

AWS Lambda Plugin

Plugin Information

View AWS Lambda [on the plugin site](#) for more information.

This plugin adds AWS Lambda invocation and deployment abilities to build steps and post build actions

Currently the plugin can deploy and invoke functions as a build step and post build action. When invoking a function it is possible to inject the output as Jenkins environment variables.

Github link: <https://github.com/XT-i/aws-lambda-jenkins-plugin>

Jenkins wiki link: <https://wiki.jenkins-ci.org/display/JENKINS/AWS+Lambda+Plugin>

Installation

Look for the AWS Lambda plugin in the available plugins after clicking "manage jenkins" and "manage plugins".

Updates	Available	Installed	Advanced
Install ↓	Name	Version	
Artifact Uploaders			
<input type="checkbox"/>	Appaloosa Plugin Publish your mobile applications (Android, iOS, ...) to the appaloosa-store.com platform.	1.4.2	
<input type="checkbox"/>	AppThwack Plugin A Jenkins CI plugin for running Android/iOS mobile tests on 100s of real devices using AppThwack.	1.9	
<input type="checkbox"/>	ArtifactPromotionPlugin Using this plugin you can promote artifacts by moving release candidates into release repositories.	0.3.4	
<input type="checkbox"/>	ArtifactDeployer Plugin This plugin makes it possible to copy artifacts to remote locations.	0.33	
<input type="checkbox"/>	AWS Lambda Plugin This plugins adds a post build action that deploys a zip file or folder to AWS Lambda	0.2.1	

IAM setup

For deployment you'll need access to the GetFunction, CreateFunction, UpdateFunctionCode and UpdateFunctionConfiguration Lambda commands. You'll also need access to iam:PassRole to attach a role to the Lambda function.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1432812345671",
      "Effect": "Allow",
      "Action": [
        "lambda:GetFunction",
        "lambda:CreateFunction",
        "lambda:UpdateFunctionCode",
        "lambda:UpdateFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:REGION:ACCOUNTID:function:FUNCTIONNAME"
      ]
    },
    {
      "Sid": "Stmt14328112345672",
      "Effect": "Allow",
      "Action": [
        "iam:Passrole"
      ],
      "Resource": [
        "arn:aws:iam::ACCOUNTID:role/FUNCTIONROLE"
      ]
    }
  ]
}

```

For invocation you only need access to InvokeFunction.

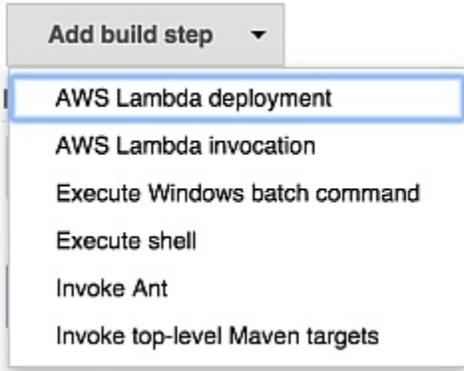
```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt14328112345678",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "arn:aws:lambda:REGION:ACCOUNTID:function:FUNCTIONNAME"
      ]
    }
  ]
}

```

AWS Lambda function deployment

After creating a job you can add a build step or post build action to deploy an AWS Lambda function.



Due to the fact that AWS Lambda is still a rapid changing service we decided not to have select boxes for input. The AWS Access Key Id, AWS Secret Key, region and function name are always required. All other fields depend on the update mode.

If the update mode is Code you also need to add the location of a zipfile or folder. Folders are automatically zipped according to the [AWS Lambda documentation](#). You can also choose to deploy a function already on S3, if the bucket is in the same region as the Lambda function.

```
s3://bucket/key or s3://bucket/key?versionId=ABCDEF123
```

For the Configuration update mode you need the role and handler. If you want to diverge from the defaults add the memory and timeout values.

When choosing the Both update mode, both UpdateFunctionCode and UpdateFunctionConfiguration are performed.

If the function has not been created before the plugin will try to do a CreateFunction call, which needs all fields previously mentioned in addition to the runtime value.

The update mode value is ignored if the function does not exist yet, but it will take effect in future builds.

☰ AWS Lambda deployment

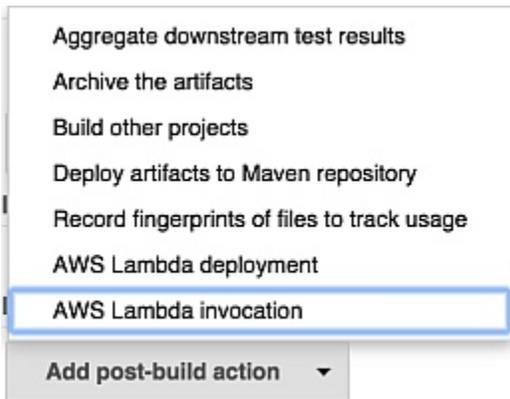
Deploy AWS Lambda functions

AWS Access Key Id	<input type="text" value="AKIA1234567812345678"/>
AWS Secret Key	<input type="password" value="....."/>
AWS Region	<input type="text" value="eu-west-1"/>
Function Name	<input type="text" value="analyze-csv"/>
Description	<input type="text" value="Deployed with AWS Lambda plugin: \${BUILD_ID}"/>
Role	<input type="text" value="arn:aws:iam::123456123456:role/lambda_exec_role"/>
Artifact Location (Zip file or directory)	<input type="text" value="src/main/lambda"/>
Handler Name	<input type="text" value="analyze-csv.handler"/>
Memory Size	<input type="text" value="1024"/>
Timeout	<input type="text" value="30"/>
Runtime	<input type="text" value="nodejs"/>
Update Mode	<input type="text" value="Code and configuration"/>

Delete

AWS Lambda function invocation

To invoke a function once again open up the add build step or post build action menu.



You need to add the AWS Access Key Id, AWS Secret key, region and function name. Optionally you can add a payload that your function expects.

If you enable the Synchronous checkbox you will receive the response payload that can be parsed using the Json Parameters. You will also get the logs from Lambda into your Jenkins console output.

AWS Lambda invocation

Invoke AWS Lambda functions

AWS Access Key Id	<input type="text" value="redacted"/>	
AWS Secret Key	<input type="password" value="....."/>	
AWS Region	<input type="text" value="eu-west-1"/>	
Function Name	<input type="text" value="invalidate-cloudfront"/>	
Payload	<pre>["cloudfront-invalidate":["web/*","docs/*"]]</pre>	
Synchronous	<input checked="" type="checkbox"/>	
Json Parameters	<input type="button" value="Add"/>	

The json parameters allow you to parse the output from the lambda function. The parsed value will then be injected into the Jenkins environment using the chosen name.

An empty jsonPath field allows you to inject the whole response into the specified environment variable.

Json Parameters

Name 

Json Path 

Delete

Name 

Json Path 

Delete

Add

Examples:

```
{
  "key1": "value1",
  "array1": [
    {
      "arraykey": "arrayvalue"
    },
    {
      "arraykey": "arrayvalue2"
    }
  ]
}
```

\$.key1 => value1

\$.array1[1].arraykey => arrayvalue2

More info about JsonPath:

github link: <https://github.com/jayway/JsonPath>

try out expressions: [http://jsonpath.herokuapp.com/?path=\\$.store.book](http://jsonpath.herokuapp.com/?path=$.store.book)

These environment variables can be used as parameters in further build steps and actions which allow a Lambda function to have a deciding factor in the deployment process.

Job build result

On the job build result page you'll get a summary of all deployed and invoked functions and their success state.



Revision: 2559
No changes.



Started by anonymous user



Invoked Lambda: invalidate-cloudfront



Deployed Lambda: analyze-csv