



## AWS Machine Learning Blog

# Analyze Emotion in Video Frame Samples Using Amazon Rekognition on AWS

by Cyrus Wong | on 04 AUG 2017 | in [Amazon Rekognition](#) | [Permalink](#) | [Comments](#) | [Share](#)

*This guest post is by AWS Community Hero [Cyrus Wong](#). Cyrus is a Data Scientist at the Hong Kong Vocational Education ([Lee Wai Lee](#)) Cloud Innovation Centre. He has achieved all 7 AWS Certifications and enjoys sharing his AWS knowledge with others through open-source projects, blog posts, and events.*

HowWhoFeelInVideo is an application that analyzes faces detected in sampled video clips to interpret the emotion or mood of the subjects. It identifies faces, analyzes the emotions displayed on those faces, generates corresponding Emoji overlays on the video, and logs emotion data. The application accomplishes all of this within a serverless architecture using Amazon Rekognition, AWS Lambda, AWS Step Functions, and other AWS services.

HowWhoFeelInVideo was developed as part of a research project at the Hong Kong Vocational Education ([Lee Wai Lee](#)) Cloud Innovation Centre. The project is focused on childcare, elder care, and community services. However, emotion analysis can be used in many areas, including rehabilitative care, nursing care, and applied psychology. My initial focus has been on applying this technology to the classroom.

In this post, I explain how HowWhoFeelInVideo works and how to deploy and use it.

## How it works

Teachers, such as myself, can use HowWhoFeelInVideo to get an overall measure of a student's mood (e.g., happy, or calm, or confused) while taking attendance. The instructor can use this data to adjust his or her focus and approach to enhance the teaching experience. This research project is just beginning. I will update this post after I receive additional results.

To use HowWhoFeelInVideo, a teacher sets up a basic classroom camera to take each students' attendance using face identification. The camera also captures how students feel during class. Teachers can also use HowWhoFeelInVideo to prevent students from falsely reporting attendance.

## Architecture and design

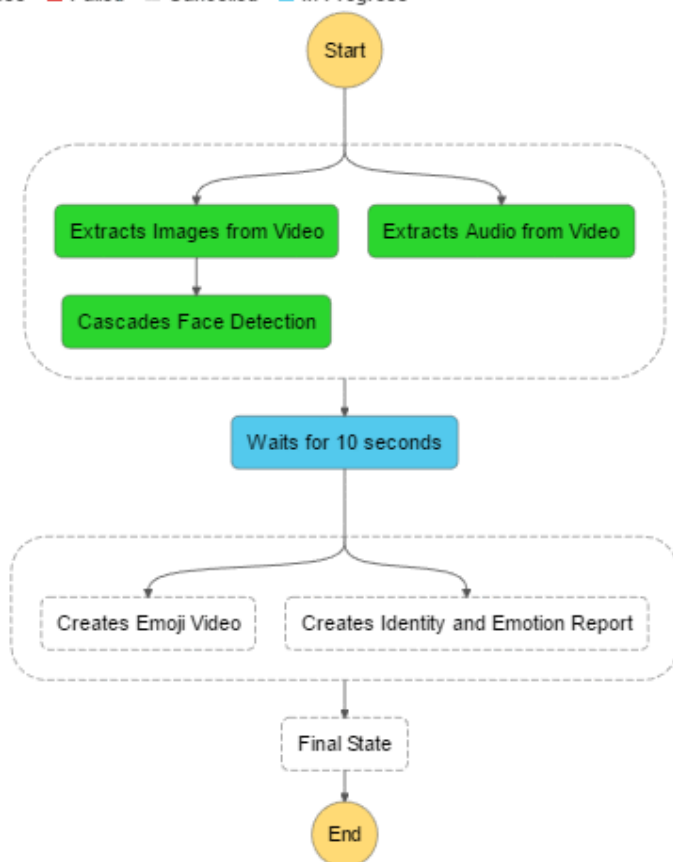
HowWhoFeelInVideo is a serverless application built using AWS Lambda functions. Five of the Lambda functions are included in the HowWhoFeelInVideo state machine. [AWS Step Functions](#) streamlines

coordinating the components of distributed applications and microservices using visual workflows. This simplifies building and running multi-step applications.

The HowWhoFeelInVideo state machine starts with the `startFaceDetectionWorkFlowLambda` function, which is triggered by an Amazon S3 PUT object event. `startFaceDetectionWorkFlowLambda` passes in the following information into the execution:

```
{
  "bucket": "howwhofeelinvideo",
  "key": "Test2.mp4"
}
```

■ Success ■ Failed ■ Cancelled ■ In Progress



With Step Functions, you can use meaningful names, making it easy to understand workflows. They also let you monitor the processing pipeline from the AWS console.

The HowWhoFeelInVideo state machine is available in the us-east-1 AWS Region. The video processing tasks are implemented using [FFmpeg](#).

## Behind the scenes

Before you start using HowWhoFeelInVideo, you need to understand how it works and a few general principles.

When you need to make use of other pre-built programs such as FFmpeg, you can run another program or start a new process in Lambda:

1. Copy the program for a new process in Lambda to the `/tmp` directory.
2. Call the shell and use `chmod` to give it execution
3. Call the shell to run the program.

Lambda saves files for data processing in the `/tmp` directory, which is limited to 500 MB. To improve Lambda's performance, the container is reused. This means that files in the `/tmp` directory might retain and use additional space during the next Lambda call. Therefore, you should always remove old files from `/tmp`, either at the beginning or the end of each step.

Face analysis is triggered by the `ProcessImage` Lambda function in Scala. The `ProcessImage` function processes only one image at a time. It performs the following tasks:

1. Downloads an image from an S3 bucket
2. Calls Amazon Rekognition to detect faces and emotion (with the `detectFaces` operation)
3. Crops the face from the image using the bounding box provided by the `detectFaces` operation
4. Attempts to identify the owner by searching each face (using the `searchFacesByImage` operation) in the specified face collection
5. Joins the result of emotion and face identification
6. Creates an Emoji Face overlap image and emotion report records
7. Uploads the Emoji Face overlap image and emotion report records to an S3 bucket

Because AWS Lambda charges for memory usage per 100 ms and Amazon Rekognition charges by the number of requests, the system is designed to run at maximum concurrency. I pay the same price whether I process all screen capture images at once or one by one!

The Cascades Face Detection step asynchronously invokes the `ProcessImage` Lambda function for each screen capture image nearly in parallel. Each `ProcessImage` function calls Amazon Rekognition for each face detected.

The following is a parallel map function which invokes the `ProcessImage` function for each image frame.

```
let invokeLambda = (key) => new Promise((resolve, reject) => {
  let data = JSON.stringify({bucket: bucket, key: prefix + "/" + key});
  let params = {
    FunctionName: process.env['ProcessImage'], /* required */
    Payload: data /* required */
  };
  lambda.invoke(params, (err, data) => {
    if (err) reject(err, err.stack); // an error occurred
    else resolve(data); // successful response
  });
});

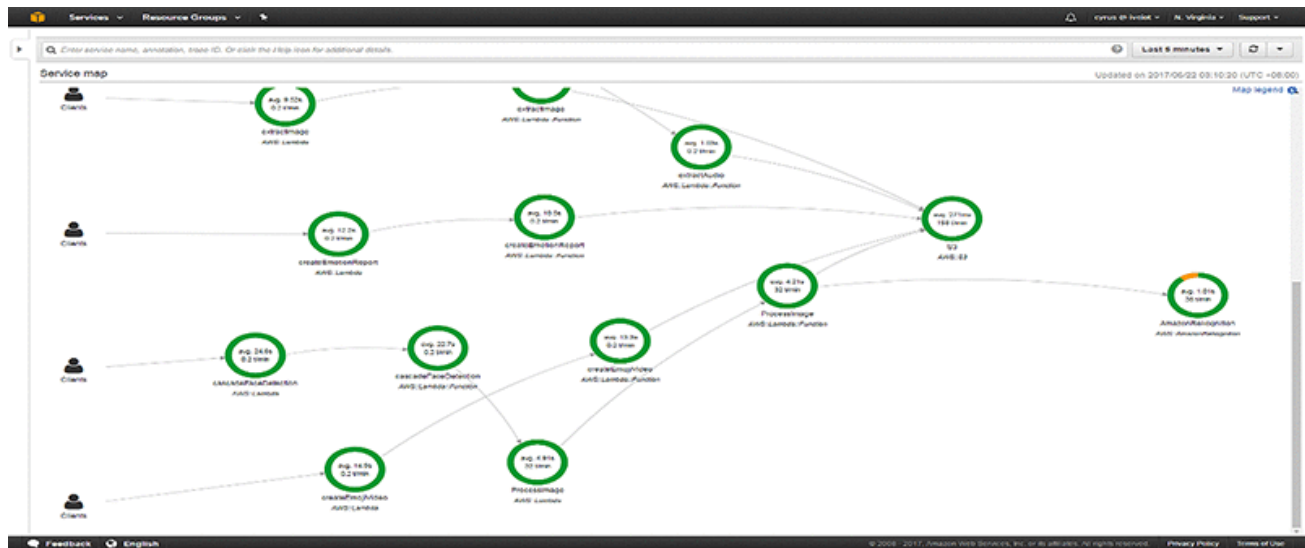
let invokeLambdaPromises = keys.map(invokeLambda);
Promise.all(invokeLambdaPromises).then(() => {
  let pngKey = keys.map(key => key.split(".")[0] + ".png");
  let data = {bucket: bucket, prefix: prefix, keys: pngKey};
```

```
console.log("involveLambdaPromises complete!");
callback(null, data);
}
```

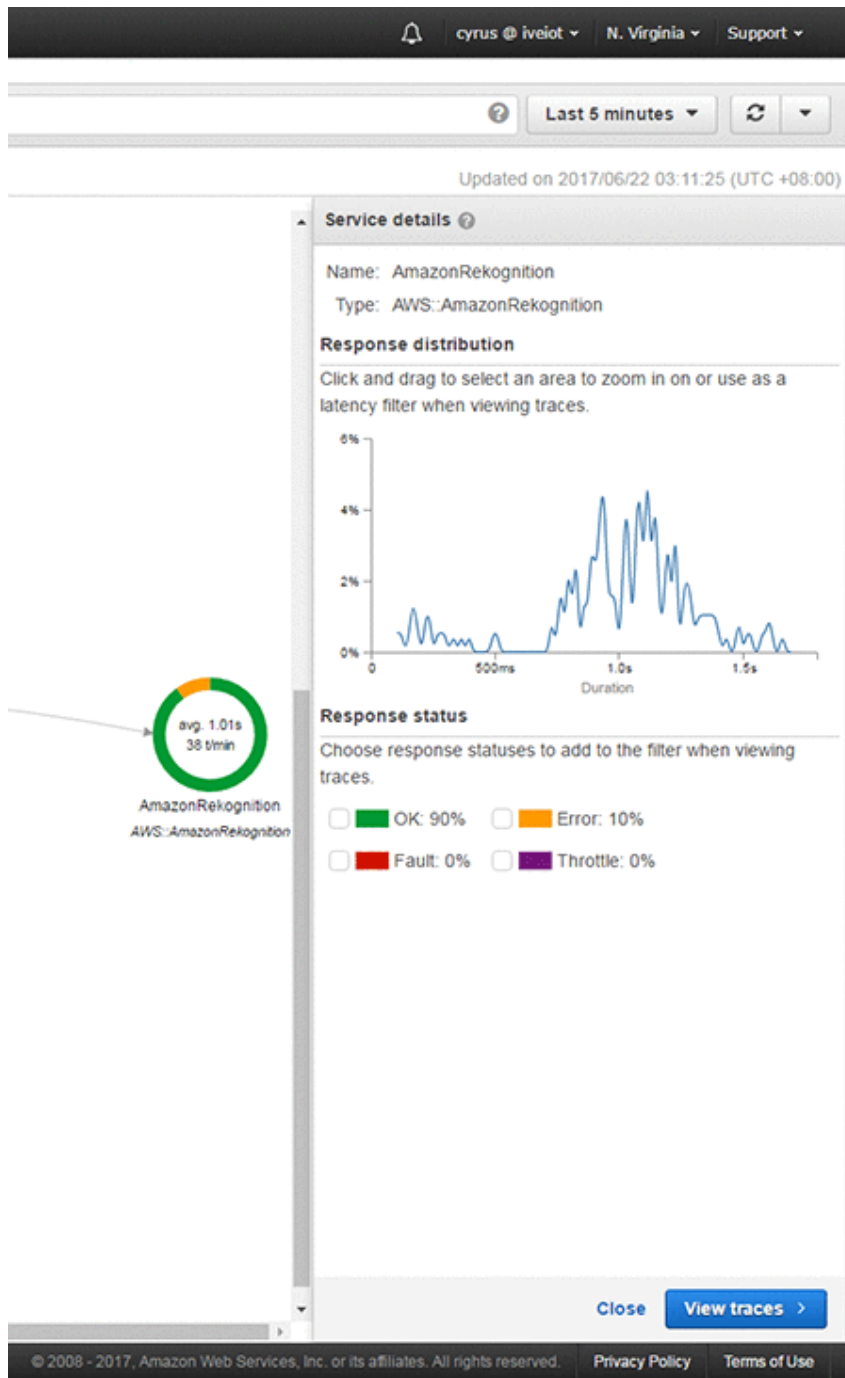
The following is a parallel map function that gets the face ID in Scala:

```
//Parallel the search request.
val faceMatchAndBoundingBoxAndEmotion = faceImagesAndBoundingBoxAndEmotion.par.map(f =>
  searchFacesByImage(f._1) match {
    case Some(face) => {
      val id = face.getFaceMatches.asScala.headOption match {
        case Some(a) => a.getFace.getExternalImageId
        case None => "?????"
      }
      (id, f._2, f._3)
    }
    case None => ("?????", f._2, f._3)
  }
})
faceMatchAndBoundingBoxAndEmotion.seq
```

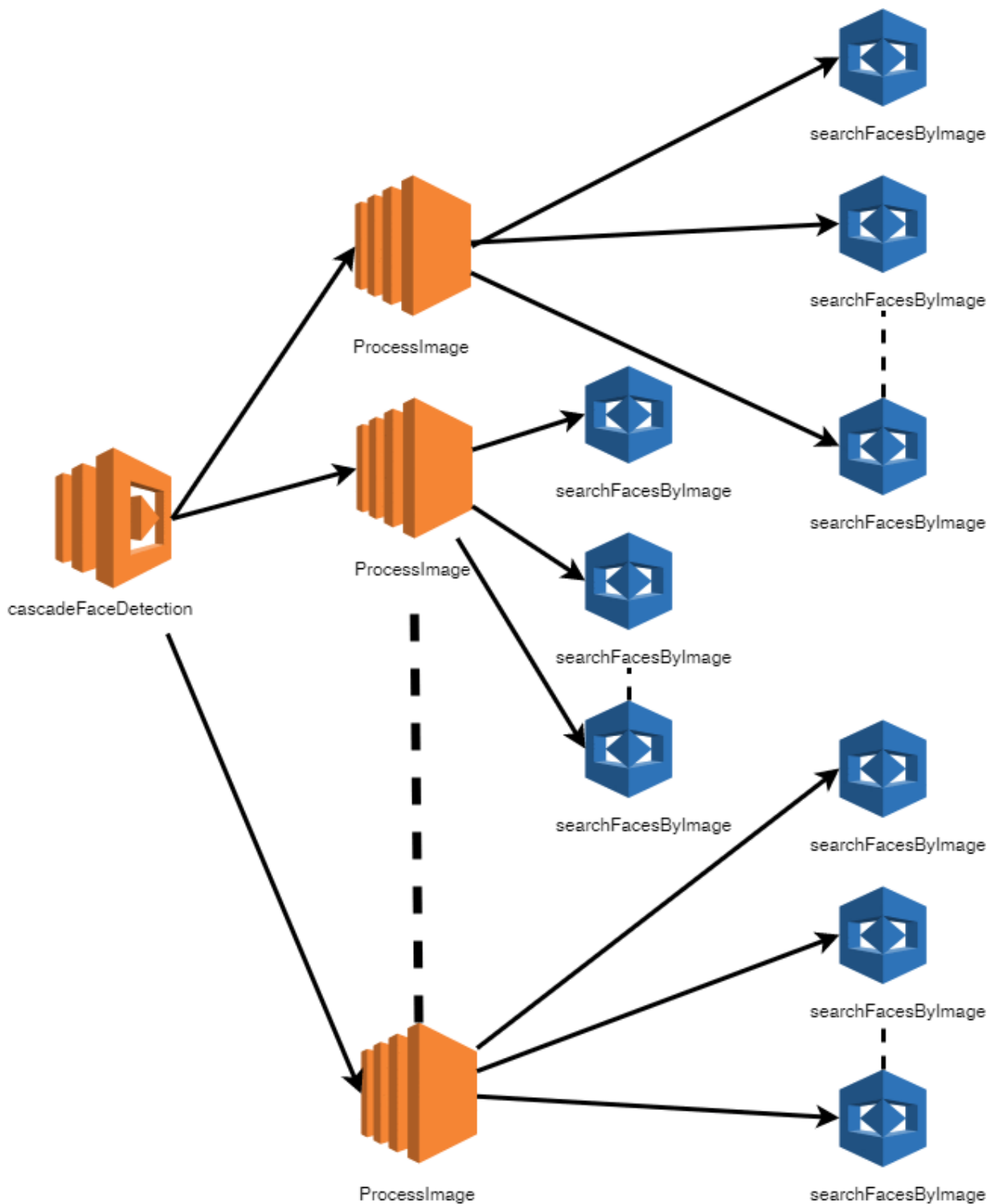
The following service map shows the dependency trees with trace data that I can use to drill into specific services or issues. This provides a view of connections between services in your application and aggregated data for each service, including average latency and failure rates.



The following is a latency distribution histogram for an Amazon Rekognition API call:

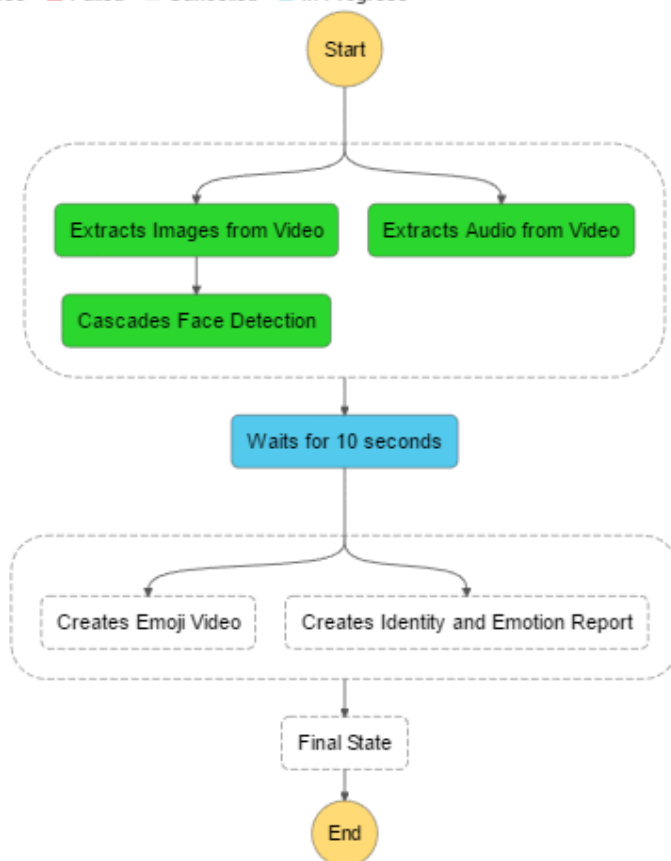


Latency is the amount of time between the start of a request and when it completes. A histogram shows a distribution of latencies. This latency distribution histogram shows duration on the x-axis, and the percentage of requests that match each duration on the y-axis.

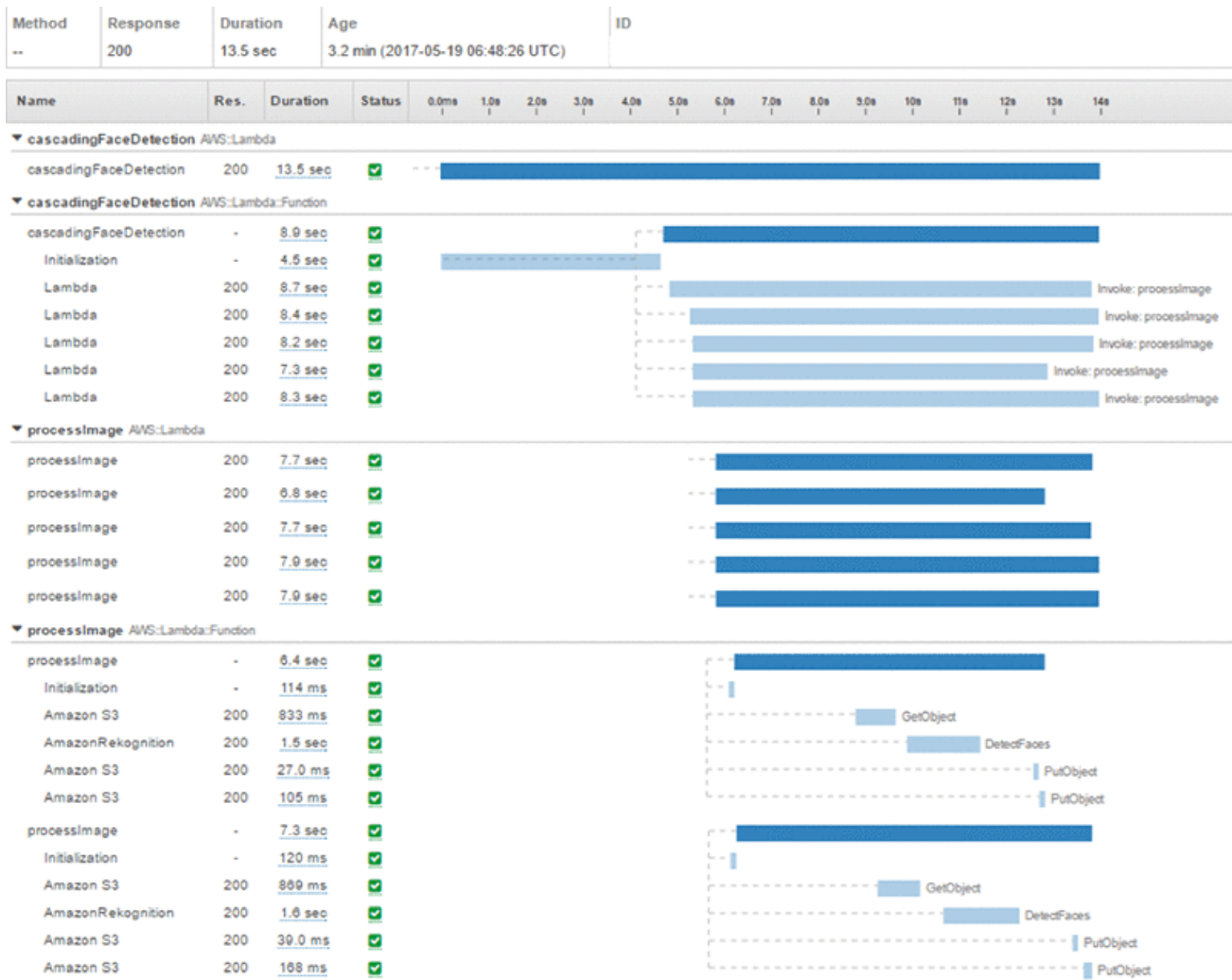


I set the maximum execution of the `ProcessImage` function to 1.5 minutes and added a 10-second wait step in the state machine to ensure that all images and emotion records are ready in Amazon S3.

■ Success ■ Failed ■ Cancelled ■ In Progress



The following Lambda cascading timeline shows how processing operates in a highly parallel manner:



## The result

The result includes a single output image:





### An output record from single image

```
[{"seq": "test5/0036", "id": "????", "happy": 11.956384658813477, "sad": 0.0, "angry": 0.0, "confused": 0.0, "disgusted": 0.0, "surprised": 0.0, "calm": 0.0, "unknown": 0.0}]
```

### An output report for all images in .csv format:

seq	id	happy	sad	angry	confused	disgusted	surprised	calm	unknown
test5/0006	????	98.02718	0.631481	0	0	0	11.18102	0	0
test5/0008	????	50.3309	6.570876	0	0	0	0	43.05873	0
test5/0009	????	24.73417	15.86392	0	0	0	0	76.01309	0
test5/0010	????	21.94545	4.583901	0	0	0	0	73.68261	0
test5/0011	????	51.10088	16.62526	0	0	0	0	38.25601	0
test5/0012	????	60.17631	5.188495	0	0	0	0	55.99194	0
test5/0013	????	3.750361	2.421415	0	0	0	90.49848	0	0
test5/0013	cyruswong	65.18288	3.520817	0	0	0	0	17.97778	0

#### Note:

I don't index images of my students' faces within the video. Instead, they are each allocated an unknown facelid.

With this report, you can easily aggregate data on overall student satisfaction and determine how each individual feels throughout the course or the event. For health research, we plan to objectively record emotional feedback during class for Special Education Needs (SEN) students when we use a different teaching method. For a class of non-SEN students, we import a CSV report into a database with the following simple SQL statement:

```
SELECT Report.id AS Student, Count(Report.seq) AS Attended, Sum(Report.happy) AS Happy, Sum(Report.sad) AS Sad, Sum(Report.angry) AS Angry, Sum(Report.confused) AS Confused, Sum(Report.disgusted) AS Disgusted, Sum(Report.surprised) AS Surprised, Sum(Report.calm) AS Calm, Sum(Report.unknown) AS Unknown
FROM Report GROUP BY Report.id;
```

Class Summary									
Student	Attended	SumOfhappy	SumOfsad	SumOfangry	SumOfconfused	SumOfdisgusted	SumOfsurprised	SumOfcalm	SumOfunknown
????	81	3545.032018	697.9098778	100.3656735	411.0281124	22	214.346653	1549.940836	0
cyruswong	14	964.4436002	26.60443783	7.937097549	95.77599835	0	4.30300808	306.4037442	0

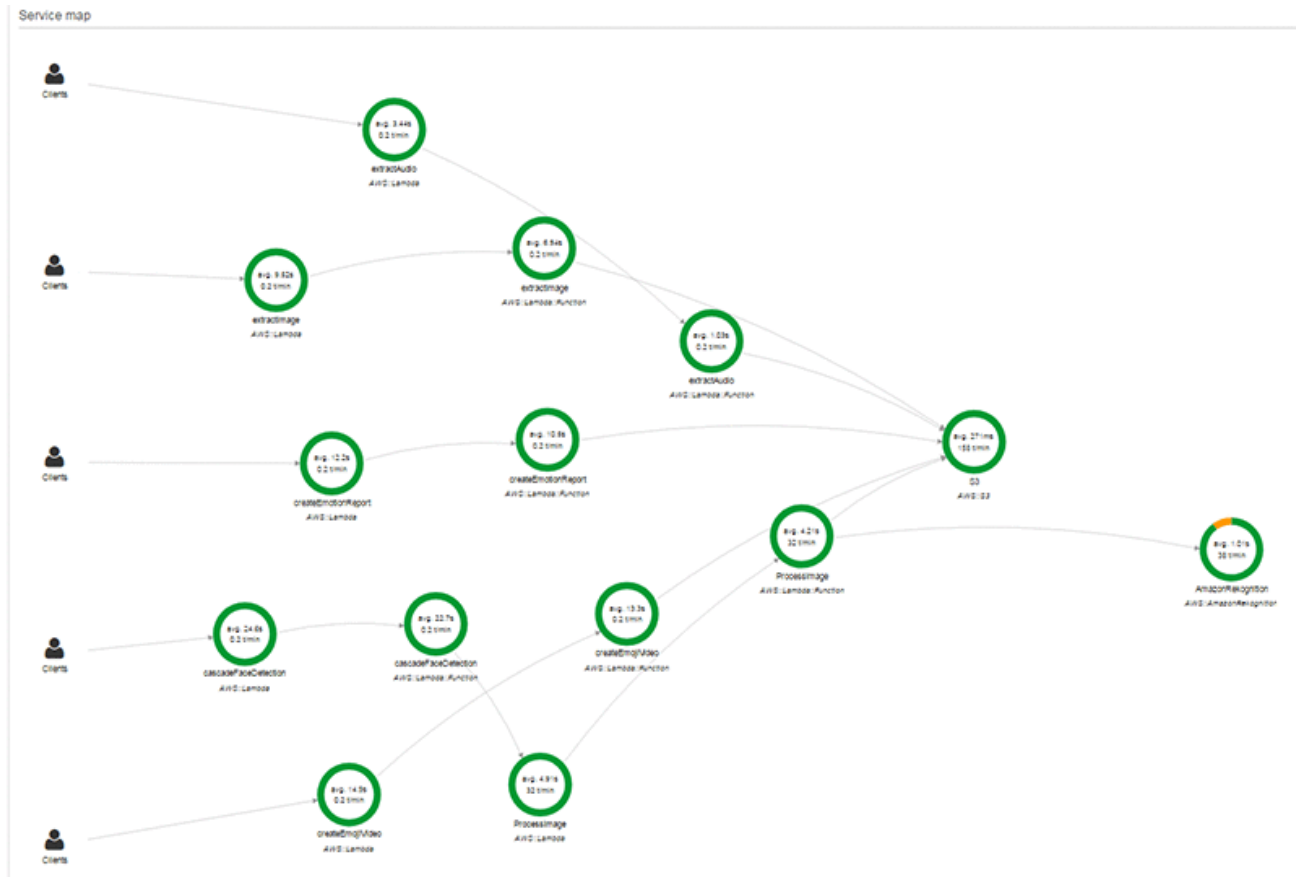
## Demo Video Output

The following video is my TV interview with four of my students about Hong Kong Open Data and it has been processed using HowWhoFeelInVideo.



The step that extracts the video to images must complete within the maximum Lambda execution time of 5 minutes, so you cannot directly process long-running video. However, it is easy to [create fragmented MP4 files with Amazon Elastic Transcoder](#) and process the analysis over the MP4 fragments.

## Overall AWS X-Ray Service Map



Source code for HowWhoFeelInVideo is available in [GitHub](#).

## Deploying HowWhoFeelInVideo

Deployment is very simple. I created an AWS CloudFormation template with AWS Serverless Application Model (AWS SAM). AWS SAM is a specification for describing Lambda-based applications. It offers a syntax designed specifically for expressing serverless resources. To deploy the application, perform the following steps:

1. In Amazon Rekognition, [create a face collection](#) named **student**.
2. Use the AWS CLI to [store faces](#).
3. Create an S3 source bucket in the us-east-1 AWS Region.
4. Download the three files in the [Deployment folder](#) on GitHub.
5. Upload two source packages that you get from the Deployment folder, `FaceAnalysis-assembly-1.0.jar` and `ProcessVideoLambda_latest.zip`, into the S3 source bucket.
6. In the AWS CloudFormation console, choose **Create Stack**.
7. Select **Upload a template to Amazon S3**, choose **HowWhoFeelInVideo.yaml**, then choose **Next**:

The screenshot shows the AWS CloudFormation 'Create stack' wizard. The 'Select Template' step is selected in the left-hand navigation pane. The main content area shows the 'Select Template' section with the instruction: 'Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.' There are two main options: 'Design a template' (with a 'Design template' button) and 'Choose a template'. Under 'Choose a template', there are three radio buttons: 'Select a sample template', 'Upload a template to Amazon S3' (which is selected), and 'Specify an Amazon S3 template URL'. The 'Upload a template to Amazon S3' option has a 'Choose file' button next to it, and the file 'HowWhoFeelinVideo.yaml' is listed. At the bottom right, there are 'Cancel' and 'Next' buttons.

8. Specify the following parameters, and choose **Next**.

- a. **Stack name:** `howwhofeelinvideo` (A unique name for the stack in your AWS region.)
- b. **CollectionId:** The name of the indexed face collection that you created in Step 1: `student`
- c. **FaceMatchThreshold:** Type `70`. The face match threshold ranges from 0 to 100. It specifies the minimum confidence in the face match required to consider it a match.
- d. **PackageBucket:** The name of the S3 source bucket that you created in Step 3.
- e. **VideoBucketName:** The name of the bucket that you want to create. This bucket starts the workflow for .mp4 and .mov files. The bucket name must be unique. You cannot use the bucket name used in the following screenshot. When you delete the AWS CloudFormation stack, this

bucket remains.

Create A New Stack

Secure | <https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/new>

Services ▾ Resource Groups ▾ CloudFormation ▾ Stacks > Create Stack

cyrus @ iverlot N. Virginia Support ▾

### Create stack

Select Template

**Specify Details**

Options

Review

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

### Parameters

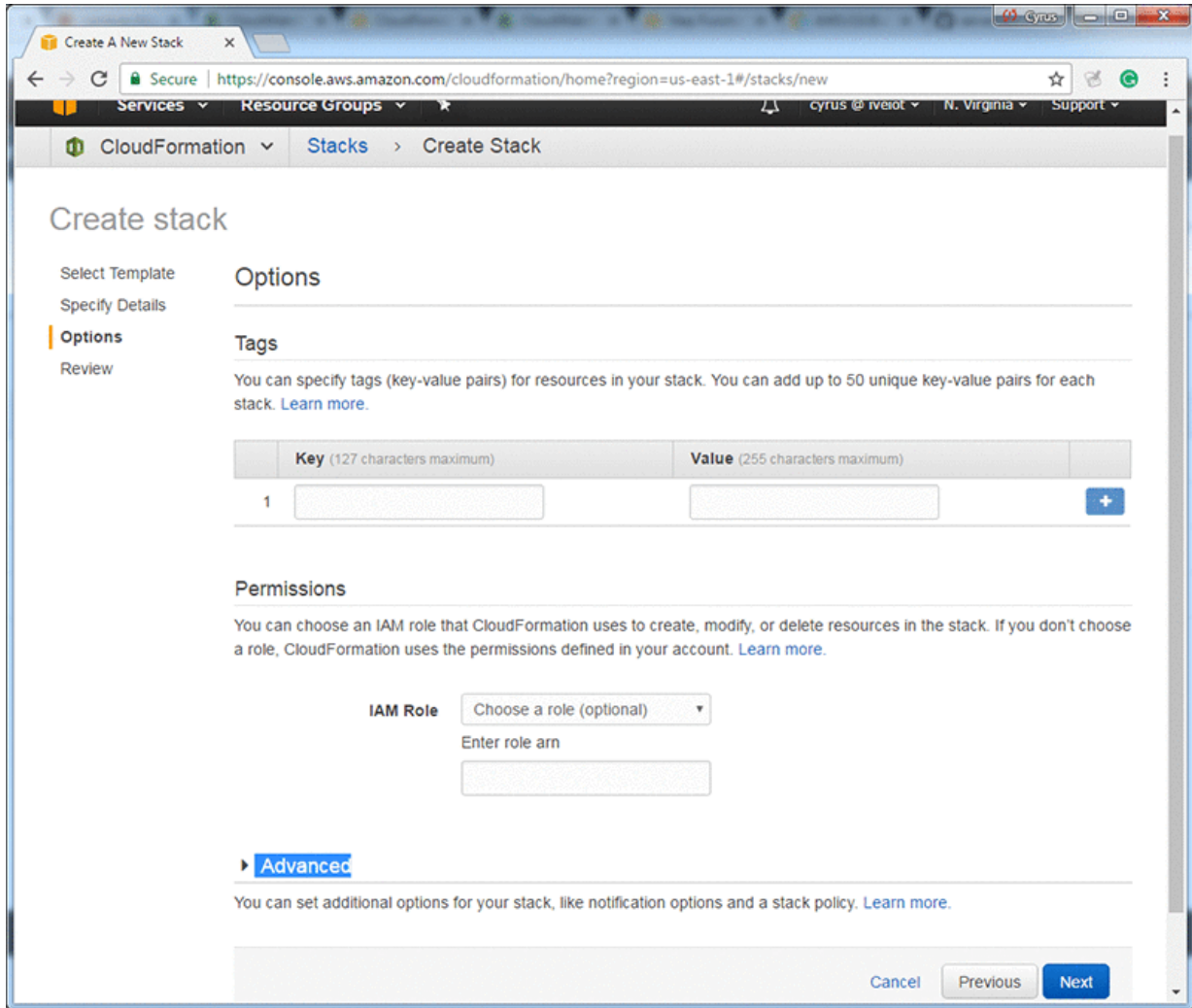
CollectionId

FaceMatchThreshold  Face Match Threshold from 0 to 100

PackageBucket

VideoBucketName   
Bucket will be created and process videos with suffix mp4, move or MOV.

Cancel Previous Next

9. On the Options page, choose **Next**:

The screenshot shows the AWS CloudFormation console's 'Create stack' page, specifically the 'Options' tab. The page is titled 'Create stack' and has a sidebar with 'Select Template', 'Specify Details', 'Options' (selected), and 'Review'. The main content area is divided into sections: 'Options', 'Tags', 'Permissions', and 'Advanced'. The 'Tags' section allows adding key-value pairs for resources. The 'Permissions' section includes an 'IAM Role' dropdown and a text field for 'Enter role arn'. The 'Advanced' section is currently collapsed. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

Create A New Stack

Services Resource Groups

CloudFormation Stacks Create Stack

### Create stack

Select Template  
Specify Details  
**Options**  
Review

#### Options

#### Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 50 unique key-value pairs for each stack. [Learn more.](#)

	Key (127 characters maximum)	Value (255 characters maximum)	
1	<input type="text"/>	<input type="text"/>	<a href="#">+</a>

#### Permissions

You can choose an IAM role that CloudFormation uses to create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses the permissions defined in your account. [Learn more.](#)

**IAM Role**

Enter role arn

**Advanced**

You can set additional options for your stack, like notification options and a stack policy. [Learn more.](#)

[Cancel](#) [Previous](#) [Next](#)

10. Select all acknowledgment boxes, and choose **Create Change Set**:

**Create A New Stack**

Secure | <https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/new>

### Capabilities

**Transforms might require access capabilities**

A transform might add Identity and Access Management (IAM) resources that could provide entities access to make changes to your AWS account. If a transform adds IAM resources, you must acknowledge their capabilities to create or update them. Ensure that you want to create or update the IAM resources, and that they have the minimum required permissions. In addition, if they have custom names, check that the names are unique within your AWS account. [Learn more.](#)

☒ I acknowledge that AWS CloudFormation might create IAM resources.

☒ I acknowledge that AWS CloudFormation might create IAM resources with custom names.

### Transforms

**Check the following transforms: ["AWS::Serverless-2016-10-31"]**

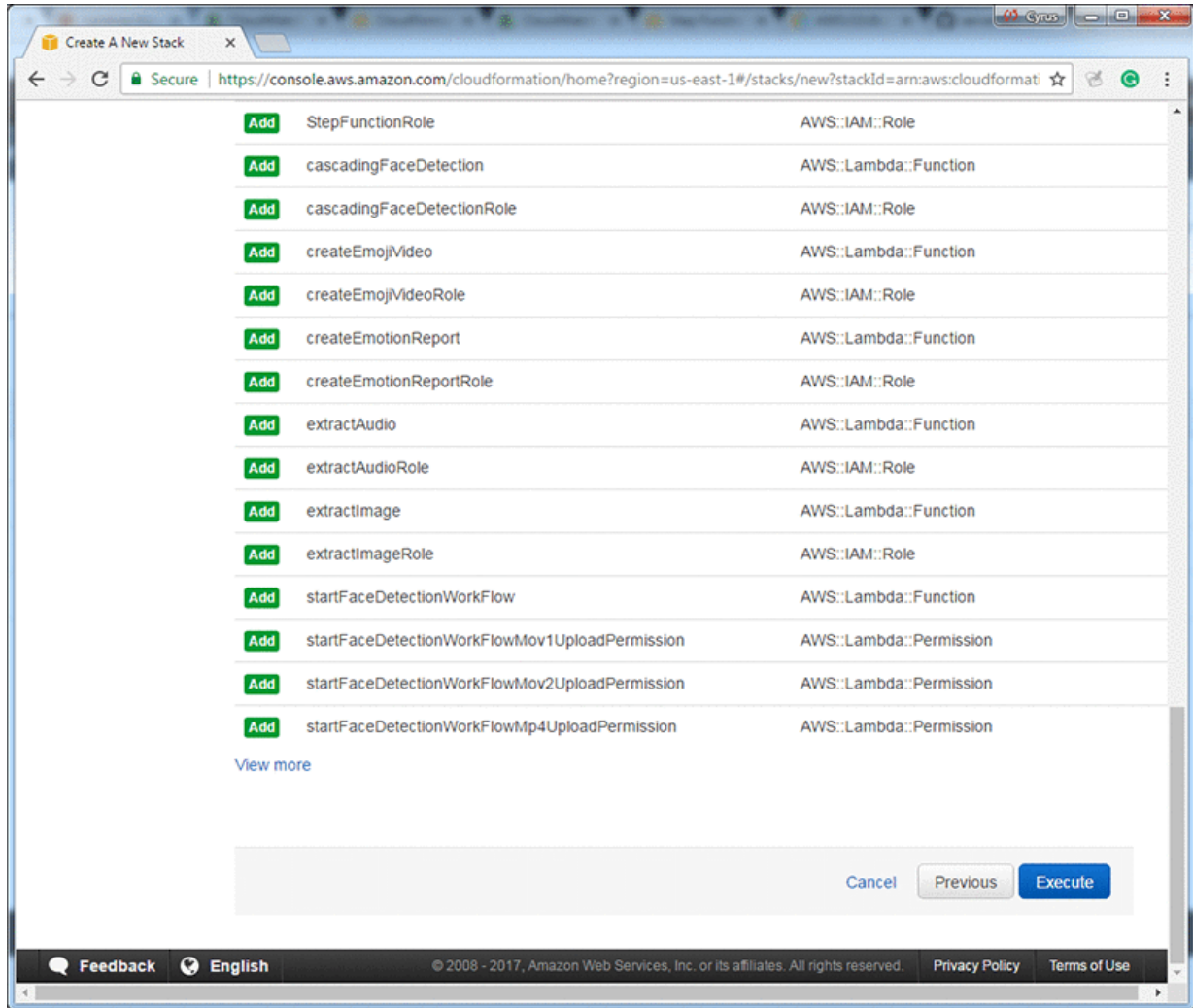
You must use a change set to create this stack because it includes one or more transforms. The change set shows the resources that transforms add to your stack's template. Choose Create Change Set, check the resources that the transforms add, and then choose Execute. [Learn more.](#)

[Create Change Set](#)

[Cancel](#) [Previous](#) [Execute](#)

[Feedback](#) [English](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

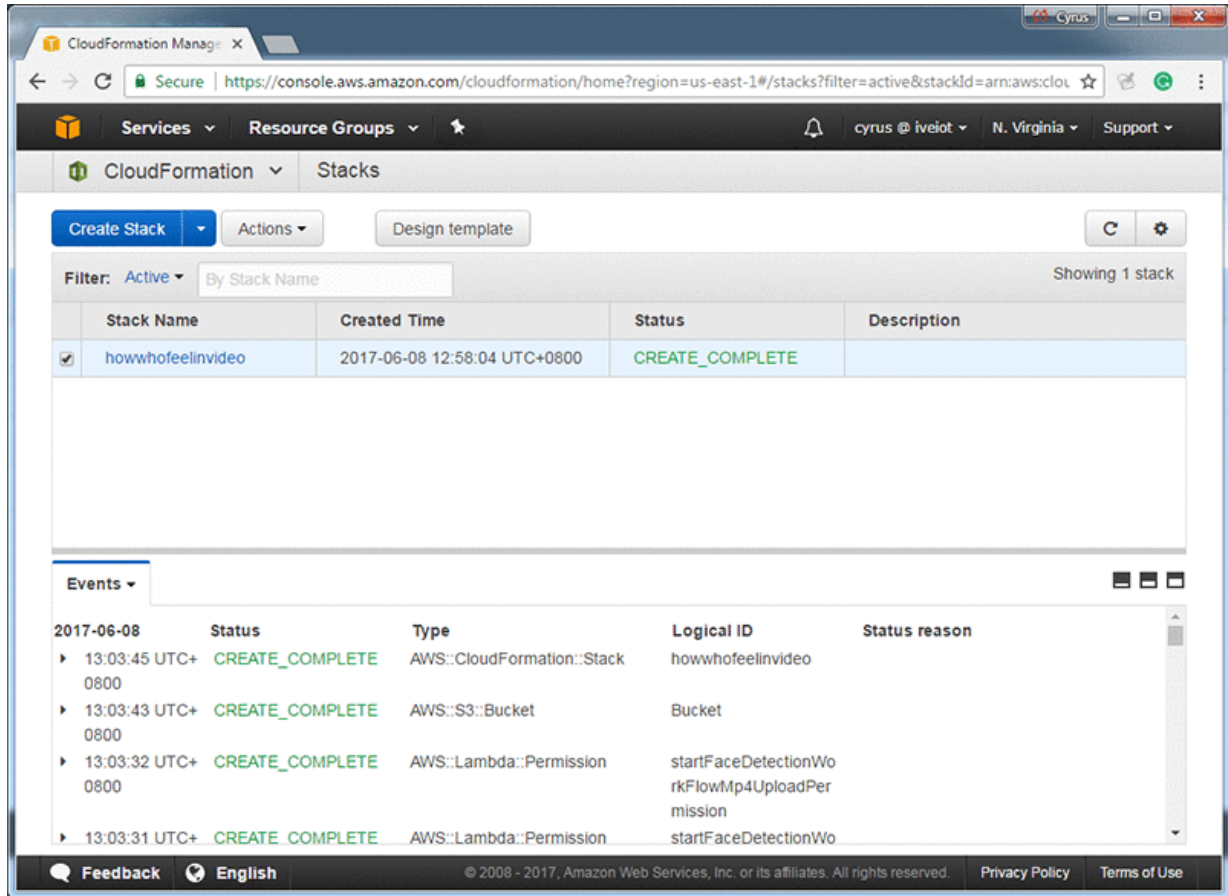


11. When the change set has been created, choose **Execute**:

&gt;



## 12. Wait while the AWS CloudFormation stack is created:



## Try your deployment

1. Go to [S3 console](#) and login into your AWS account.



Account:

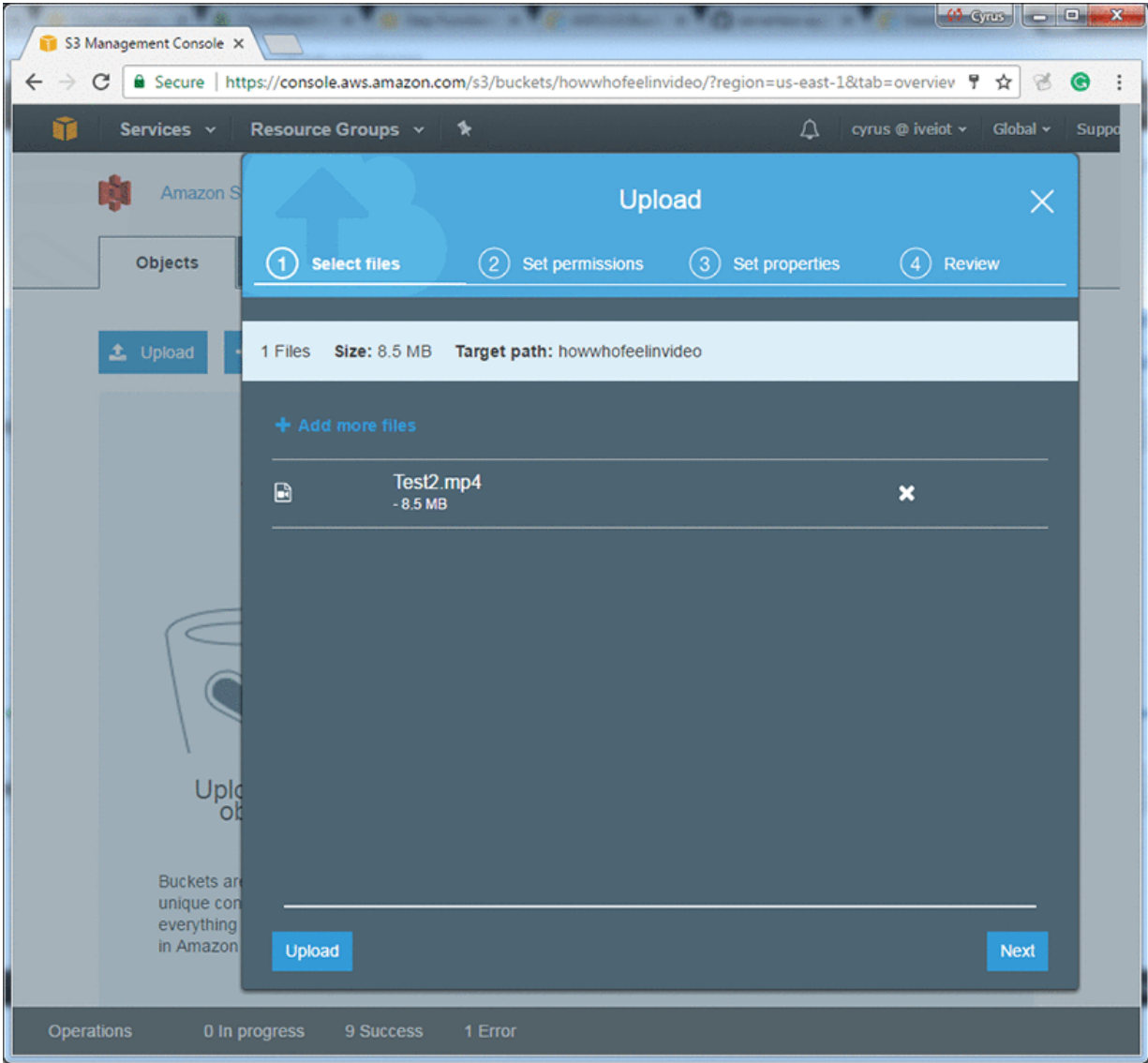
User Name:

Password:

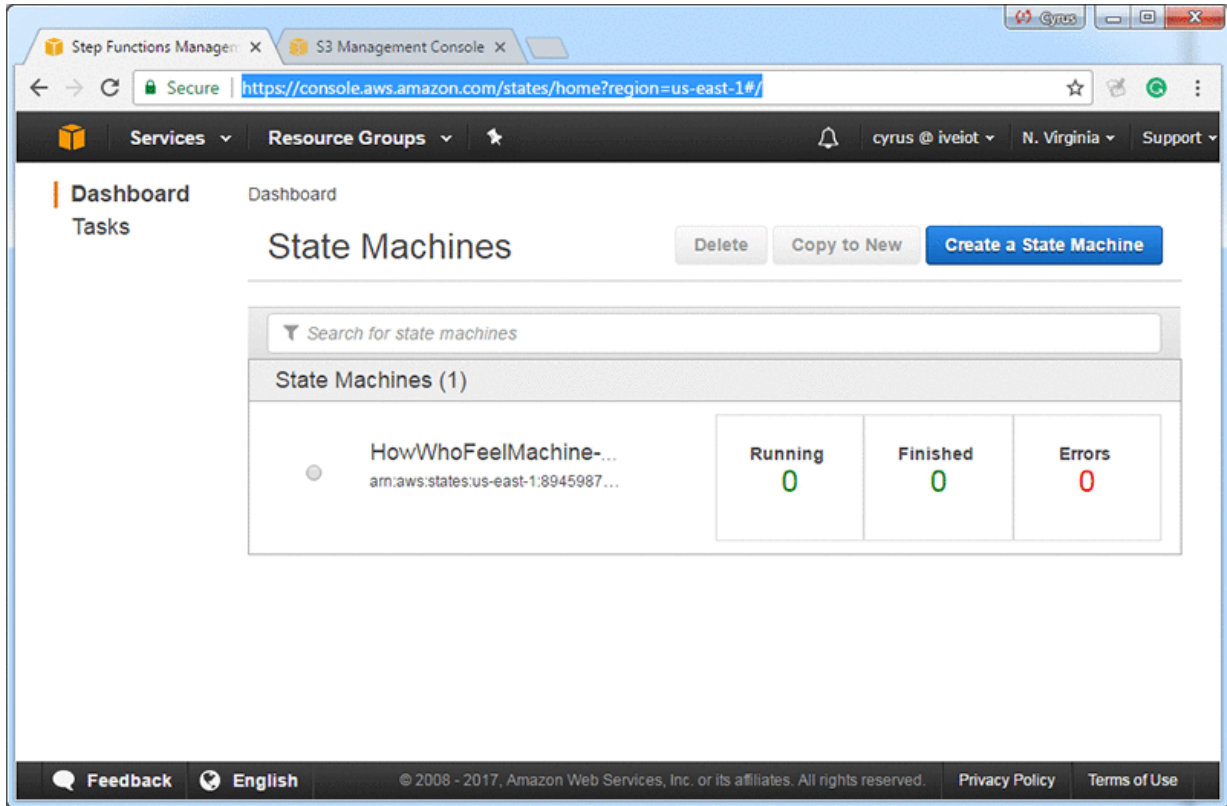
MFA users, enter your code on the next screen.

[Sign-in using root account credentials](#)

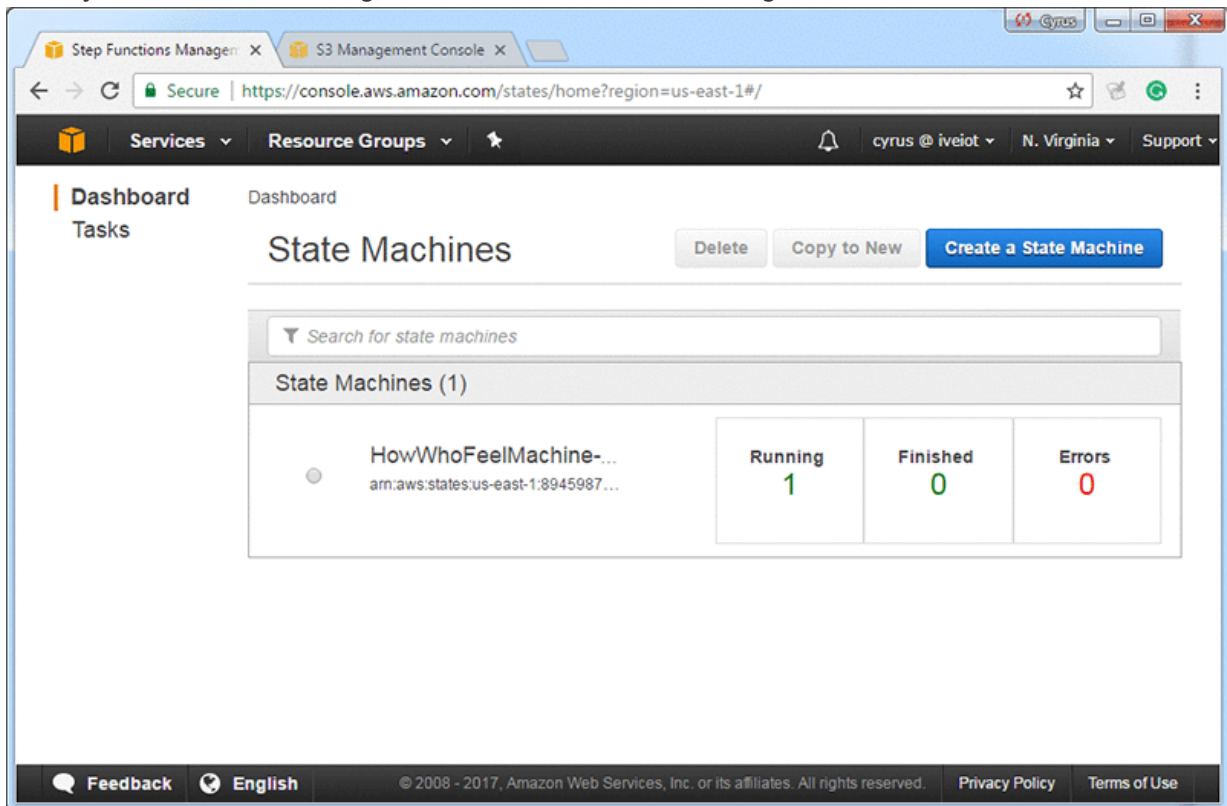
2. In the S3 console, upload a short video into the video bucket. If you are not familiar with the Amazon S3 console, please follow this tutorial: [How Do I Upload Files and Folders to an S3 Bucket?](#)



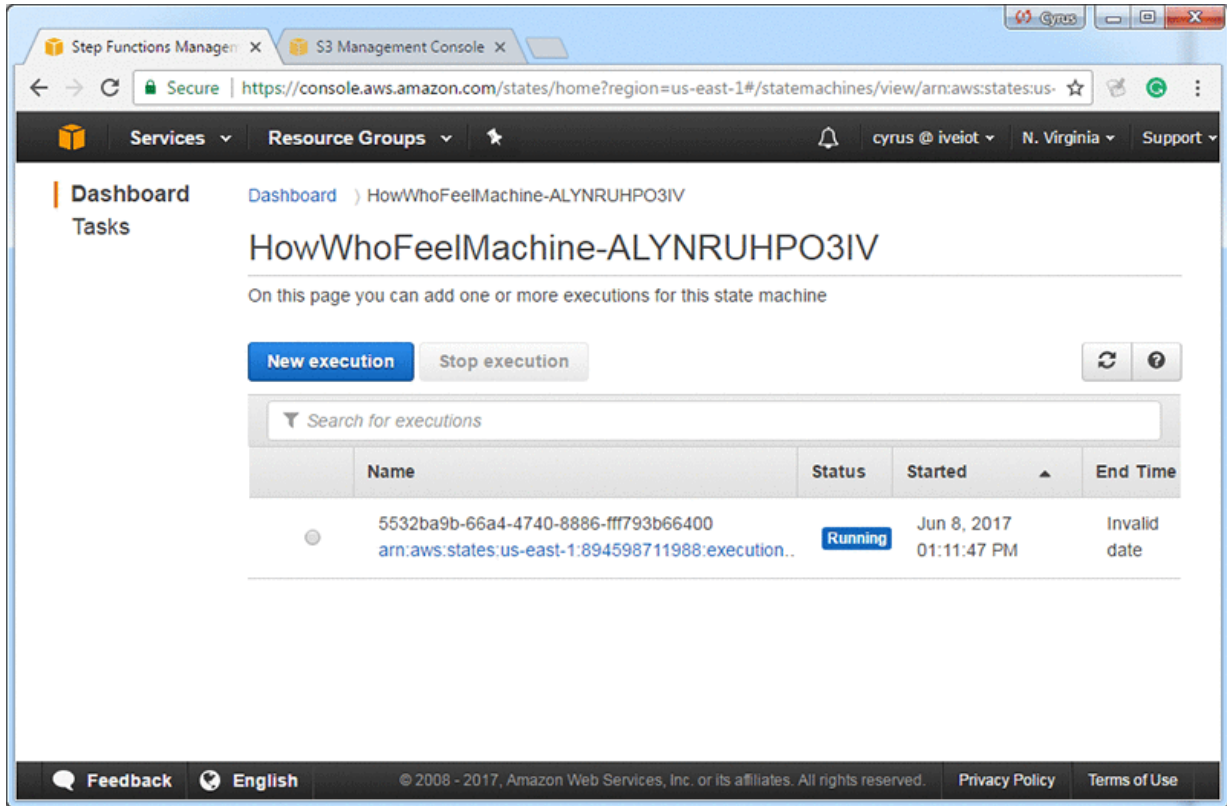
3. In a new browser, open the [Step Function console](#):



4. When you see the task running, choose **State Machines**. You might need to refresh the browser.



## 5. Select the execution instance of the state machine that is running:

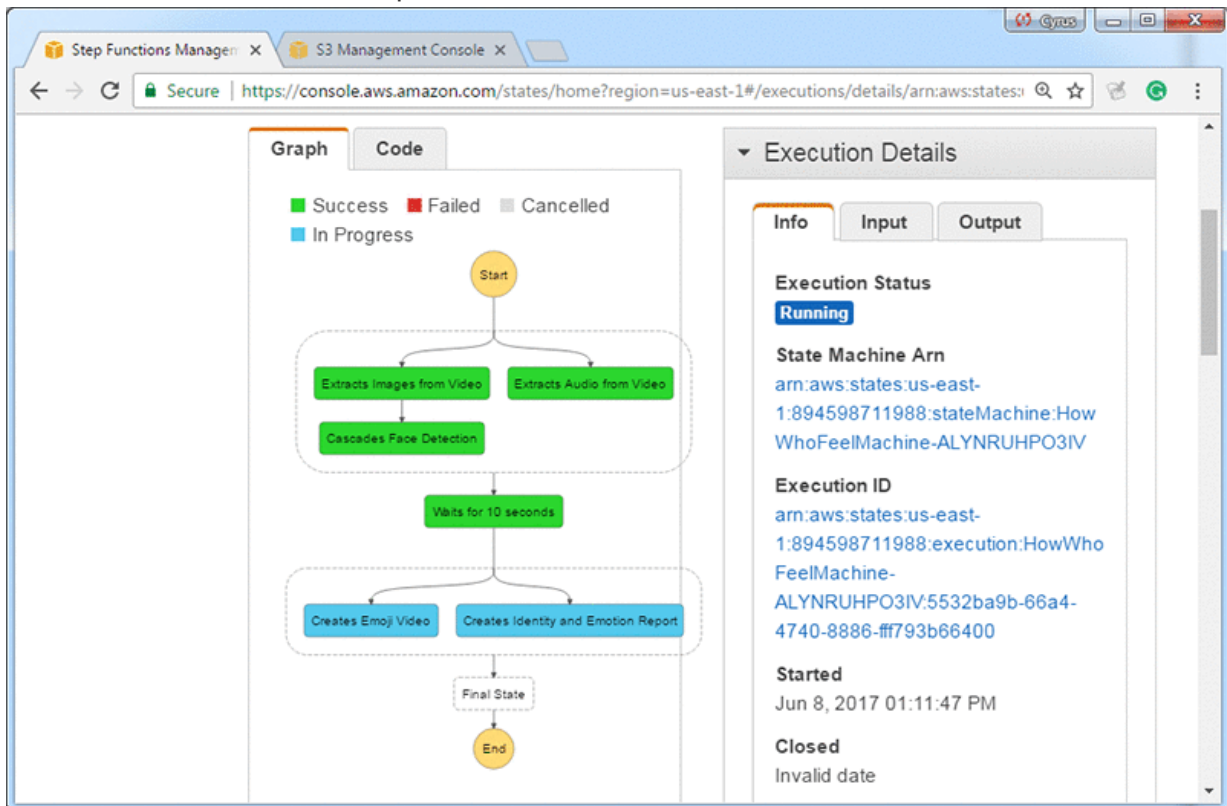


The screenshot shows the AWS Step Functions console. The breadcrumb navigation is **Dashboard > HowWhoFeelMachine-ALYNRUHPO3IV**. The page title is **HowWhoFeelMachine-ALYNRUHPO3IV**. Below the title, it says "On this page you can add one or more executions for this state machine". There are two buttons: **New execution** and **Stop execution**. A search bar is labeled "Search for executions". Below the search bar is a table with the following columns: **Name**, **Status**, **Started**, and **End Time**. The table contains one row with the following data:

Name	Status	Started	End Time
5532ba9b-66a4-4740-8886-fff793b66400 <a href="#">arn:aws:states:us-east-1:894598711988:execution..</a>	Running	Jun 8, 2017 01:11:47 PM	Invalid date

At the bottom of the console, there is a footer with **Feedback**, **English**, copyright information, and links to **Privacy Policy** and **Terms of Use**.

You will see an animation of the process:

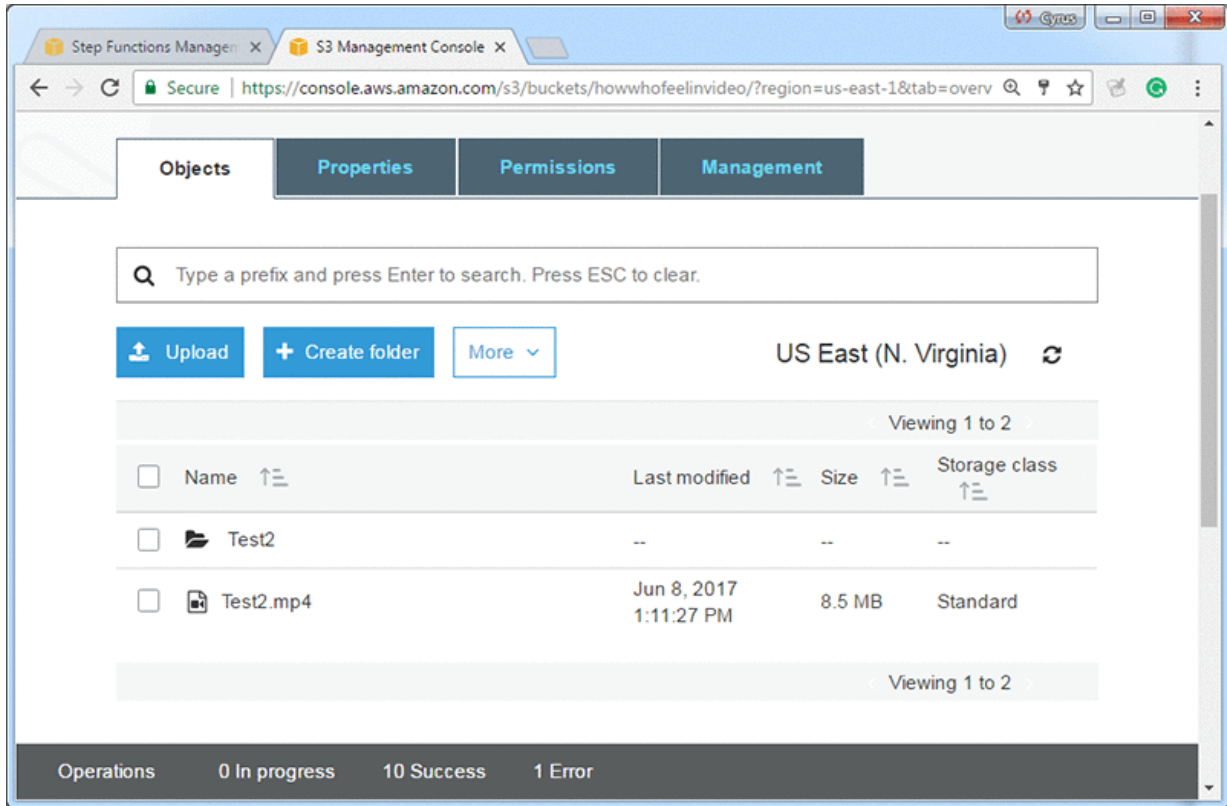


The screenshot shows the AWS Step Functions console with the **Execution Details** tab selected. The breadcrumb navigation is **Step Functions Manager > S3 Management Console > HowWhoFeelMachine-ALYNRUHPO3IV > executions/details/arn:aws:states:us-east-1:894598711988:execution:HowWhoFeelMachine-ALYNRUHPO3IV:5532ba9b-66a4-4740-8886-fff793b66400**. The page title is **Execution Details**. There are three tabs: **Info**, **Input**, and **Output**. The **Info** tab is selected, showing the following information:

- Execution Status**: Running
- State Machine Arn**: [arn:aws:states:us-east-1:894598711988:stateMachine:HowWhoFeelMachine-ALYNRUHPO3IV](#)
- Execution ID**: [arn:aws:states:us-east-1:894598711988:execution:HowWhoFeelMachine-ALYNRUHPO3IV:5532ba9b-66a4-4740-8886-fff793b66400](#)
- Started**: Jun 8, 2017 01:11:47 PM
- Closed**: Invalid date

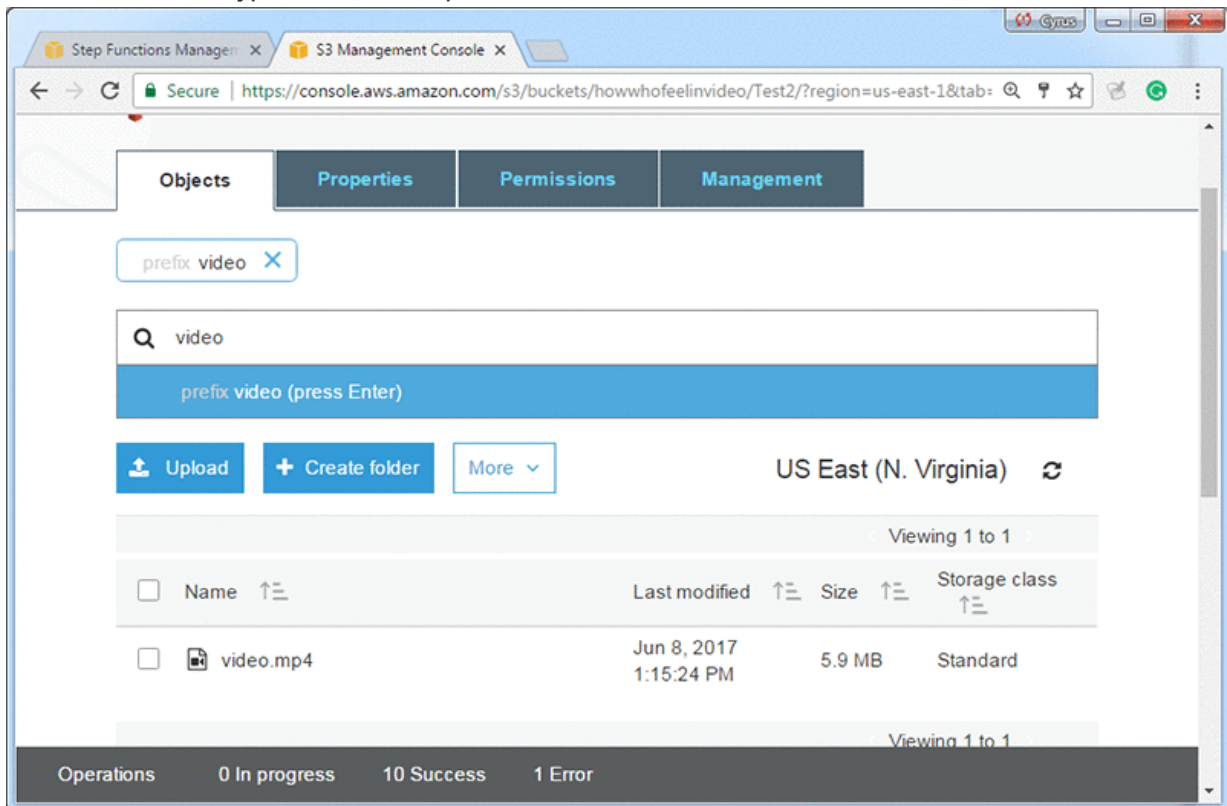
On the left, there is a **Graph** tab showing the state machine workflow. The workflow starts with a **Start** node, followed by a parallel split into **Extracts Images from Video** and **Extracts Audio from Video**. Both lead to **Cascades Face Detection**, which then leads to **Waits for 10 seconds**. This is followed by another parallel split into **Creates Emoji Video** and **Creates Identity and Emotion Report**. Both lead to a **Final State** node, which then leads to an **End** node. A legend indicates: **Success** (green), **Failed** (red), **Cancelled** (grey), and **In Progress** (blue).

6. When the process has completed, refresh the Amazon S3 console. A new folder appears:



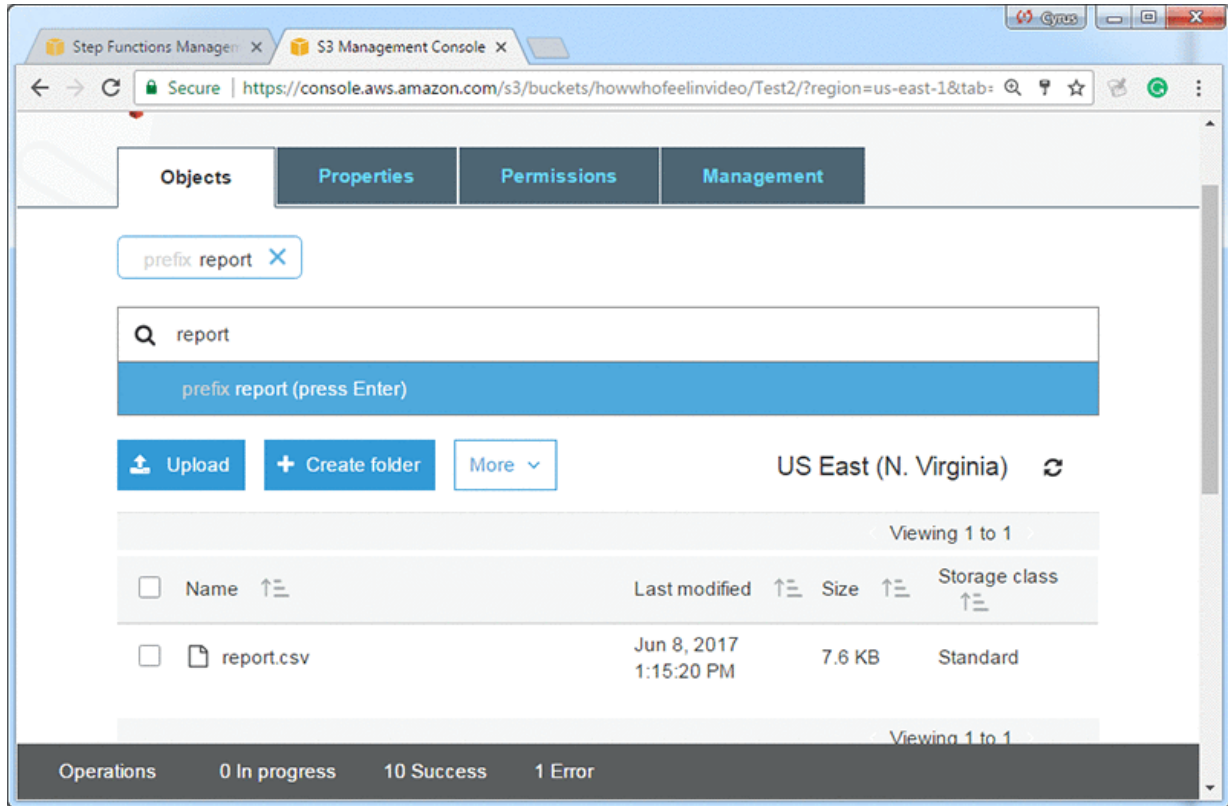
7. Choose the new folder.

8. In the Search box, type video, then open or download the video file:





9. To get the report, in the Search box, type `result`, and open or download the report file.



## Conclusion

HowWhoFeelInVideo can help us understand the emotions of all of the people captured in a video. It has a variety of applications, including education, training, rehabilitative care, and customer interactions. Deployment is simple with an AWS CloudFormation template. Just take a video with your smart phone and upload it to an S3 bucket. In a few minutes, you'll get the emotional analytic report!

This project has been developed in collaboration with four of my students from the IT114115 Higher Diploma in [Cloud and Data Centre Administration](#): [Ng Ka Yin](#), [Lai Kam To](#), [Karlos Lam](#), and [Pang Chin Wing](#). Also, thanks to the [AWS Academy](#) curriculum, which helps my students learn how to use AWS services!

## Additional Reading

Learn how to create a [serverless solution for video frame analysis and alerting with Amazon Rekognition](#).



TAGS: [Amazon Rekognition](#)

## Related Posts

---

[Amazon Rekognition announces updates to its face detection, analysis, and recognition capabilities](#)

[New Case Study: National Geographic Speeds Innovation with Serverless Computing on AWS](#)

[Using Amazon Rekognition to detect celebrities in an iOS app](#)

[Guest Post from Dalet Digital Media Systems: Making Media Businesses Smarter: AI, Cloud and Workflow Orchestration](#)

[What's in Your S3 Bucket? Using Machine Learning to Quickly Visualize and Understand Your Media Files in S3](#)

[Track the number of coffees consumed using AWS DeepLens](#)

[Shopper Sentiment: Analyzing in-store customer experience](#)

[New Engen improves customer acquisition marketing campaigns using Amazon Rekognition](#)