

# Lambda Setting Review And Execute your lambda by S3 PUT

## 1. define value : you can change value as your setting

```
/* **define value ** */
/* *Check file name* */
public static final int d_CSV_FILE_NAME_LEN = 34;
public static final int d_CSV_FILE_DATE_LEN = 14;
public static final int d_CSV_FILE_SUFFIX_LENGTH = 4;
public static final int d_CSV_FILE_PREFIX_LENGTH = 4;
public static final String d_CSV_FILE_PREFIX = "csv/";
public static final String d_CSV_FILE_SUFFIX = ".csv";
public static final String d_CSV_FILE_DATE_FORMAT = "yyyy-MM-dd HH:mm:ss"; Date Format
public static final String d_TARGET_CONN_URL = "http://218.211.35.215:8080/api/api_delivery.php"; Target URL Location

/* *Error Code* */
public static final byte d_OK = 0x00;
public static final byte d_DATE_FORMAT_ERROR = 0x01;
public static final byte d_FILE_LEN_ERROR = 0x02;
public static final byte d_FILE_PREFIX_ERROR = 0x03;
public static final byte d_FILE_SUFFIX_ERROR = 0x04;
public static final byte d_FILE_DATE_LEN_ERROR = 0x05;

/* *SOMETHING* */
private static String d_LINE_FEED = "\r\n";
private static String d_FONT = "UTF-8";

private static final String d_POST_HEADER_CONNECT = "keep-alive"; POST package's HEADER FIELD
private static final String d_POST_HEADER_CONTENT_TYPE = "multipart/form-data";
private static final String d_POST_HEADER_ACCEPT = "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/";
private static final String d_POST_HEADER_ACCEPT_LANGUAGE = "zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7";

private static final String d_POST_BODY_CONTENT_TYPE = "application/octet-stream";
private static final String d_POST_BODY_FILES_PARM = "Upload"; POST package's BODY FIELD
private static final String d_POST_BODY_POST_PARM = "CMD";
private static final String d_POST_BODY_USERNAME = "admin";
private static final String d_POST_BODY_PASSWORD = "castles8418";
private static final String d_POST_BODY_FUNCTION_NAME = "CREATEDELIVERY";
private static final String d_POST_BODY_CREATEMODE = "2";

private static final String d_CONN_URL_REQUEST_METHOD = "POST"; HTTP Request Method

private static final String d_POST_BOUNDARY = "----WebKitFormBoundary"+Long.toHexString(System.currentTimeMillis());

/* **Process ** */

private static final String d_TAG_CHECK_FILENAME = "CHECK FILE NAME:";
private static final String d_PROCESS_CHECK_FILE_NAME = "-----Strat Checking file name-----";
private static final String d_PROCESS_CHECK_END = "-----Check File Name End-----";
private static final String d_PROCESS_CHECK_CORRECT = "---File Name Correct---";
private static final String d_PROCESS_POST = "-----Start Post to URL-----";
private static final String d_POST_STATUS = "HTTP POST Status:";

private static final String d_STATUS = "Return Code:";
```

## 2. s3 and lambda' s communication

When someone upload file to S3,it will trigger Lambda you already set.

Then,lambda function will check file name whether conform with the regulations

E.g : There is a file Named '20180720101112\_delInst.csv' .

Then suffix is .csv and date is 20180720101112 .

Check file name step :

- 1.Check suffix+date length if correct.
- 2.Check suffix if same value as **d\_CSV\_FILE\_SUFFIX**
- 3.Check date length if correct.
- 4.Check date if same value as **d\_CSB\_FILE\_DATA\_FORMAT**

5.And if not conform with the regulations ,it will return error code and log.

```
        }catch (ParseException e) {
            bReturnCode = d_DATE_FORMAT_ERROR;
            context.getLogger().log("DATE_FORMAT_ERROR");
        }

        }else {
            bReturnCode = d_FILE_DATE_LEN_ERROR;
            context.getLogger().log("FILE_DATE_LEN_ERROR");
        }
    }else {
        bReturnCode = d_FILE_SUFFIX_ERROR;
        context.getLogger().log("FILE_SUFFIX_ERROR");
    }
}
}else {
    bReturnCode = d_FILE_PREFIX_ERROR;
    context.getLogger().log("FILE_PREFIX_ERROR");
}
}
}else {
    bReturnCode = d_FILE_LEN_ERROR;
    context.getLogger().log("d_FILE_LEN_ERROR");
}
```

### 3. Lambda and WEB API ' Communication

1.if file name formate confirm is correct,it will return **d\_OK** .

Then ,use `HttpURLConnection` API to start post .

And according **d\_TARGET\_CONN\_URL** to post package to assign location

→(**MalformedURLException** · when URL format error it will generate LOG)

→(**Exception** · any exception happen it will generate error LOG)

2. Prepare post package and post it to assign location

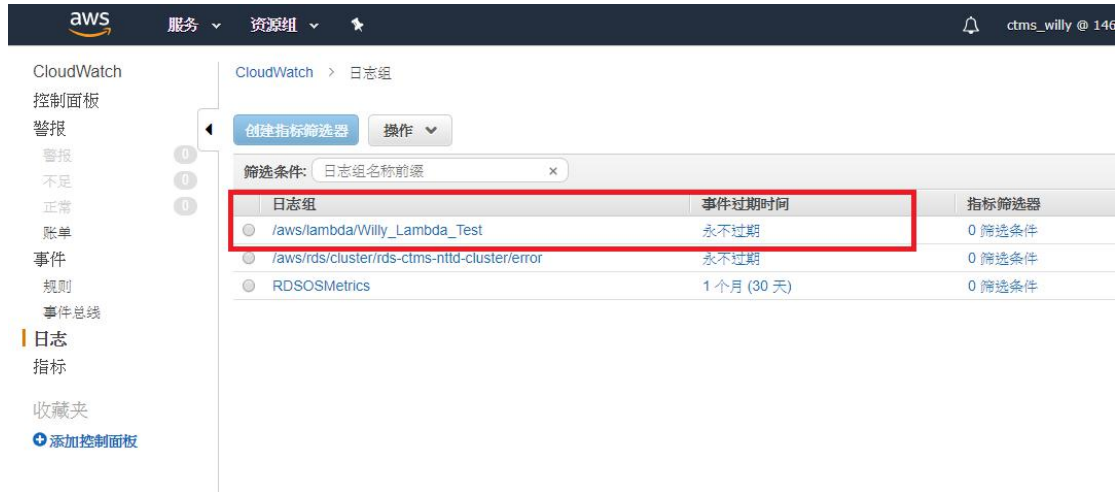
→(**IOException** · when use **OutputStream** to post to URL have some exception will generate error log)

3.Use `HttpURLConnection.getResponseCode();` to get connection Status and confirm if post method is correct execute.

```
}catch (MalformedURLException e1) {
    System.out.println("URL Foramt Error!!!");
}catch (IOException e2) {
    System.out.println("UPload failed!!!");
    e2.printStackTrace();
}catch (Exception e3) {
    System.out.println("InetAddress format error");
    e3.printStackTrace();
}
```

#### 4. CloudWatch Log Generate

Use `System.out.println("context");` or `context.getLogger().log("context");` on your lambda function ,then function will generate log and save to CloudWatch service.



Go into Log Group you will see Log streams that generated by lambda function , as below picture.

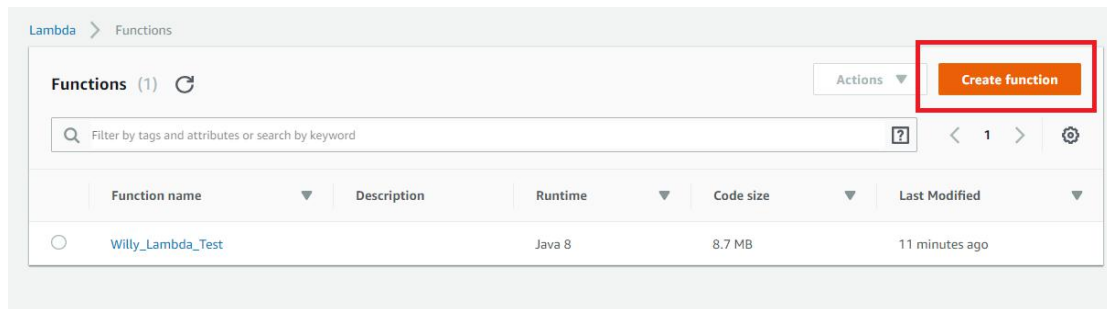


click any date if log group , you will see detail of log

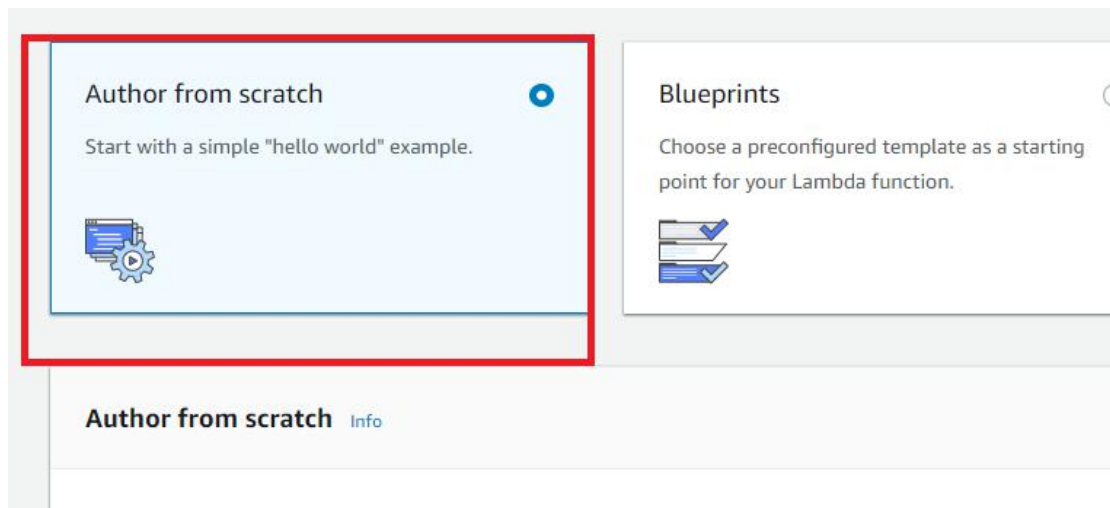


## 5. Build Lambda Function

Go to AWS Lambda Service, and click 'Create function' button.



And, choose 'Author from scratch'



Enter the function name and Runtime Environment 'Java 8'

**Author from scratch** [Info](#)

Name

Runtime

C# (.NET Core 2.0)

C# (.NET Core 2.1)

Go 1.x

Java 8

Node.js 4.3

Node.js 6.10

Node.js 8.10

[Learn more](#)

[Cloudwatch logs](#)

Choose the Role that already existing or Create new role.

#### Role

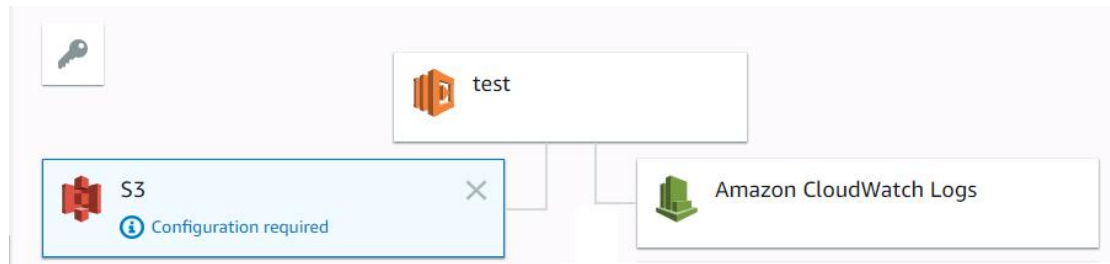
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more about roles.](#)

Choose an existing role	▼
Choose an existing role	
Create new role from template(s)	Cloudwatch Logs f
Create a custom role	

And Choose the Trigger Service. In this step please choose 'S3' .

▼ Designer
CloudWatch Logs
CodeCommit
Cognito Sync Trigger
DynamoDB
Kinesis
S3
SNS
SQS

Then, you can see S3 is added on your display.



Next, you have to choose Bucket that you already build in S3.

**Bucket**  
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

castles-dlc-20180808 ▼

**Event type**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

PUT ▼

PUT
POST
COPY
Complete Multipart Upload

Choose Event type 'PUT' and if you want to set special condition, you can set Prefix and Suffix value.

E.g. If you set Prefix as *csv/* and Suffix as *.csv*, then your trigger will happen when people upload *.csv* file into *csv* folder on S3.

**Prefix**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

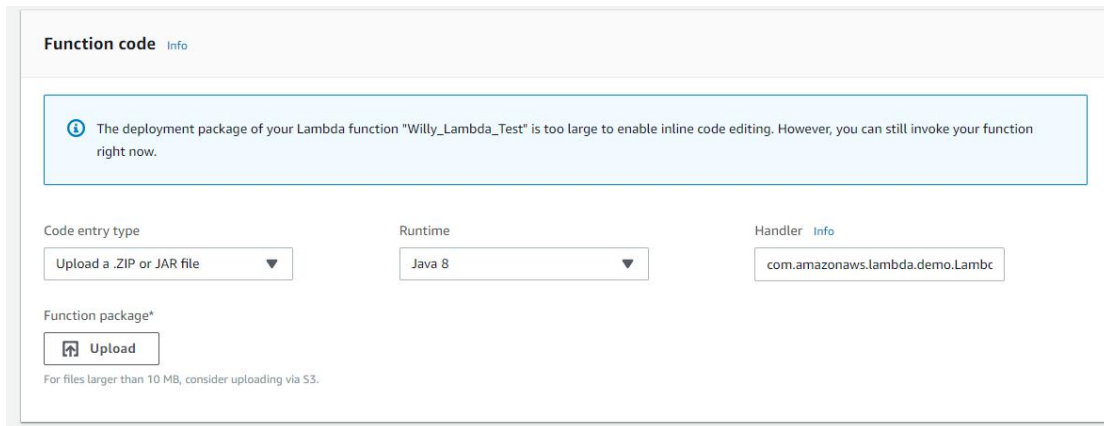
*e.g. images/*

**Suffix**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

*e.g. .jpg*



Now, you can upload you code by JAR or .ZIP



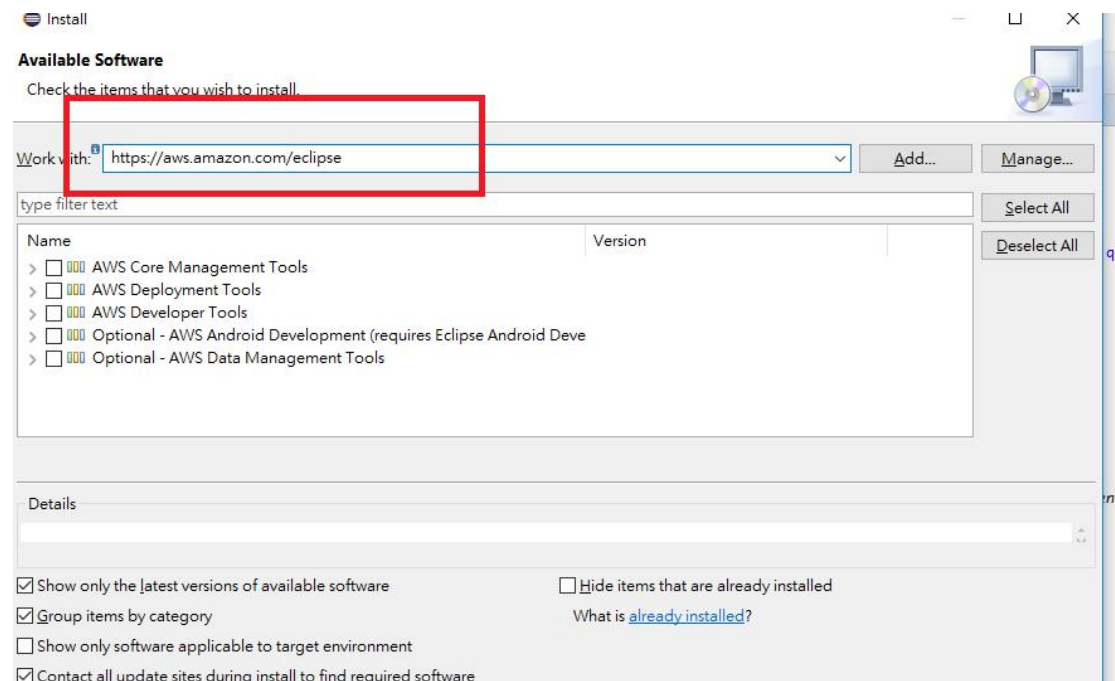
But, there are better ways to create lambda function : Eclipse Plugin AWS Toolkit

You have to install eclipse and set up your JRE and you should use ' **JavaSE 1.8**' as your environment.

Then go to **HELP >> Install new software...** to open dialog like below.

And enter ' <https://aws.amazon.com/eclipse>' to input box Work with.

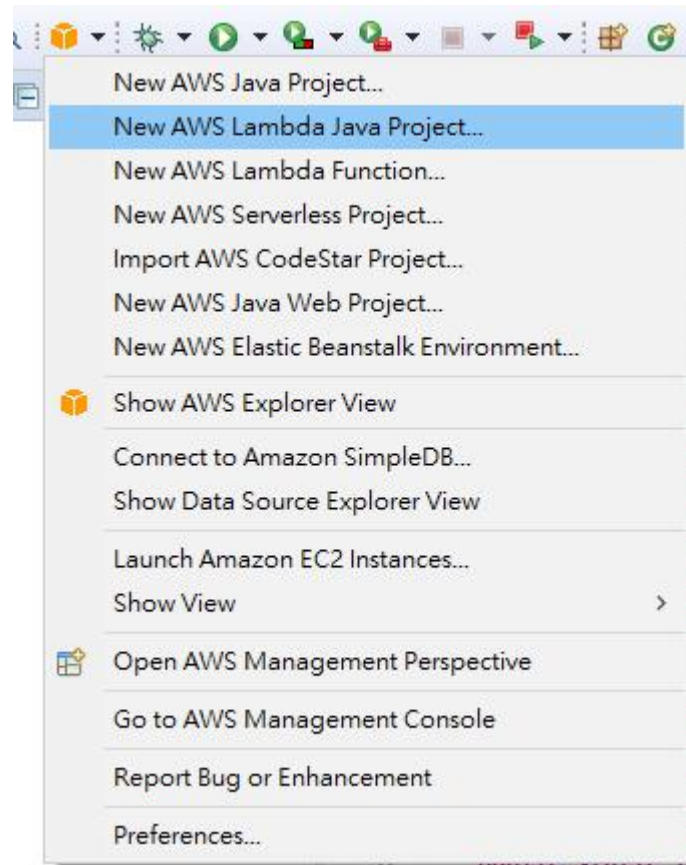
Then download the ALL AWS plugin tool to your eclipse.



If install was success you will see the AWS icon in you tool bar.



Then click it and create New AWS Lambda Java Project.





Then finish all field and press finish button ,you can build you lambda project.

**Create a new AWS Lambda Java project**  
Create a new AWS Lambda Java project in the workspace

Project name:

Maven configuration

Group ID:

Artifact ID:

Version:

Package name:

Lambda Function Handler

Each Lambda function must specify a handler class which the service will use as the entry point to begin execution.  
[Learn more about Lambda Java function handler.](#)

Class Name:

Input Type:

A hello world Lambda function.

Preview:

```
package com.amazonaws.lambda.demo;

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

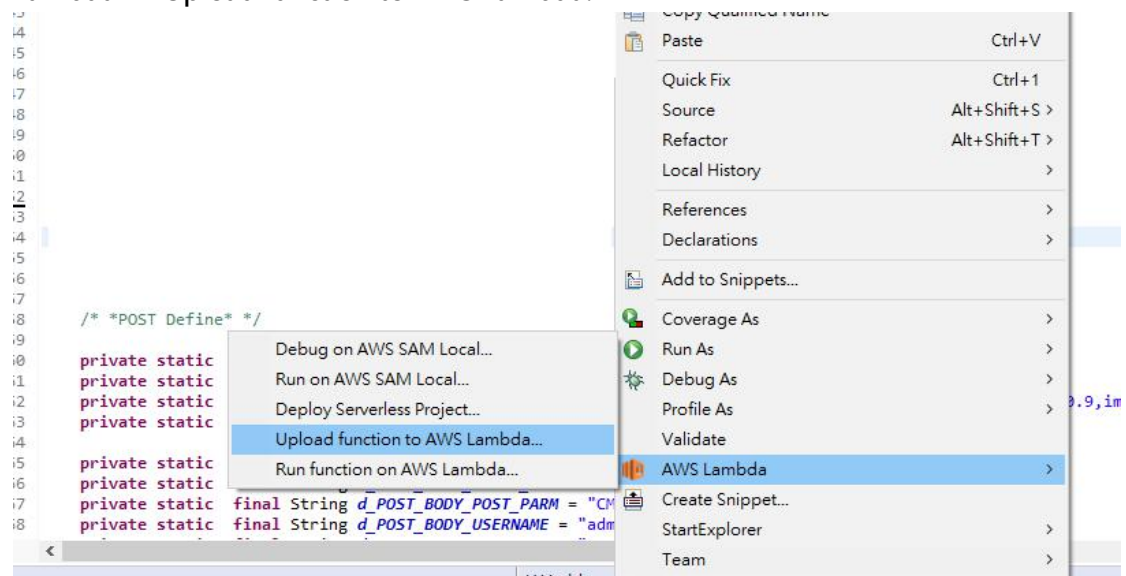
public class LambdaFunctionHandler implements RequestHandler<Object, String> {

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

        // TODO: implement your handler
    }
}
```

After you build you project ,you can type your code just like before you code your normal java project.

After you finish your code , you can 'right click' blank space and click AWS Lambda>> Upload function to AWS Lambda.



After those setting ,you can go online and test you lambda by Upload CSV to S3 bucket.