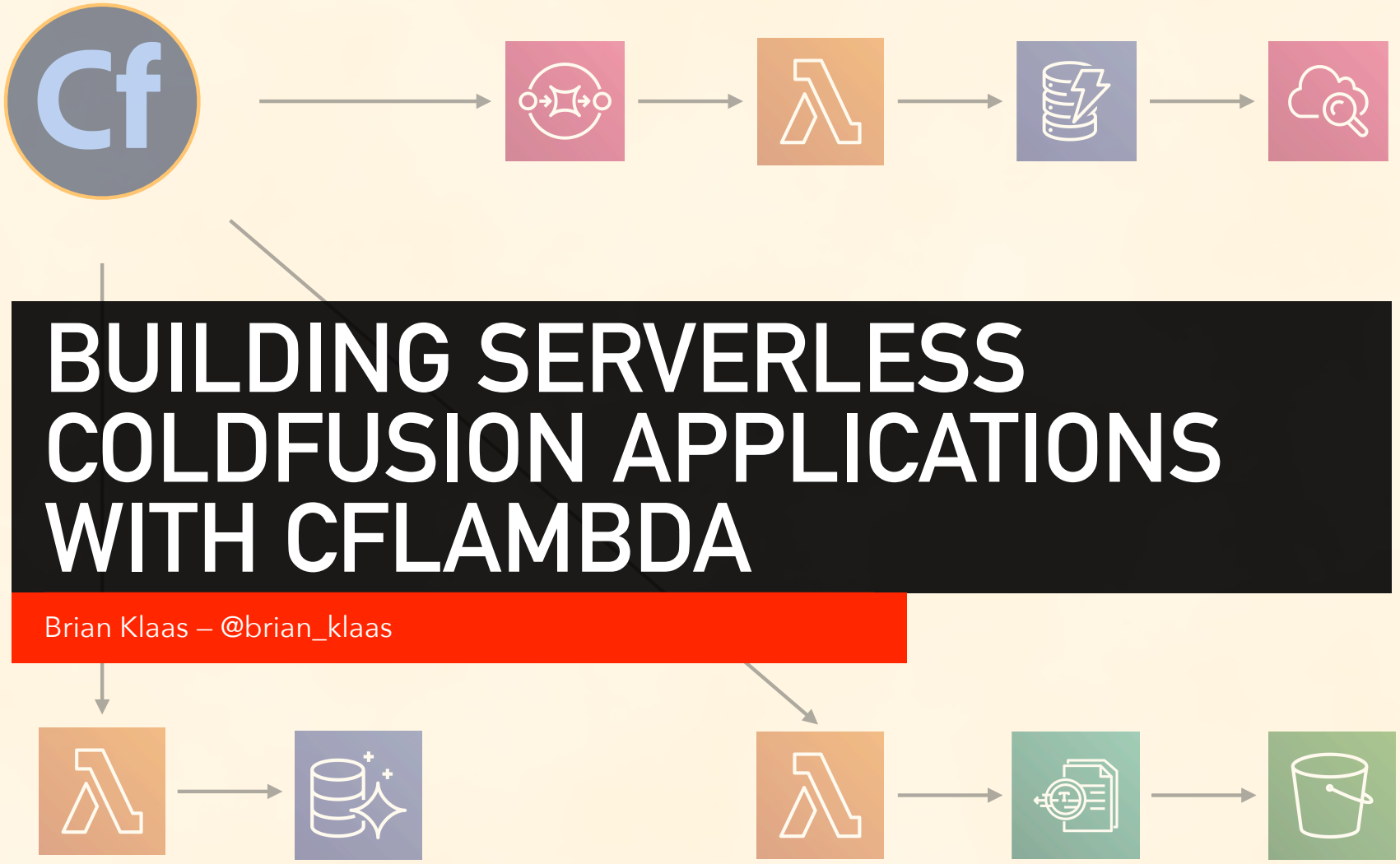


BUILDING SERVERLESS COLDFUSION APPLICATIONS WITH CFLAMBDA

Brian Klaas – @brian_klaas

WOULDN'T IT BE NICE?

SERVERLESS != FUNCTIONS

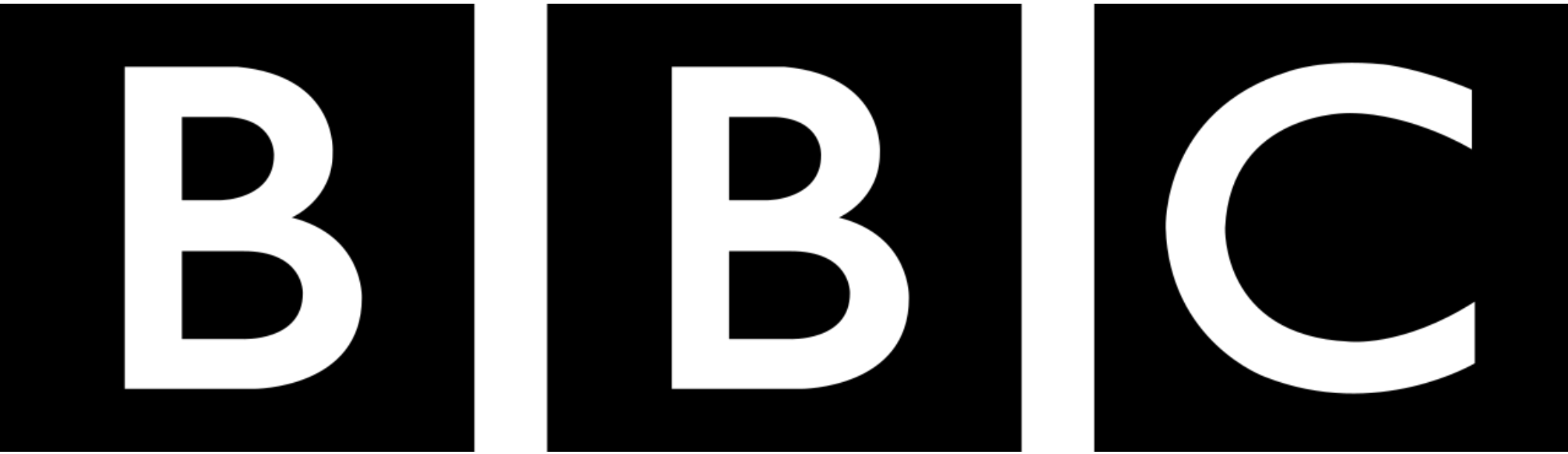
SERVERLESS = SCALABLE, PAY-AS-YOU-GO, FULLY MANAGED SERVICES

"If the first era of the cloud is defined by primitives, its days are coming to an end. The next is likely to be defined by, as the computing industry has since its inception, the abstractions we build on top of those primitives."

- STEPHEN O'GRADY, REDMONK

SKATE TO WHERE THE PUCK WILL BE.

LEGO SHOP + CRM



BBC + BBC WORLD

AMAZON: SHOPPING CART + PRIME DAYS

Yes, we know it's not all unicorns and rainbows.

Yes, we know.


BUT SERVERLESS STILL HAS SERV...
THAT'S NOT THE POINT
imgflip.com

Yes, we know.

Yes, we know.


CAN I HAVE YOUR USE CASE DIAGRAM
I'LL "CHANGE" IT
memegenerator.net

SKATE TO WHERE THE PUCK WILL BE WITH COLDFUSION.

SKATE TO WHERE THE PUCK WILL BE WITH CFLAMBDA.

Cloud services are a core part of CF2021

AWS:

- Simple Storage Service (S3)
- Simple Queue Service (SQS)
- Simple Notification Service (SNS)
- DynamoDB

Azure:

- Azure Blob Storage
- Azure Service Bus

cflambda is part of CF2021

A mini ColdFusion runtime that lets you create AWS Lambda services with ColdFusion

The 15 Step cflambda

1. Get a copy of the cflambda ZIP
2. Put your code inside <cflambda>/cfusion/wwwroot
3. Update permissions on all files in <cflambda>/cfusion/bin to allow execute privileges
4. Run cfpm to install needed modules
5. Run cfsetup to configure the server settings (if needed)
6. Run the lambdazip utility
7. Get the resulting .zip file from <cflambda>/cfusion/bin/dist
8. Upload the .zip file to S3
9. Create a new Java 11 Lambda function in the AWS Console
10. Link to the .zip file on S3
11. Edit Lambda configuration to increase RAM
12. Specify the appropriate Java class handler
13. Add an environment variable to point to the handler function name
14. Configure a test event
15. Run your function for fun and profit!

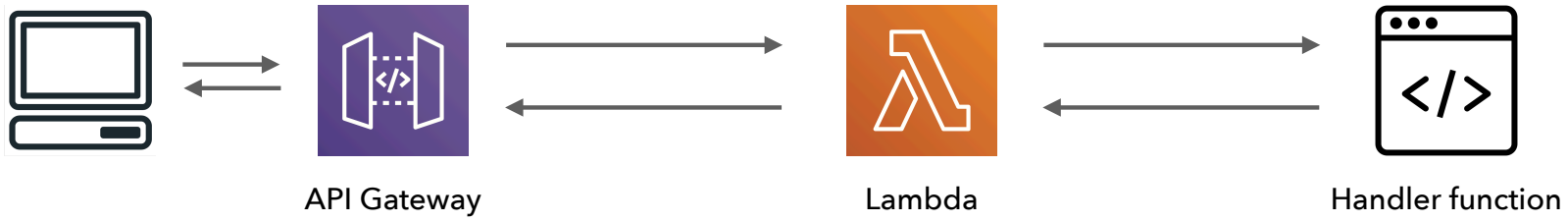
TOOLING AND DEBUGGING ARE TERRIBLE ACROSS THE ENTIRE SERVERLESS SPACE

TWO WAYS TO INVOKE A LAMBDA FUNCTION

Sync vs. Async

Async Invocation

Synchronous Invocation



Invoking a Lambda function

I'm going to focus on async invocation today.

GETTING UP AND RUNNING IN AWS IS COMPLICATED

Running Your cflambda Function

1. Get a copy of the cflambda ZIP
2. Put your code inside <cflambda>/cfusion/wwwroot
3. Update permissions on all files in <cflambda>/cfusion/bin to allow execute privileges
4. Run cfpm to install needed modules
5. Run cfsetup to configure the server settings (if needed)
6. Run the lambdazip utility
7. Get the resulting .zip file from <cflambda>/cfusion/bin/dist
8. Upload the .zip file to S3
9. Create a new Java 11 Lambda function in the AWS Console
10. Link to the .zip file on S3
11. Edit Lambda configuration to increase RAM
12. Specify the appropriate Java class handler
13. Add an environment variable to point to the handler function name
14. Configure a test event
15. Run your function for fun and profit!

The “Complex” Calculator CFC

```
component output="false" hint="A complex, time-consuming calculator" {

  public numeric function performComplexCalculation(any event, any context) {

        var returnValue = 0;

        var multiplier = Val(event.multiplier);

        cfhttp(method="GET", charset="utf-8", url="https://finnhub.io/api/v1/quote?symbol=AMZN", result="currentAmazonStockInfo") {};

        if (currentAmazonStockInfo.statusCode eq "200 OK") {
            var stockInfo = deserializeJSON(currentAmazonStockInfo.fileContent);
            var currentPrice = stockInfo.c;
            returnValue = multiplier * stockInfo.c;
        }

    return returnValue;
  }

}
```

The “Complex” Calculator CFC

```
component output="false" hint="A complex, time-consuming calculator" {

  public numeric function performComplexCalculation(any event, any context) {

        var returnValue = 0;

        var multiplier = Val(event.multiplier);

        cfhttp(method="GET", charset="utf-8", url="https://finnhub.io/api/v1/quote?symbol=AMZN", result="currentAmazonStockInfo") {};

        if (currentAmazonStockInfo.statusCode eq "200 OK") {
            var stockInfo = deserializeJSON(currentAmazonStockInfo.fileContent);
            var currentPrice = stockInfo.c;
            returnValue = multiplier * stockInfo.c;
        }

    return returnValue;
  }

}
```

All Lambda functions require an event and context argument.

The “Complex” Calculator CFC

```
component output="false" hint="A complex, time-consuming calculator" {

    public numeric function performComplexCalculation(any event, any context) {

            var returnValue = 0;

            var multiplier = Val(event.multiplier);

            cfhttp(method="GET", charset="utf-8", url="https://finnhub.io/api/v1/quote?symbol=AMZN", result="currentAmazonStockInfo") {};

            if (currentAmazonStockInfo.statusCode eq "200 OK") {
                var stockInfo = deserializeJSON(currentAmazonStockInfo.fileContent);
                var currentPrice = stockInfo.c;
                returnValue = multiplier * stockInfo.c;
            }

        return returnValue;
    }

}
```

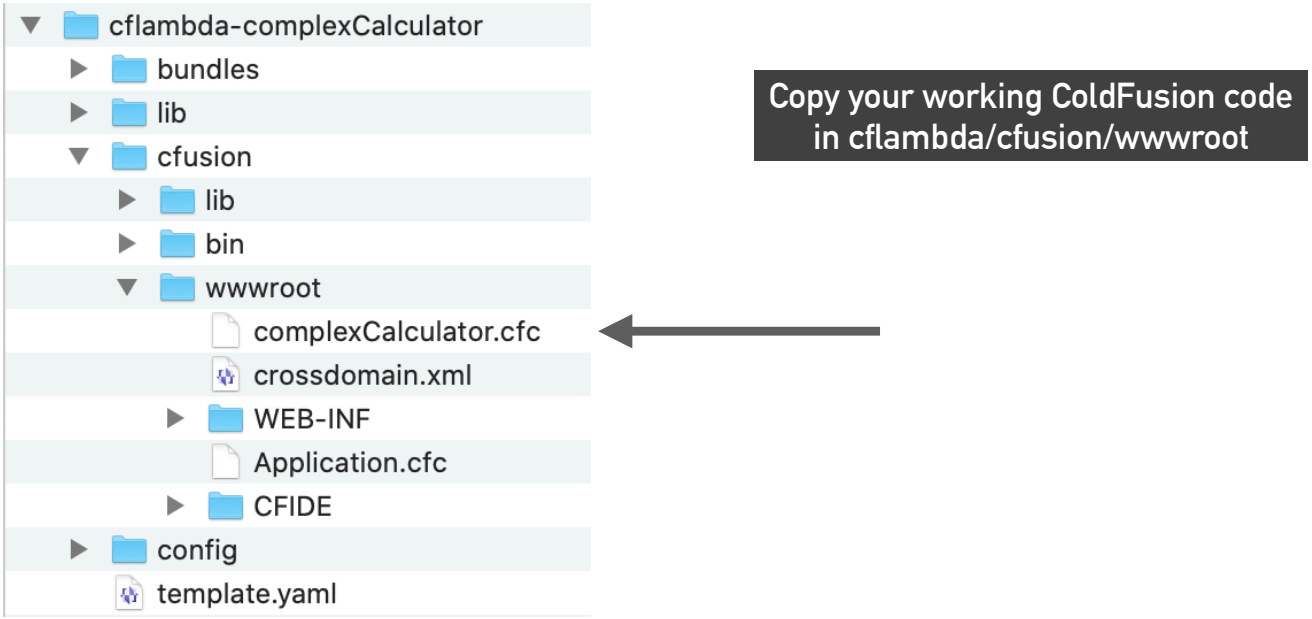
The event argument contains all incoming data to the function.

The “Complex” Calculator CFC

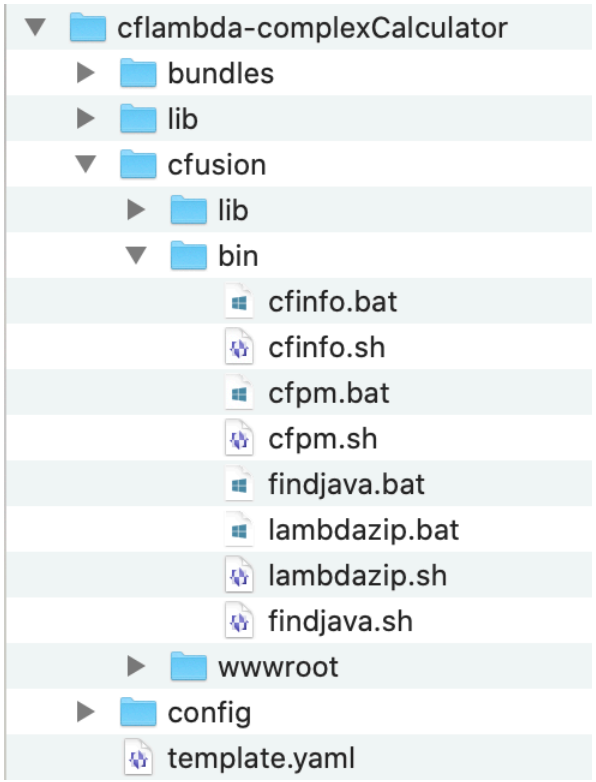
```
component output="false" hint="A complex, time-consuming calculator" {

   public numeric function performComplexCalculation(any event, any context) {

                var returnValue = 0;

                var multiplier = Val(event.multiplier);

                cfhttp(method="GET", charset="utf-8", url="https://finnhub.io/api/v1/quote?symbol=AMZN", result="currentAmazonStockInfo") {};

                if (currentAmazonStockInfo.statusCode eq "200 OK") {
                        var stockInfo = deserializeJSON(currentAmazonStockInfo.fileContent);
                        var currentPrice = stockInfo.c;
                        returnValue = multiplier * stockInfo.c;
                }

        return returnValue;
   }

}
```

The context argument contains information about the Lambda runtime. Example: context.getAwsRequestId()

The cflambda File Bundle



Run cfpm (ColdFusion Package Manager)



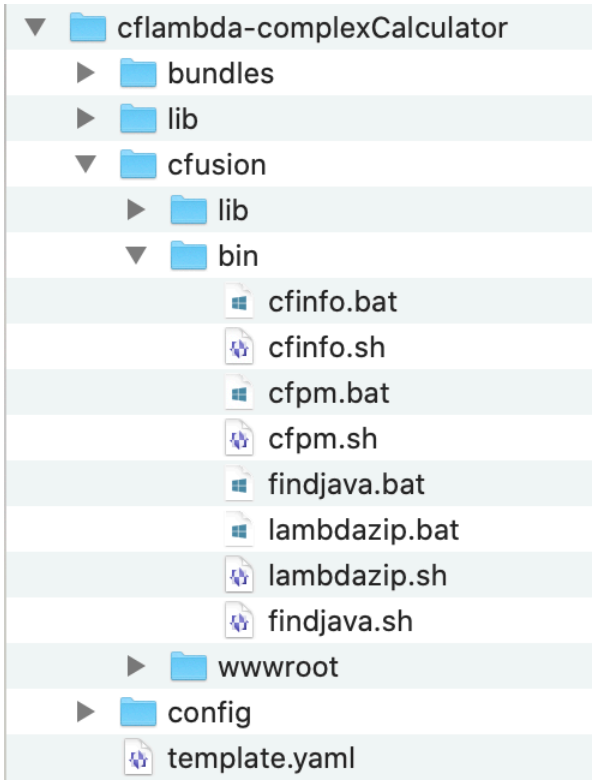
You'll need to set executable permission on this file first!

Run cfpm (ColdFusion Package Manager)

scanandinstall path/to/your/cflambda/cfusion/wwwroot
http://your.local.CF2021.server/

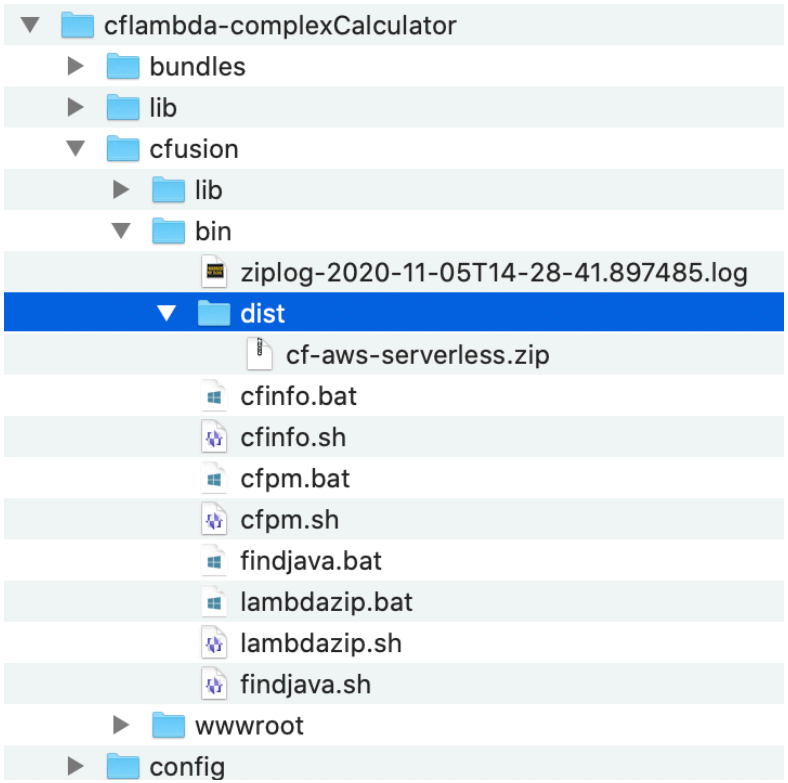
```
[BK-Home-iMac-2019:bin brianklaas$ ./cfpm.sh
cfpm>scanandinstall /Users/brianklaas/Downloads/cflambda-complexCalculator/cfusion/wwwroot
 http://127.0.0.1:8502/
The following packages will be installed : debugger,derby
debugger (2021.0.0.321466) package is already installed.
derby (2021.0.0.321466) package is already installed.


Scanning and installation completed.
cfpm>
```

Run lambdazip



You'll need to set executable permission on this file first!

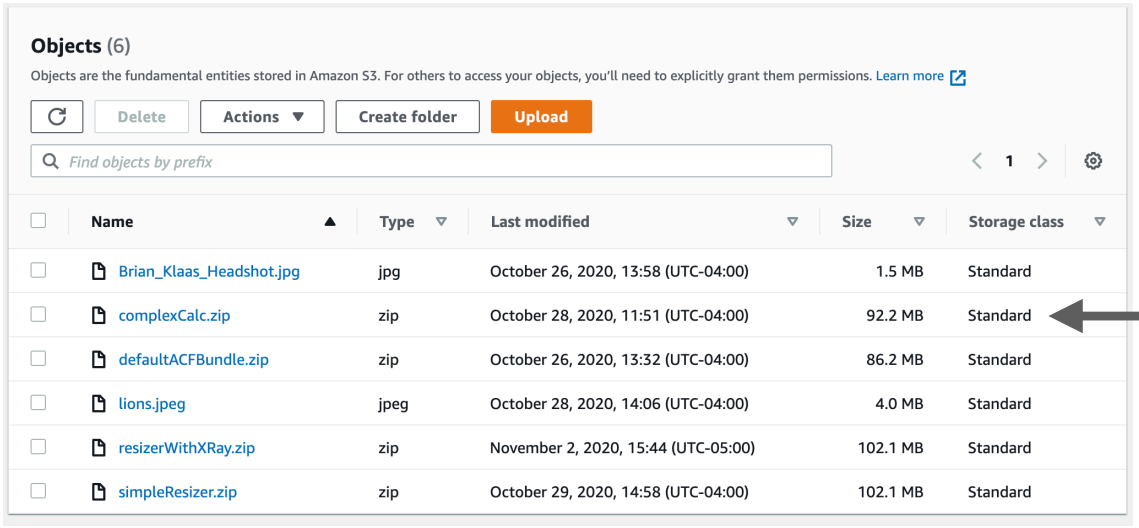
Get the resulting ZIP file



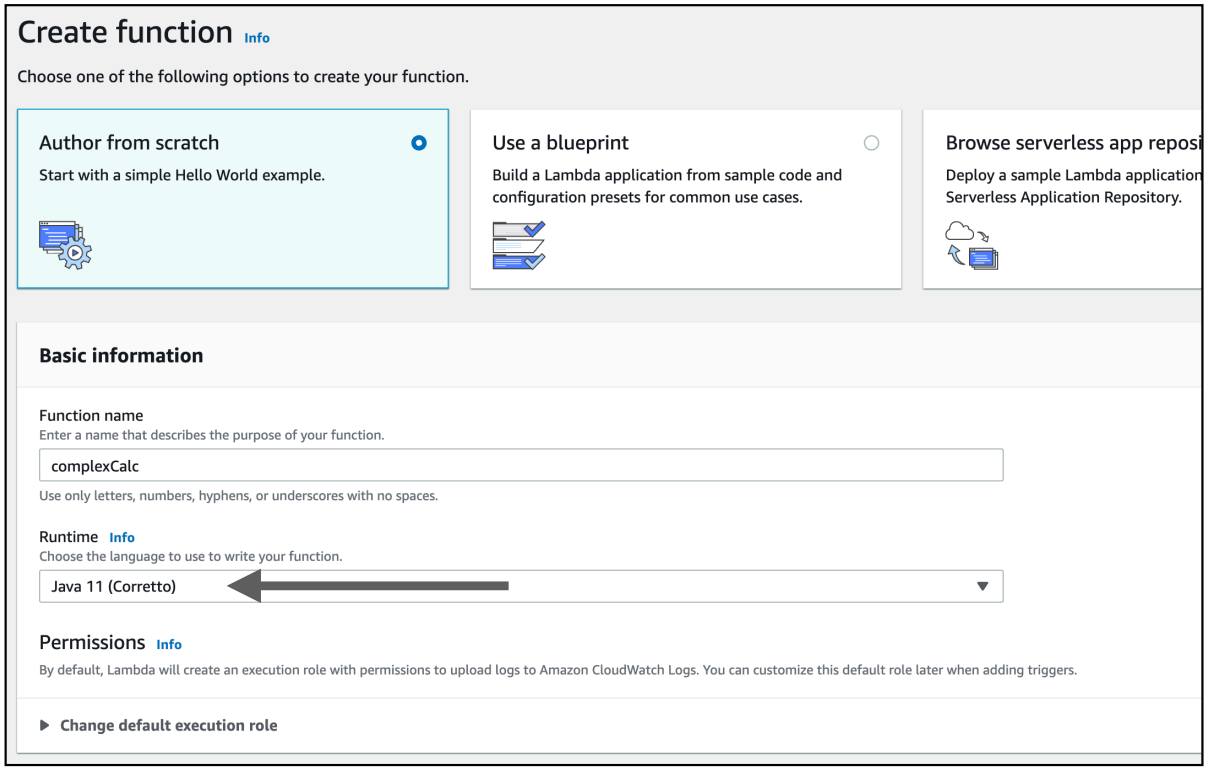
It's always named
cf-aws-serverless.zip

Upload the ZIP file to S3

The Lambda console has a max file size limit of 50MB. The smallest (current) lambdazip-generated bundle is 90MB.

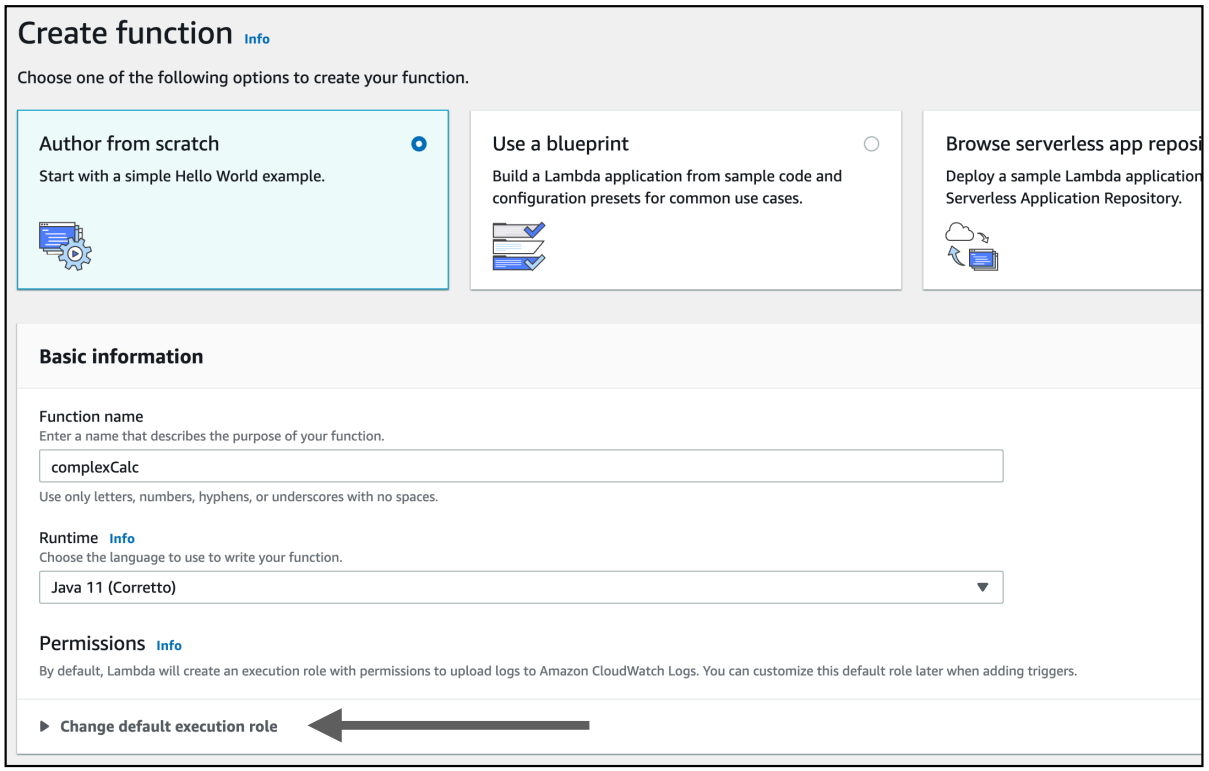


Create a New Lambda Function



cflambda functions use the Java 11 (Corretto) runtime

Are You Using the Right IAM Permissions?

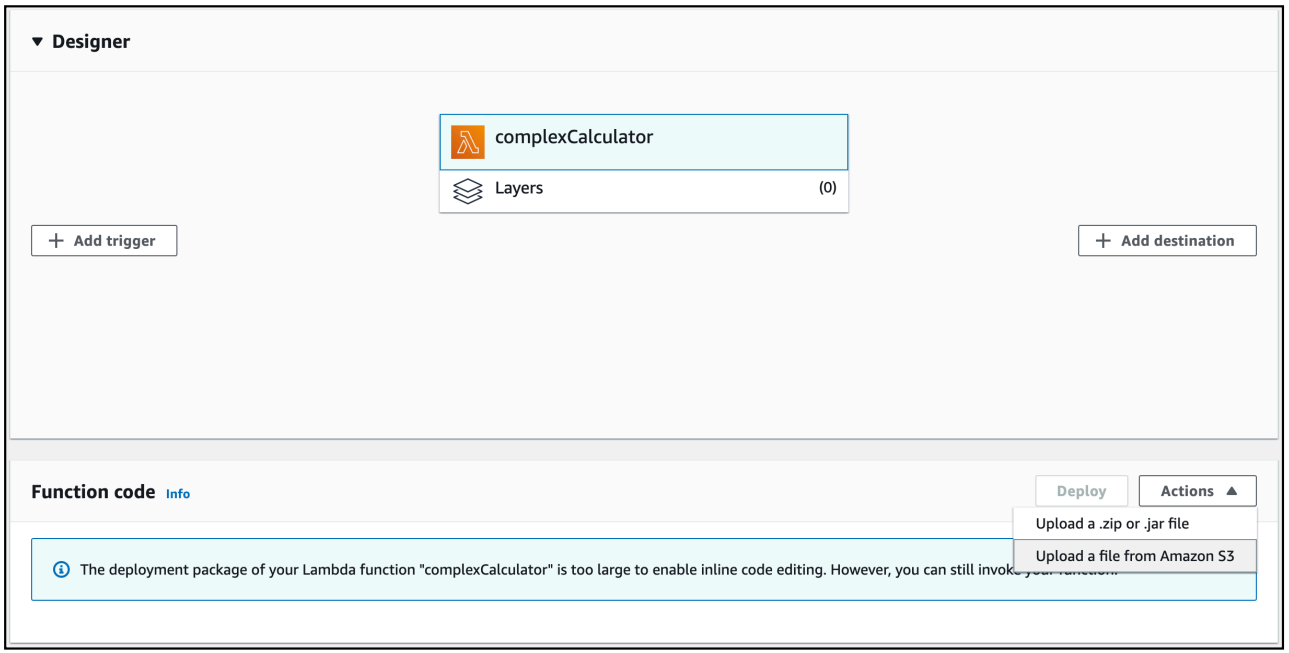


Make sure the Lambda function has permission to access all services required by the function!

Lambda functions run under IAM roles, not user accounts.

The minimum permission needed is logging to CloudWatch.

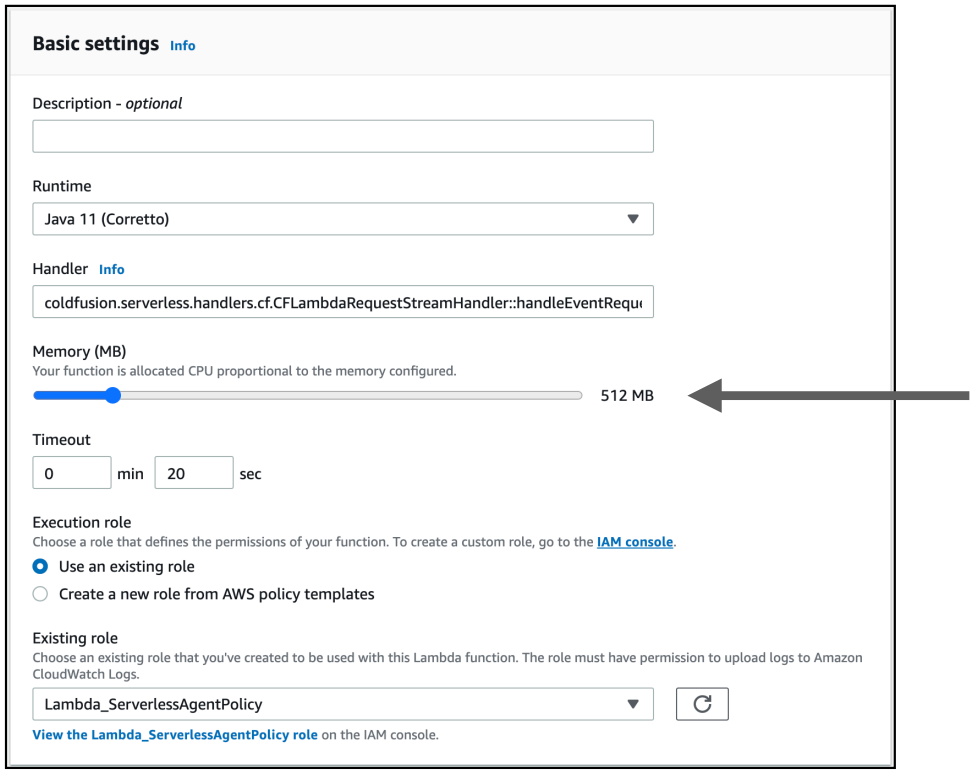
Link Your Function to the ZIP on S3



You'll need the full http path to your ZIP bundle in S3.

YOU HAVE TO RE-LINK TO THE ZIP IN S3 EVERY TIME YOU MAKE A CODE CHANGE

Edit “Basic Settings”: Memory

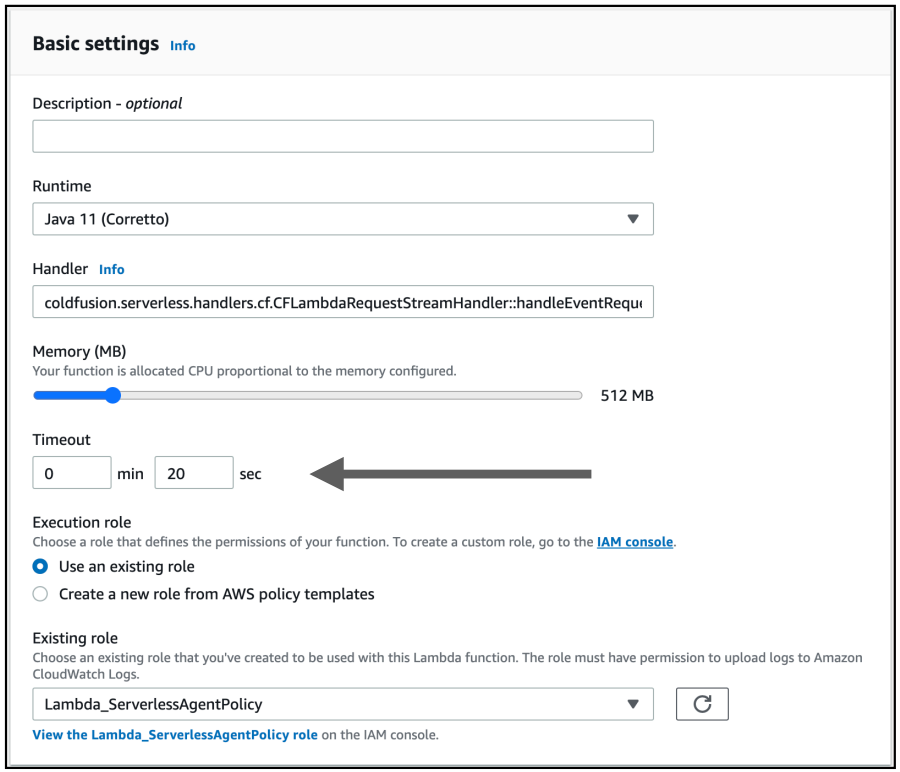


Lambda allocates CPU power linearly in proportion to the amount of memory configured.

Don’t over-allocate. This costs you money.

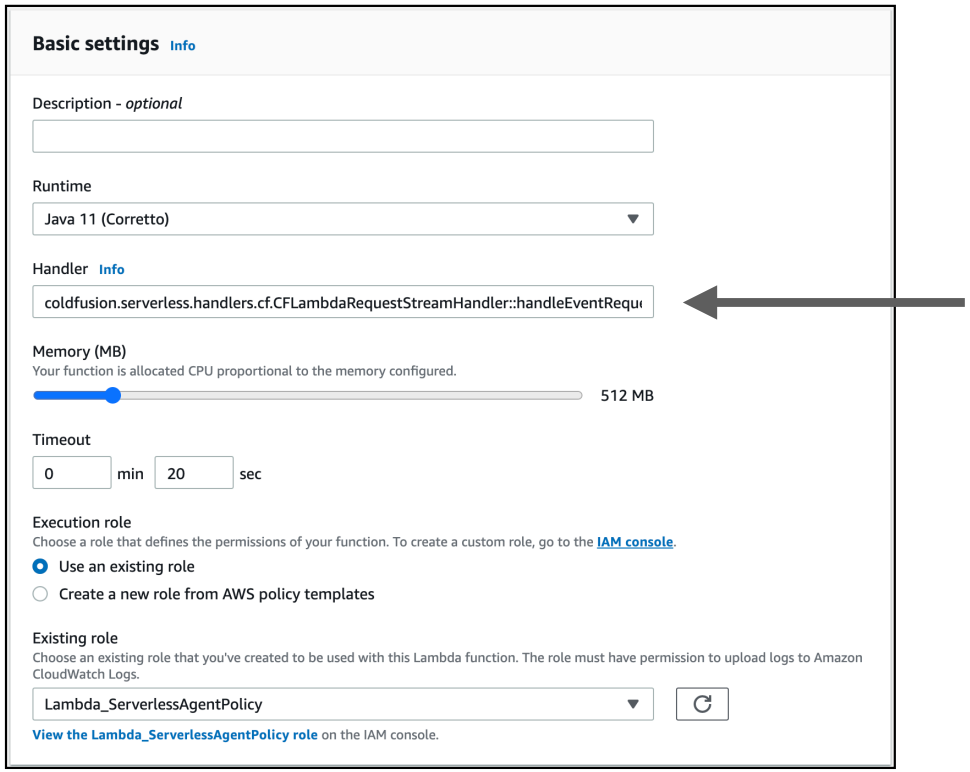
Use aws-lambda-power-tuning: https://github.com/alexcasalboni/aws-lambda-power-tuning

Edit “Basic Settings”: Timeout



Be generous with the timeout.
Cold starts can take a long time.

Edit “Basic Settings”: Handler

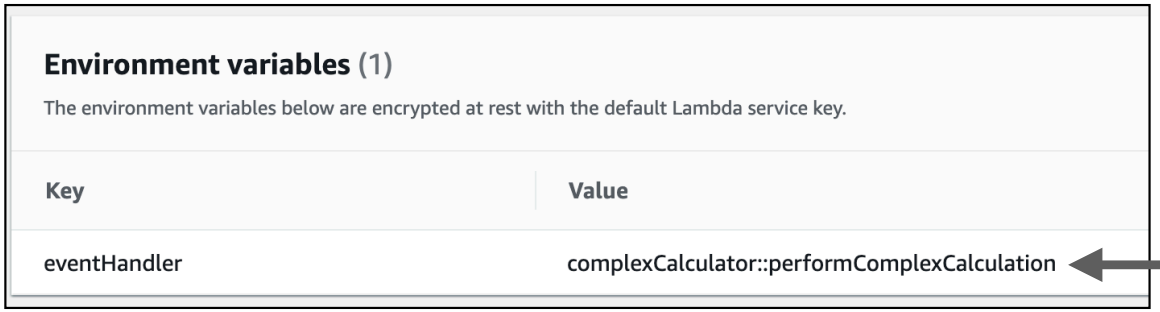


The “handler” is the java servlet, not your CFC.

coldfusion.serverless.handlers.cf.
CFLambdaRequestStreamHandler::handleEventRequest

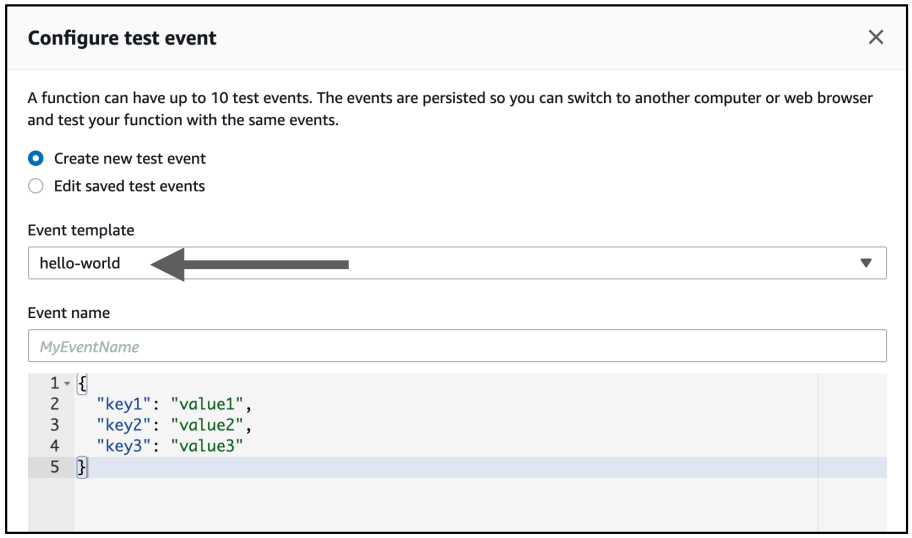
Add the Function as an Environment Variable

The eventHandler is
the function in your CFC.

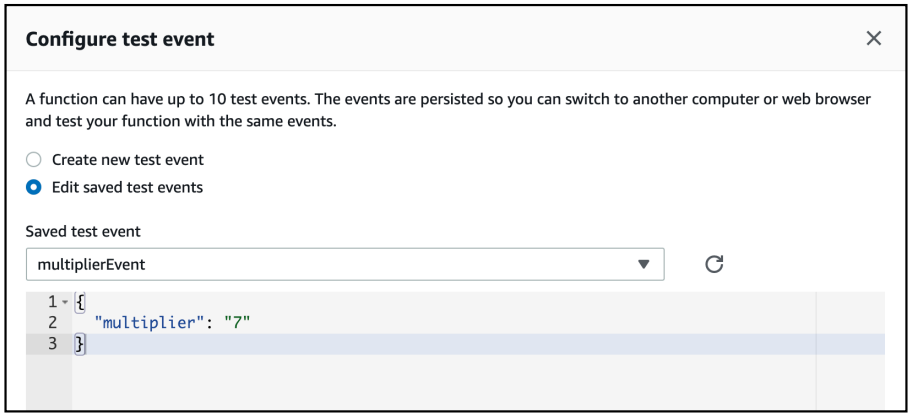


Not providing an eventHandler environment variable for asynchronous execution defaults to executing the serverlessEvent() method present in the default Application.cfc provided as part of the cflambda bundle.

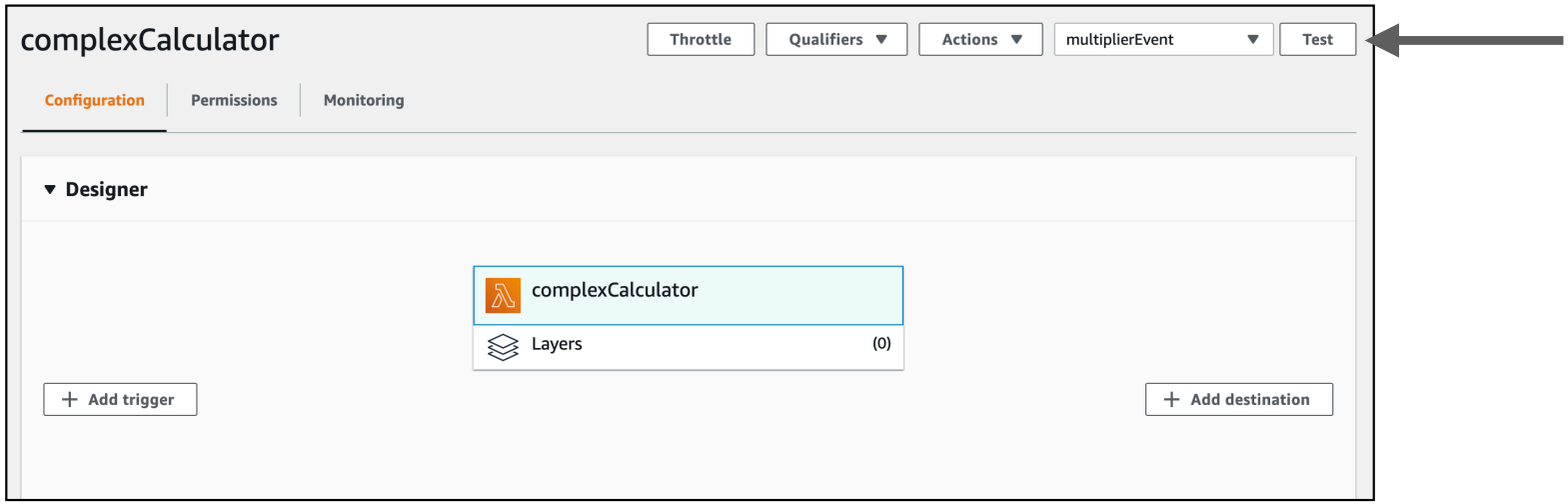
Configure a Test Event



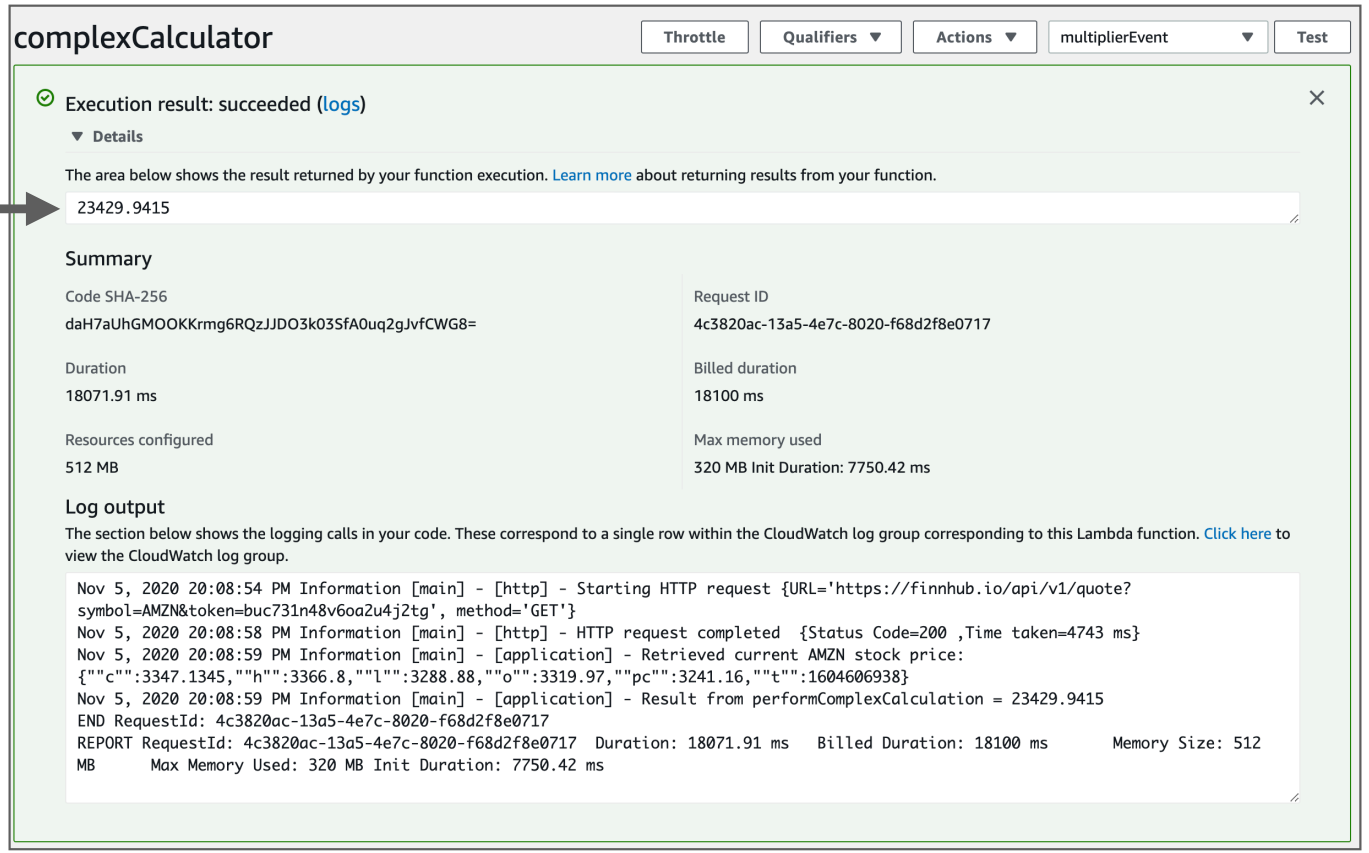
All arguments required by your function need to be in the test event.



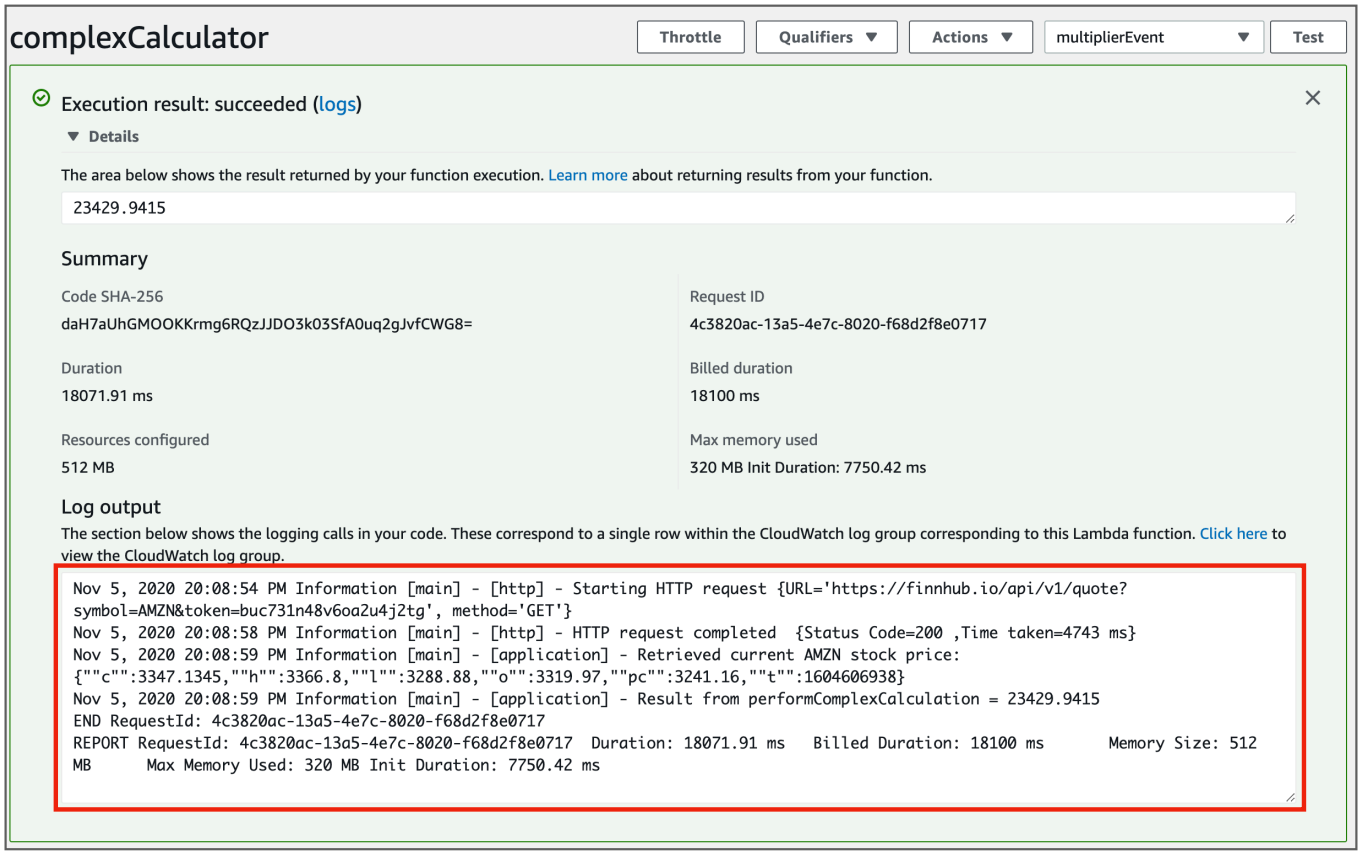
Run Your cflambda Function!



See the Results



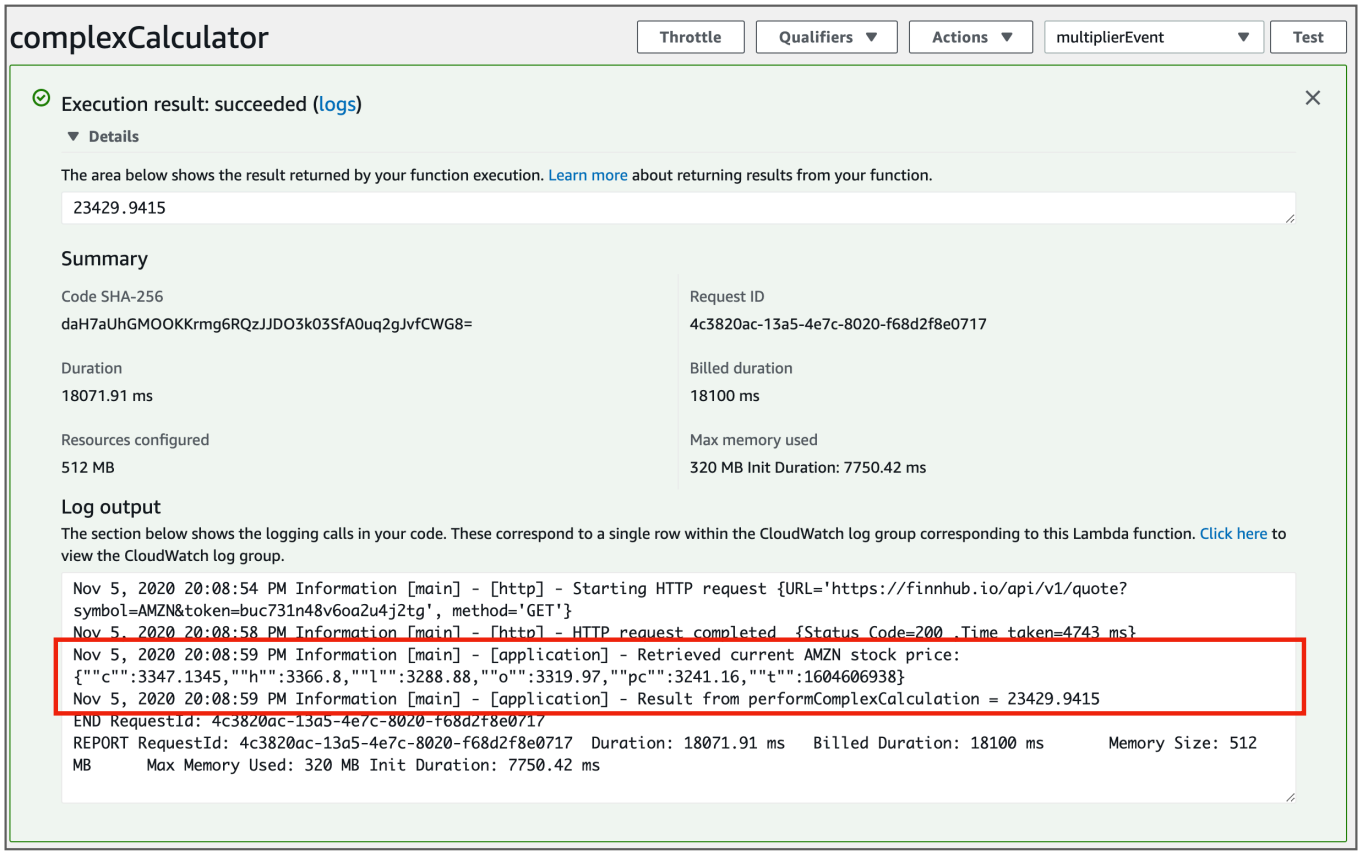
Where is the Log Info Coming From?



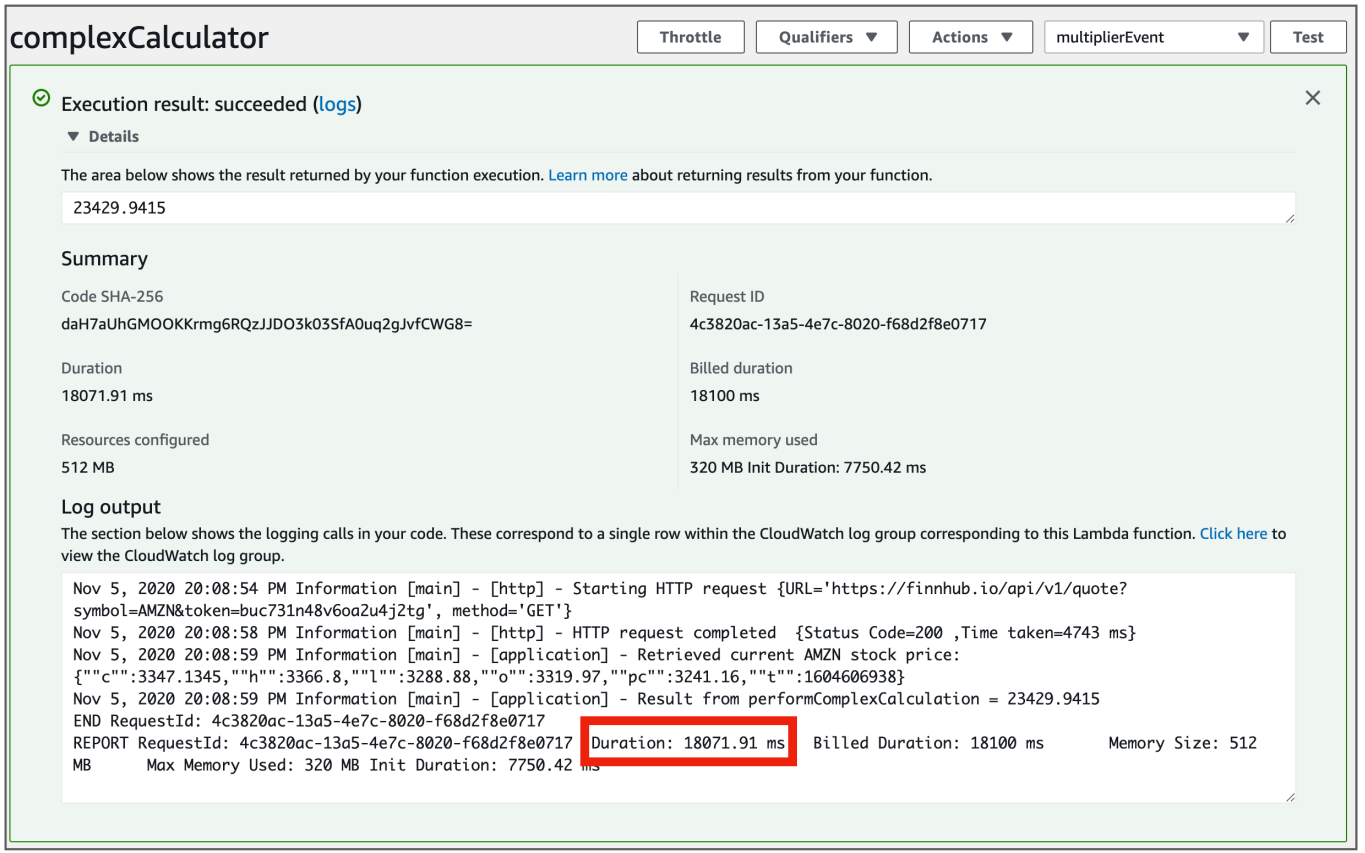
Add to CloudWatch Logs with the APPLICATION_LOG

```
var cflogObject = CreateObject("java","coldfusion.log.CFLogs");
var logger = cflogObject.APPLICATION_LOG;

logger.info("Starting Request to performComplexCalculation: " & context.getAwsRequestId());

logger.info("Result from performComplexCalculation = " & returnValue);
```

Your Log Entries in the CloudWatch Logs



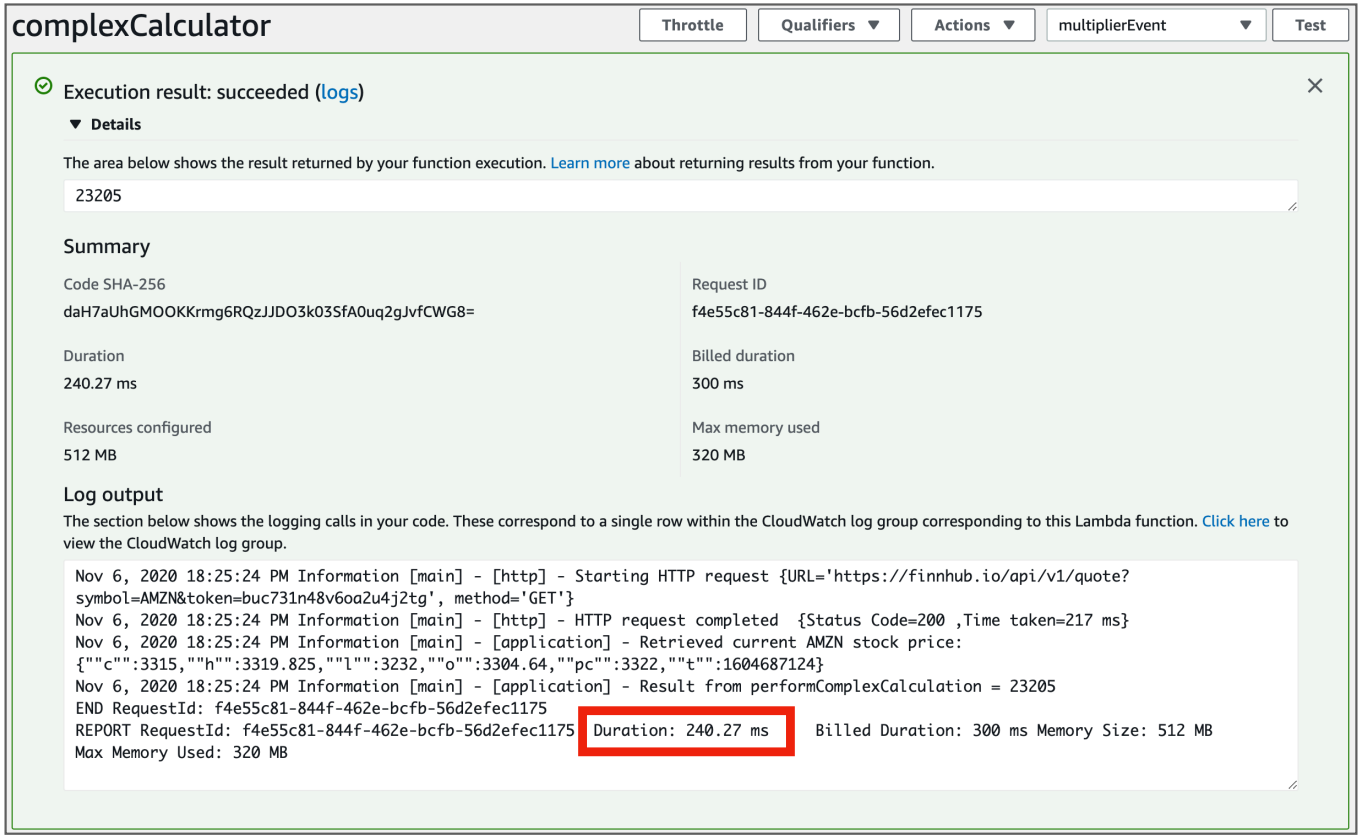
What's With That Super-Long Duration?



Why So Long, Lambda?

1. Fire up the container
2. Download your source ZIP file
3. Unpack your source ZIP file
4. Start Java
5. Start the ColdFusion server
6. Execute your code

Subsequent Invocations: Fast



COLD STARTS ARE STILL A PROBLEM IN JAVA

Avoiding Long Cold Starts

1. Use provisioned concurrency
2. Reduce the size of the deployment bundle
3. Use CloudWatch triggers
4. Don’t worry – it’s async!

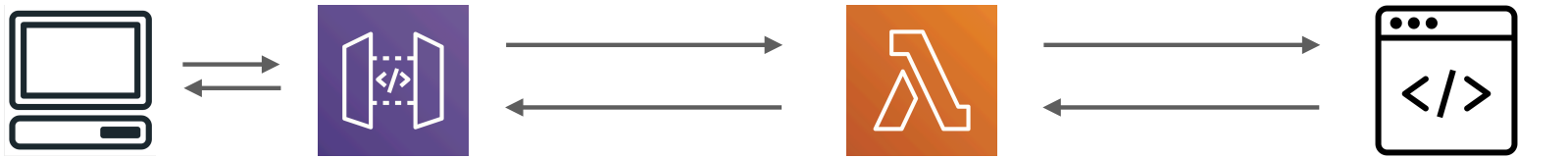
How Would You Invoke This from CF?

```
lambda = awsServiceFactory.createServiceObject('lambda');

invokeRequest = CreateObject('java', 'com.amazonaws.services.lambda.model.InvokeRequest').init();

invokeRequest.setFunctionName(lambdaFunctionARN);

invokeRequest.setPayload(jsonPayload);

result = variables.lambda.invoke(invokeRequest);
```

Full example in my AWS Playbox repo:
https://github.com/brianklaas/awsPlaybox

Async Invocation

What about synchronous invocation from a Web app?



Paul Kukiel's sample API Gateway Lambda repo:
https://github.com/kukielp/acf-lambda

AWS Serverless Application Model (SAM)

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Globals:
 Function:
  Timeout: 60
Resources:
 AWSServerless:
  Type: AWS::Serverless::Function
  Properties:
   CodeUri: ./cfusion/bin/dist/cf-aws-serverless.zip
   Handler: coldfusion.serverless.handlers.cf.CFLambdaRequestStreamHandler::handleEventRequest
   Runtime: java11
   MemorySize: 1024
```

Use AWS SAM to Manage Your cflambda Deployment

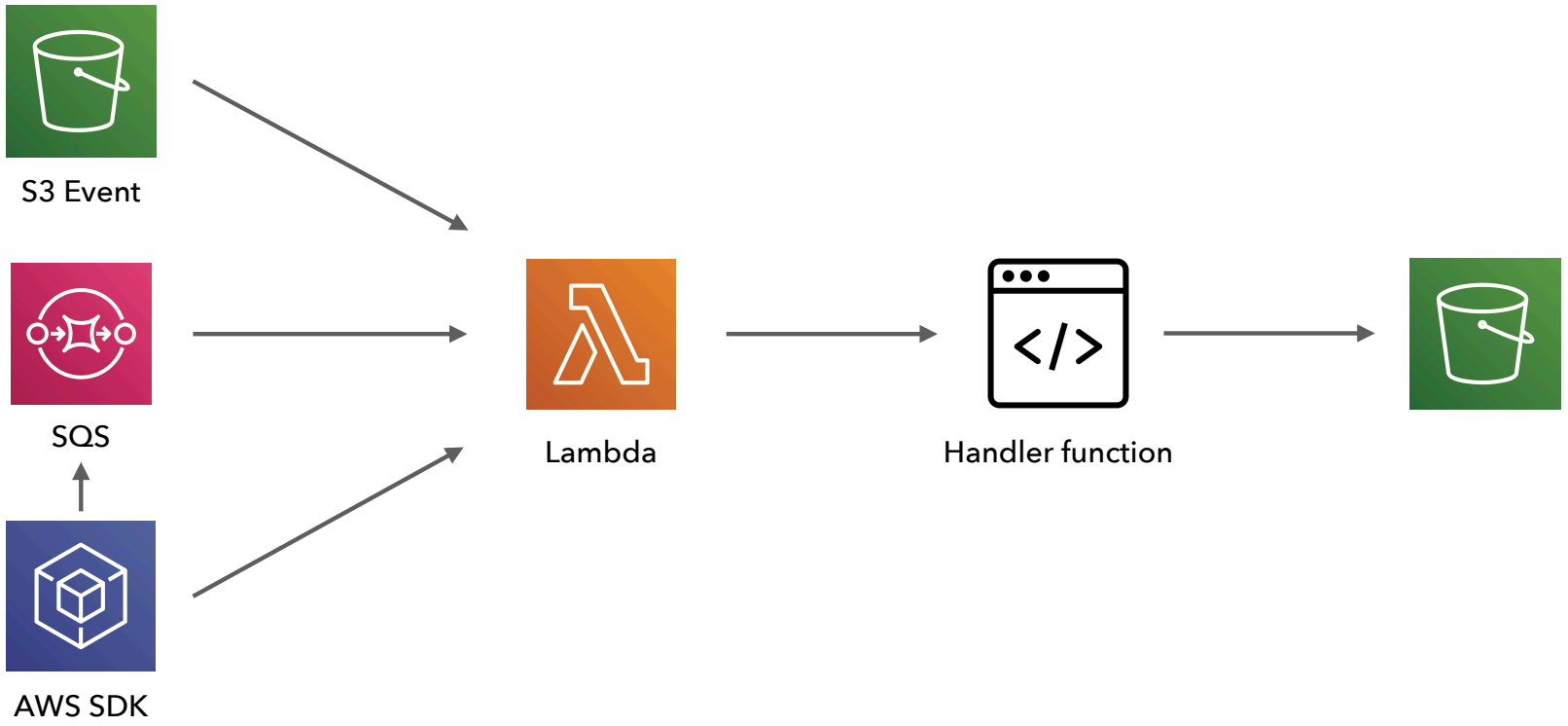
Built and supported by AWS

Infrastructure as code — no fiddling in the console

Handles upload and runtime config for you in a single command

Can run Lambda functions locally for testing!

The Real Power: Serverless Applications



The Canonical Image Resizer CFC

```
component output="false" hint="A utility for resizing images" {

  public string function resizeImage(any event, any context) {

        init();
        variables.logger.info("Executing main body of resizeImage - RequestID: " & context.getAwsRequestId());

        // Read event values
        if (structKeyExists(event, "urlOfImageOnS3")){
            variables.logger.info("This request came in via a direct invocation");
            var pathToImageOnS3 = event.urlOfImageOnS3;

        } else {
            variables.logger.info("This request came in via SQS");
            variables.logger.info("SQS message body:" & event.records[1].body);
            var sqsBodyAsJSON = deserializeJSON(event.records[1].body);
            var pathToImageOnS3 = sqsBodyAsJSON.urlOfImageOnS3;
        }
```

The Canonical Image Resizer CFC

```
var origFileExtension = ListLast(pathToImageOnS3, ".");
var origFileName = ListLast(Left(pathToImageOnS3, (Len(pathToImageOnS3) - Len(origFileExtension) - 1)), "/");
var pathToFileOnS3 = Left(pathToImageOnS3, (Len(pathToImageOnS3) - Len(origFileName) - Len(origFileExtension) - 2));
var fileNameToWrite = "tempImage." & origFileExtension;

var sourceFileInTmp = "/tmp/" & fileNameToWrite;

variables.s3Service = cloudService(this.s3Cred,this.s3Conf);
variables.bucketObj = s3Service.bucket(“nameOfMyBucket");
```

Lambda has a read-only file system!

500MB of temp storage in /tmp

The Canonical Image Resizer CFC

```
var origFileExtension = ListLast(pathToImageOnS3, ".");
var origFileName = ListLast(Left(pathToImageOnS3, (Len(pathToImageOnS3) - Len(origFileExtension) - 1)), "/");
var pathToFileOnS3 = Left(pathToImageOnS3, (Len(pathToImageOnS3) - Len(origFileName) - Len(origFileExtension) - 2));
var fileNameToWrite = "tempImage." & origFileExtension;

var sourceFileInTmp = "/tmp/" & fileNameToWrite;

variables.s3Service = cloudService(this.s3Cred,this.s3Conf);
variables.bucketObj = s3Service.bucket(“nameOfMyBucket");
```

Yes, you can use CF2021's cloudService() functionality in cflambda!

The Canonical Image Resizer CFC

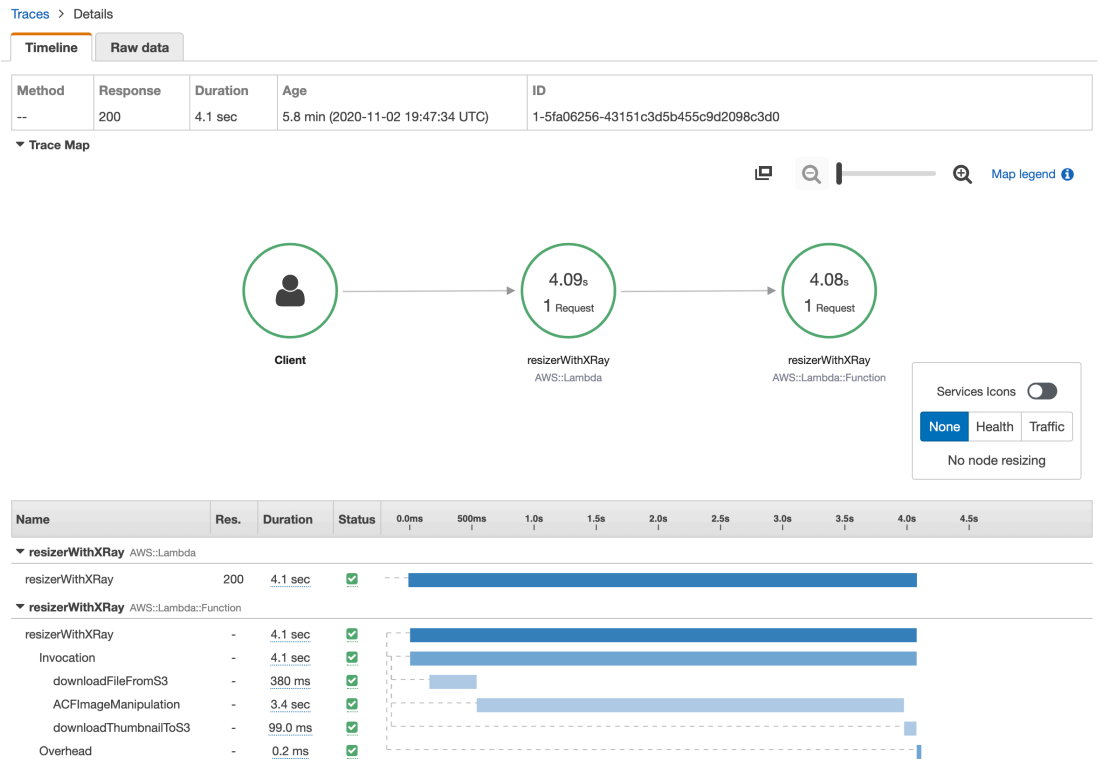
```
beginAWSXraySubsegment("downloadFileFromS3");
var downloadRequest =  {
        "destinationFile": sourceFileInTmp,
        "key": pathToImageOnS3
};
variables.logger.info("Getting file: " & pathToImageOnS3);
try {
        var downloadResponse = variables.bucketObj.downloadToFile(downloadRequest);
} catch (any e) {
        variables.logger.info("Error while getting file from S3:" & serializeJSON(e));
        return false;
}
endAWSXraySubsegment();
```

The Canonical Image Resizer CFC

```
beginAWSXraySubsegment("downloadFileFromS3");
var downloadRequest =  {
        "destinationFile": sourceFileInTmp,
        "key": pathToImageOnS3
};
variables.logger.info("Getting file: " & pathToImageOnS3);
try {
        var downloadResponse = variables.bucketObj.downloadToFile(downloadRequest);
} catch (any e) {
        variables.logger.info("Error while getting file from S3:" & serializeJSON(e));
        return false;
}
endAWSXraySubsegment();
```

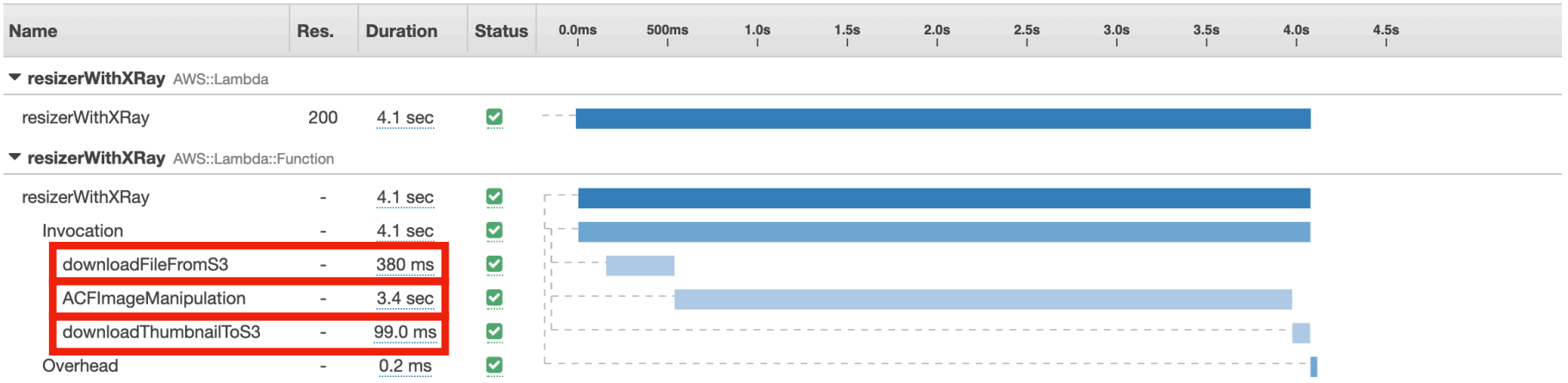
What's this X-Ray stuff?

X-Ray: Distributed Tracing and Timing



X-Ray shows you request times across AWS services or in your own code.

X-Ray: Distributed Tracing and Timing



The Canonical Image Resizer CFC

```
beginAWSXraySubsegment("downloadFileFromS3");
var downloadRequest =  {
        "destinationFile": sourceFileInTmp,
        "key": pathToImageOnS3
};
variables.logger.info("Getting file: " & pathToImageOnS3);
try {
        var downloadResponse = variables.bucketObj.downloadToFile(downloadRequest);
} catch (any e) {
        variables.logger.info("Error while getting file from S3:" & serializeJSON(e));
        return false;
}
endAWSXraySubsegment();
```

You can make as many X-Ray subsegments as you want.

The Canonical Image Resizer CFC

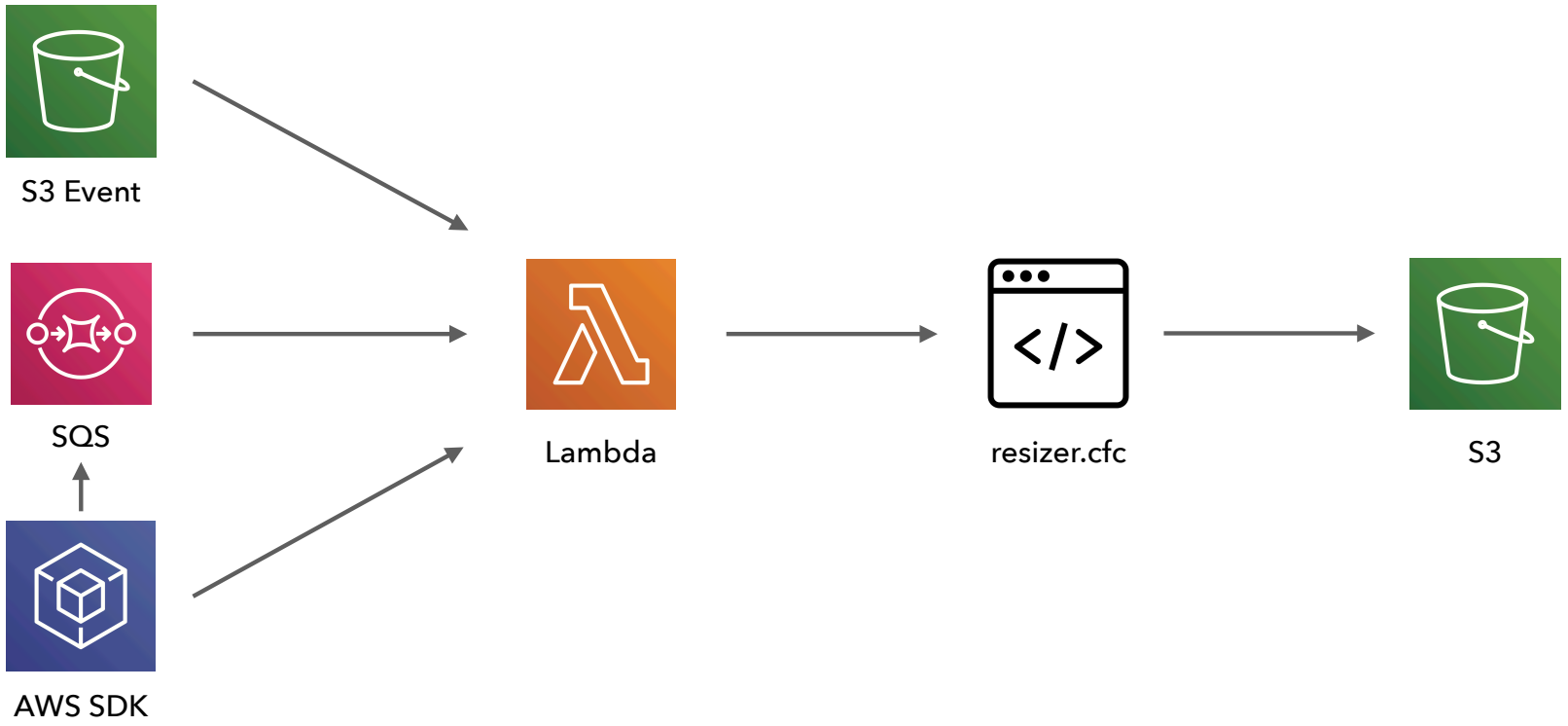
```
beginAWSXraySubsegment("ACFImageManipulation");
var sourceImage = imageRead(sourceFileInTmp);
imageResize(sourceImage, "25%", "");
var resizedImageName = origFileName & "-thumb." & origFileExtension;
var resizedImageInTmp = "/tmp/" & resizedImageName;
imageWrite(sourceImage, resizedImageInTmp);
endAWSXraySubsegment();
```

cflambda can use almost any ColdFusion language feature/function

The Canonical Image Resizer CFC

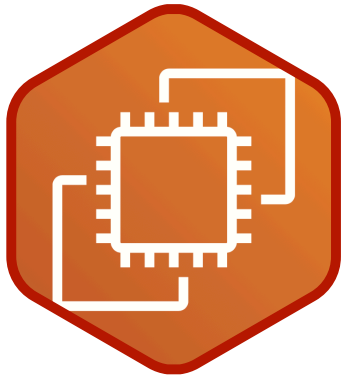
```
beginAWSXraySubsegment("uploadThumbnailToS3");
var destinationPathOnS3 = "";
	if (len(pathToFileOnS3)) {
		destinationPathOnS3 &= pathToFileOnS3 & "/";
	}
destinationPathOnS3 &= resizedImageName;
var uploadRequest = {
	"srcFile" : resizedImageInTmp,
	"key" : destinationPathOnS3
};
variables.logger.info("Writing thumbnail file: " & destinationPathOnS3);
try {
	var uploadResponse = variables.bucketObj.uploadFile(uploadRequest);
} catch (any e) {
	variables.logger.info("Error while writing resized file to S3:" & serializeJSON(e));
	return false;
}
endAWSXraySubsegment();
```

Invoke from S3, SQS or the Java SDK



Microservices without the overhead of:



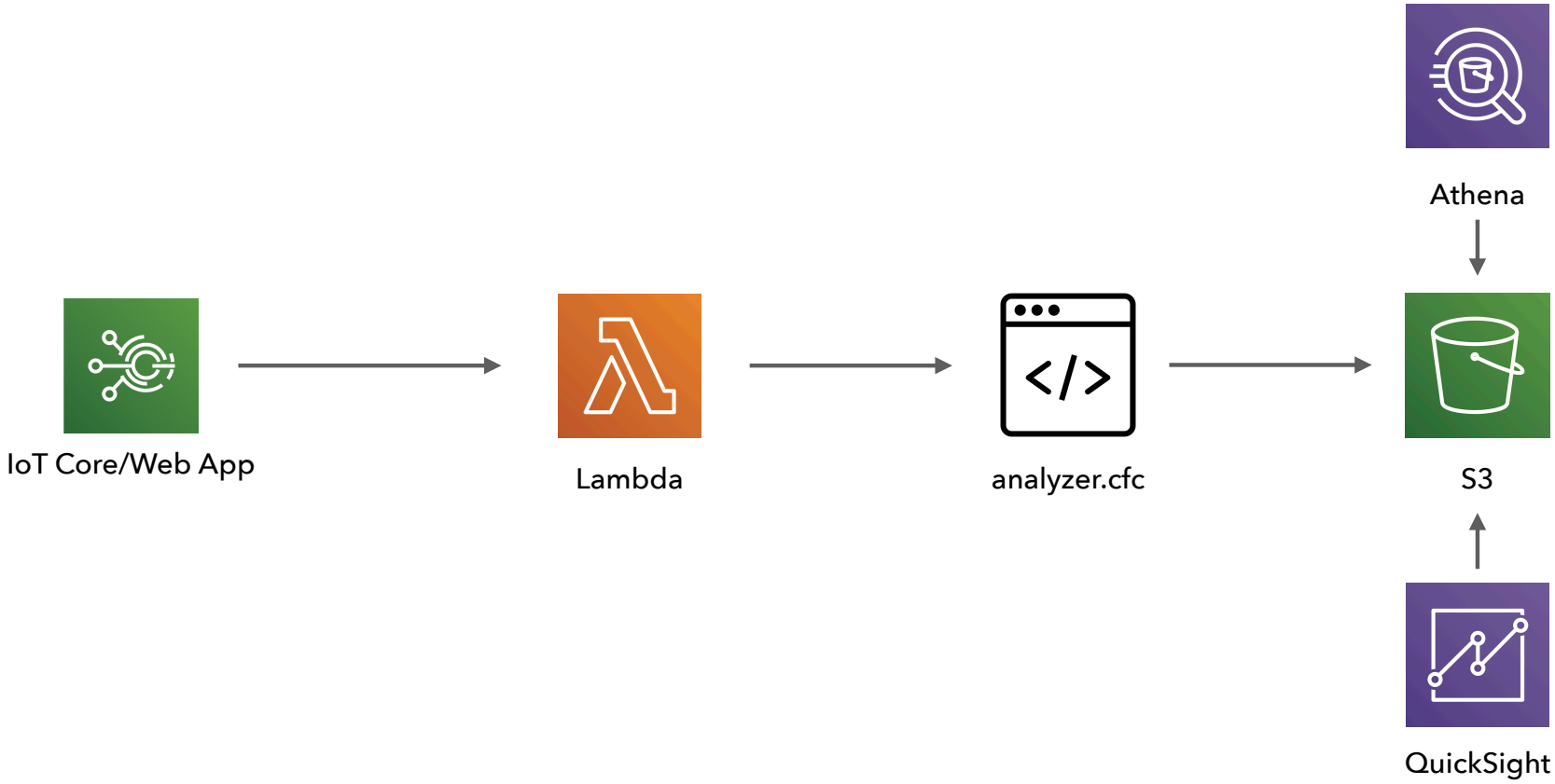



Write Aurora User–Defined Functions in ColdFusion

Capture Form Data and Write to Goole Sheets in ColdFusion

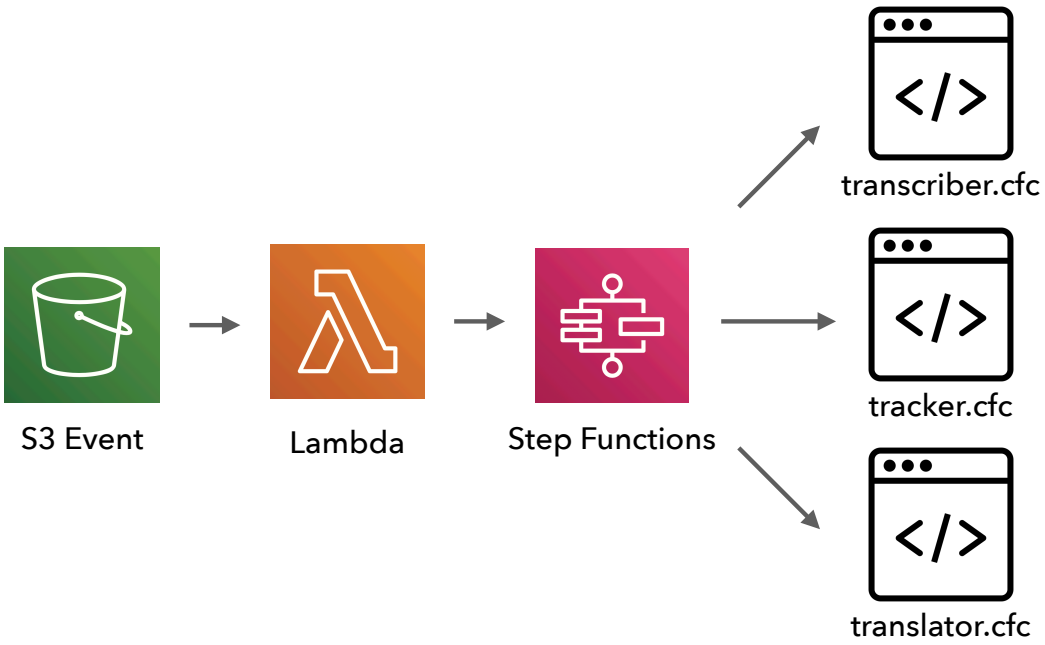
* without running a server!

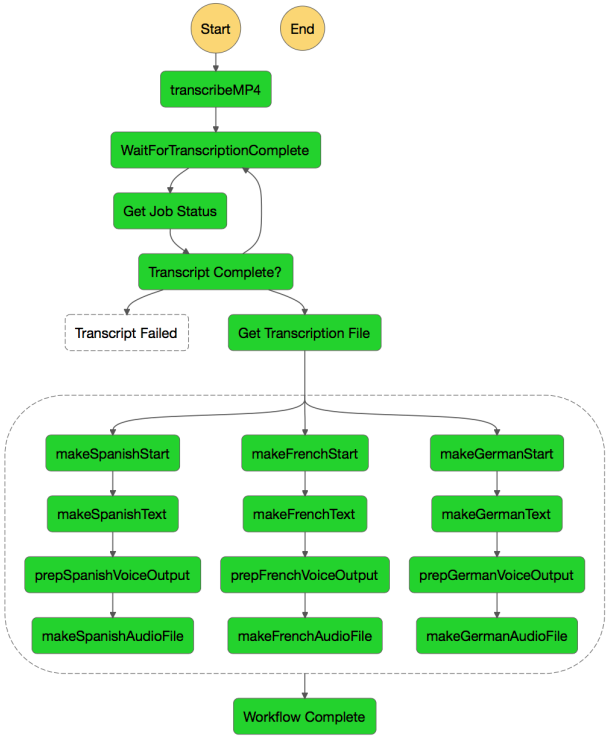
Analyze Clickstream Data in ColdFusion



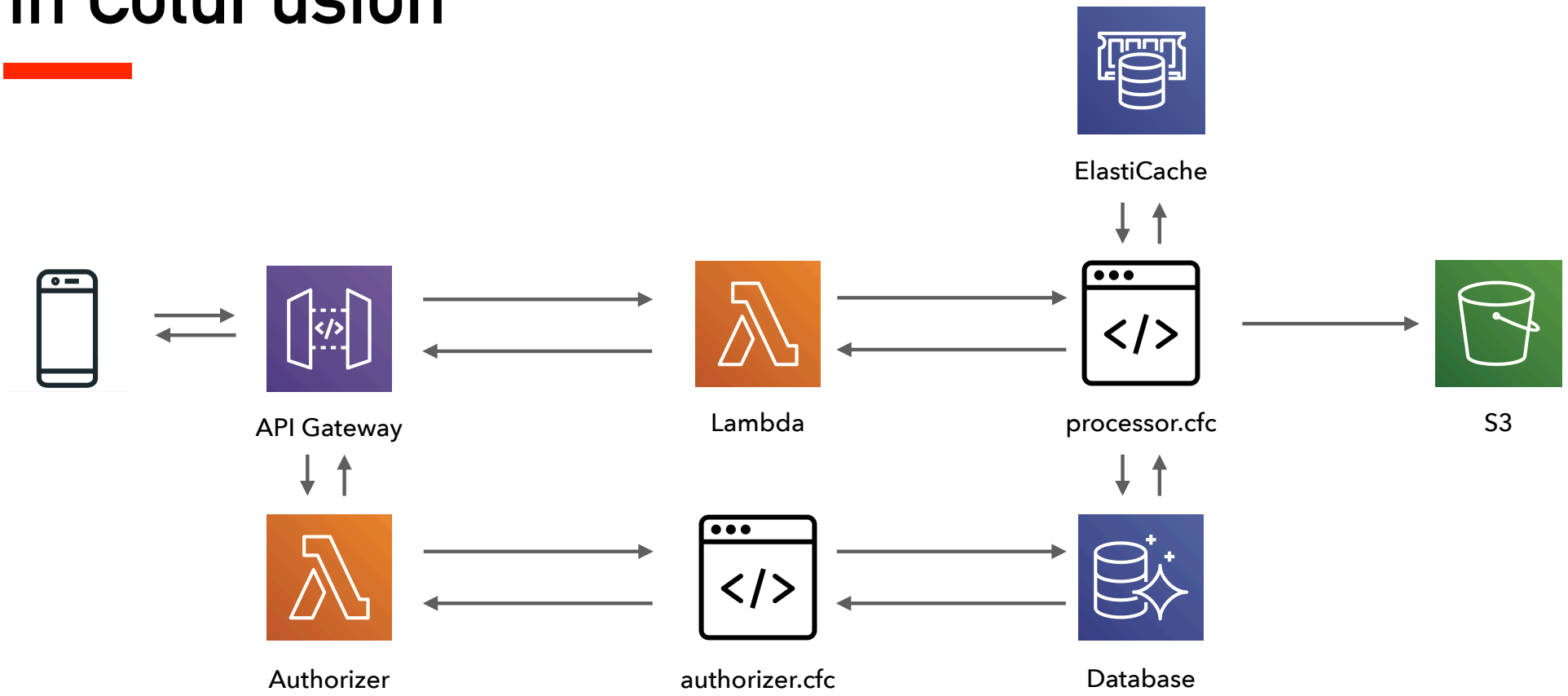
Perform Audits of Your Entire AWS Infrastructure in ColdFusion

Run ColdFusion–Based Lambdas in Your Step Functions Workflows





Build a Full-Stack Serverless Web App in ColdFusion




Go Do!

Resources from this presentation:
brianklaas.net

AWS Playbox:
github.com/brianklaas/awsPlaybox

brian.klaas@gmail.com
@brian_klaas