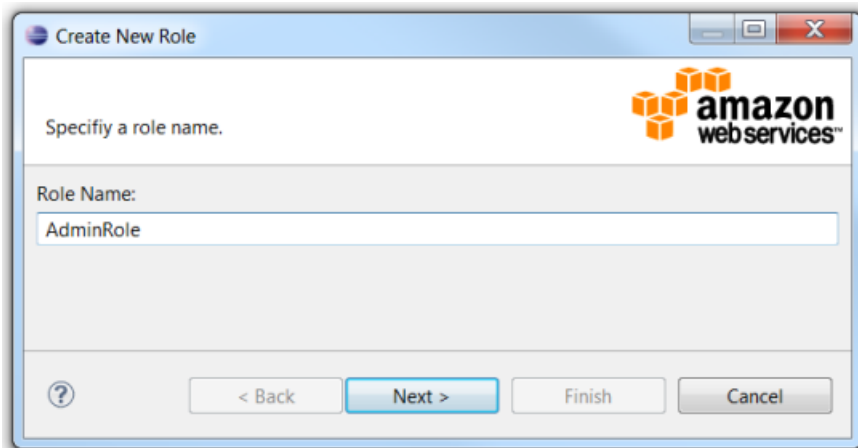
For more information about IAM roles, see IAM Roles in the IAM User Guide.
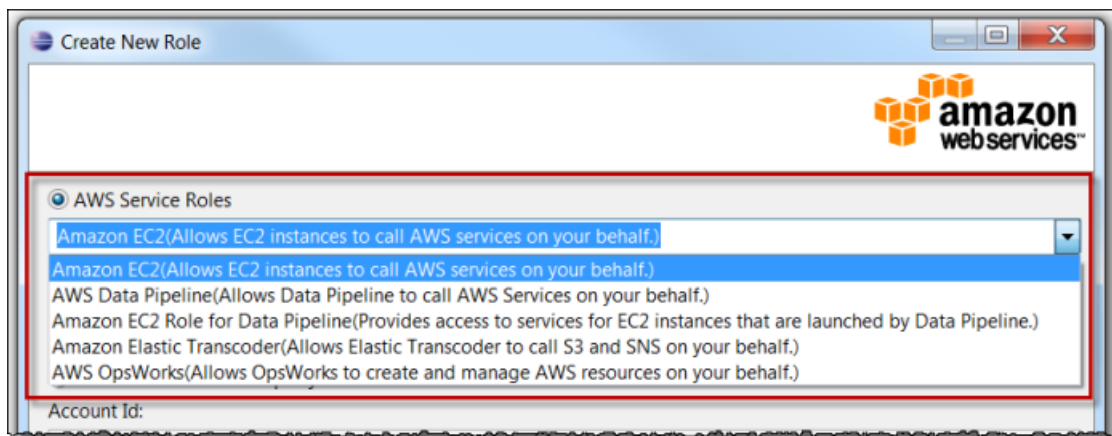
To create an IAM role

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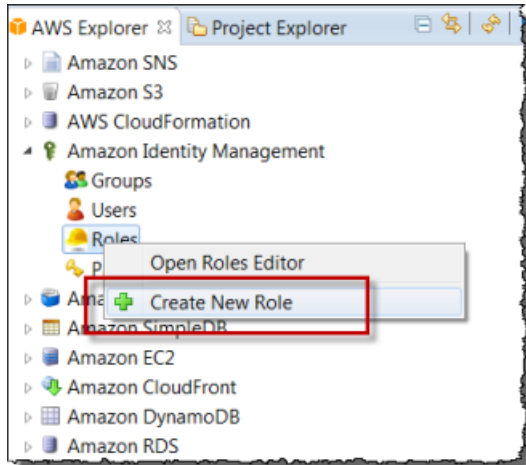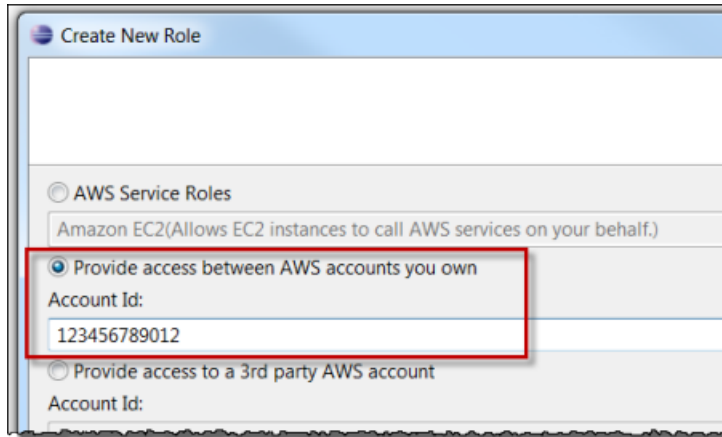
2. Enter a name for the IAM role and then click *Next*.

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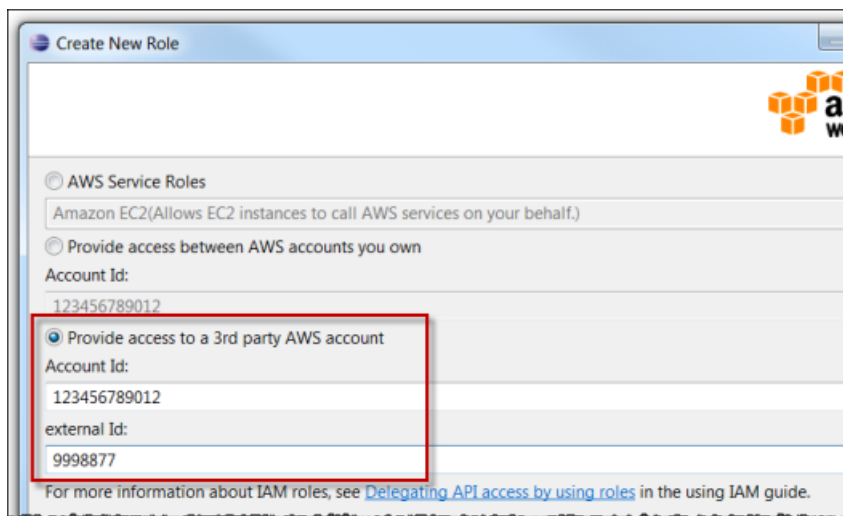
To provide access for a third-party account, select *Account ID* and enter the third party's AWS account number. If the third party has provided you with an external ID, enter that as well.
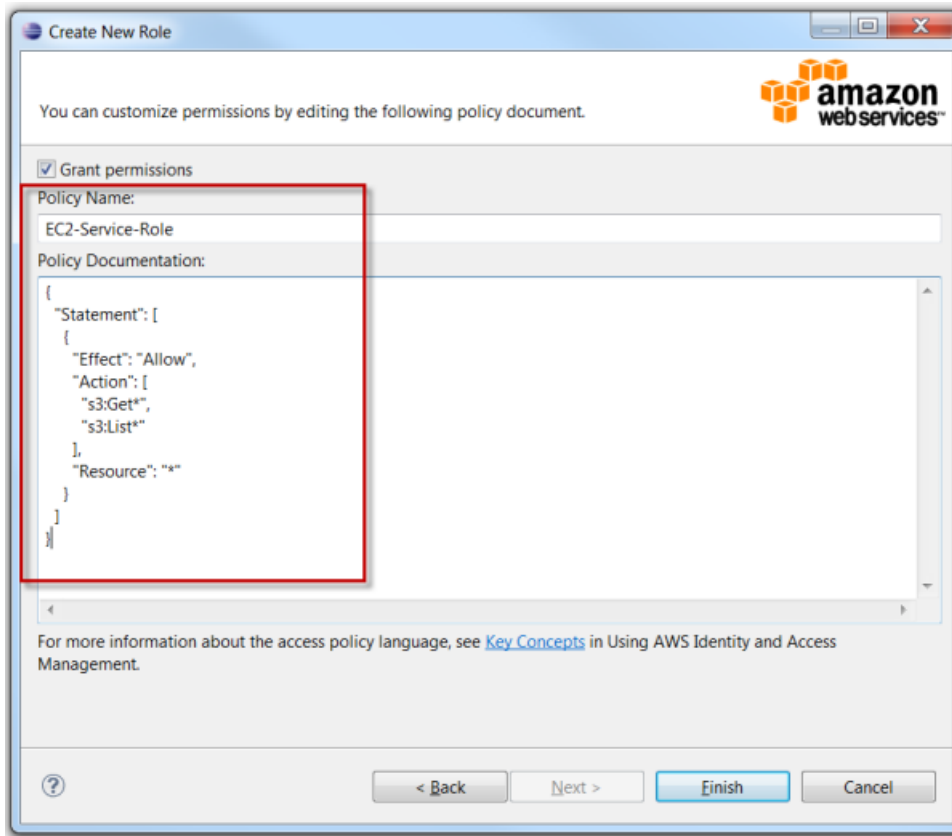

4. Click *Next*.

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*.

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 Java Developer Guide.

4.9.7 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.)

Note: If you attached a policy to a user, group, or role previously, you can use this procedure to attach an

additional policy. To edit an existing policy on a user, group, or role, use the IAM console, command-line tools, or API calls.

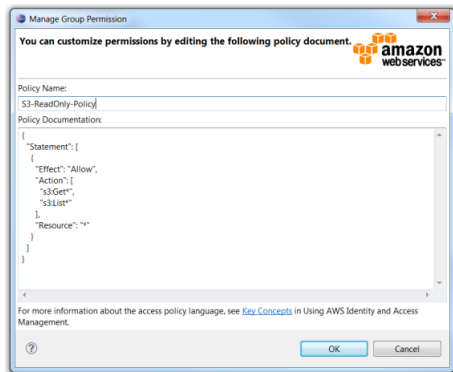**To create an IAM policy for a user, group, or role**

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.
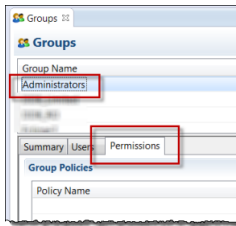
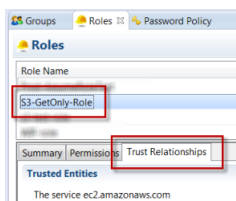3. On the right-hand side of the bottom pane, click the *Attach Policy* button.

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*.

   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*.
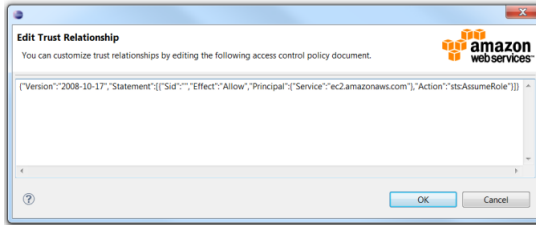
4.9.8 Set Password Policy

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*.
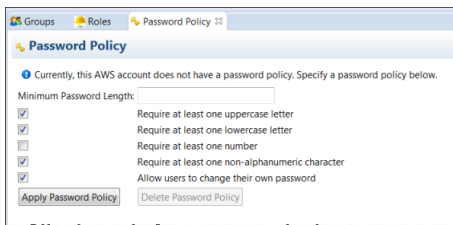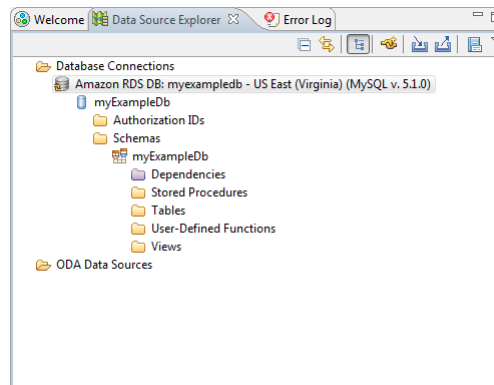
Fig. 4.17: Data Source Explorer

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: Jul 19, 2016

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*.

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 AWS Lambda with the AWS Toolkit for Eclipse*.

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*.

- 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?*.

- *Getting Started* 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.