Terry Chau
Final Project for CSCI E90
Fall 2019

Deployment of Serverless Framework for Video Transcoding

The code base for this guide is located here:
https://github.com/TerryChau/FINALPROJECT_E90

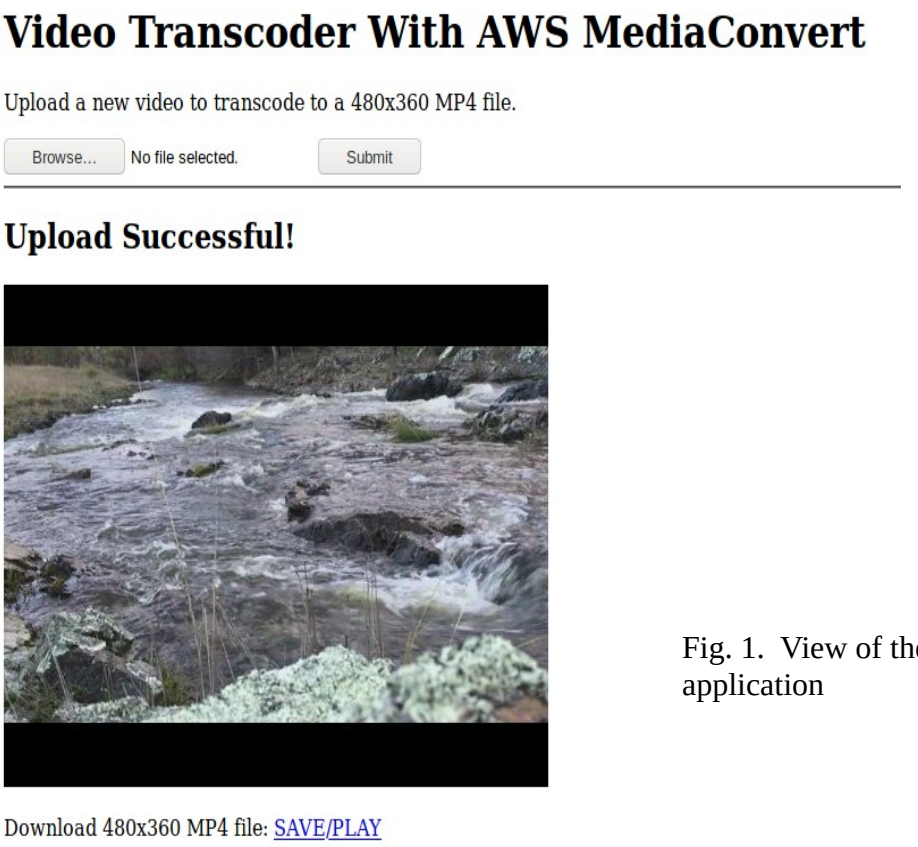


Fig. 1. View of the web application

The following document is a guide for reproducing the 'Video Transcoder with AWS MediaConverter' application. The purpose of the application is to transcode a video to a smaller 480x360 resolution using AWS services. The current on-demand pricing of AWS Elemental MediaConvert for SD resolution is $0.0075 per minute. While similar transcoding jobs can be done on a personal computer, using software such as FFmpeg or Handbrake, with lesser variable costs, MediaConvert is able to leverage the cloud computing power of Amazon Web Services to expedite the job. In essence, MediaConvert trades money for time and convenience.

The tasked performed by the 'Video Transcoder with AWS MediaConverter' application is singular in nature. However, a plurality of AWS technologies are employed in the creation and deployment of this server-less setup. The following represents an architectural diagram of the technologies used for this project.

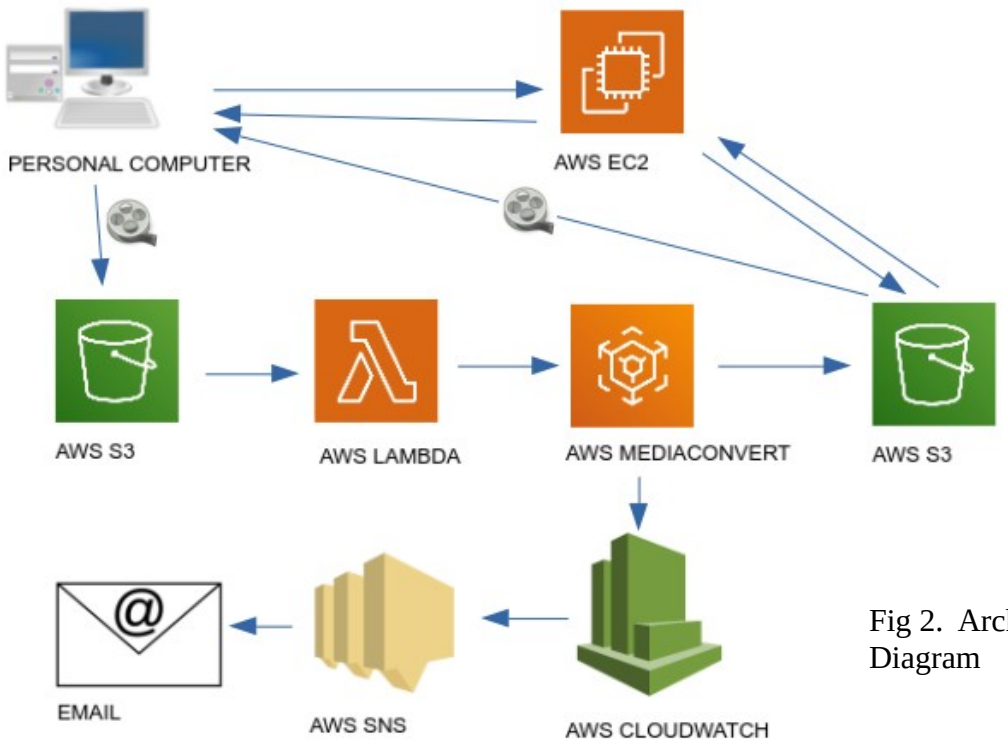


Fig 2. Architecture Diagram

From figure 2, we see that a personal computer interacts with MediaConvert indirectly, since MediaConvert only operates on files in S3 buckets. Lambda is used to monitor the leftmost bucket, and upon changes to the inputs/ folder in that bucket, it queues a new MediaConvert job, which outputs files to the rightmost bucket. CloudWatch monitors MediaConvert for a job state change event, and when the job completion event is detected, a notification is sent by way of the Simple Notification Service to all subscribers. The personal computer, without AWS credentials to the leftmost S3 bucket, can upload files directly to the leftmost S3 bucket, in a buffer-less manner, through the use of Multer-S3. The AWS EC2 instance then monitors the rightmost S3 bucket for the appearance of the converted file, and then provides a link to the converted file for the personal computer.

The usage of Lambda, MediaConvert, CloudWatch, and Simple Notification Service provides for a server-less experience. While the addition of the EC2 instance would turn this server-less setup into a setup with a server, which brings along the associated hassles of running a server, from the security to the infrastructure concerns, the offloading of the heavy computing workload to the cloud means that the EC2 instance could be a very lightweight t2 micro instance. The modular setup segregates issues in one service from the next. Should any service fail, only the failed service needs to be replaced. The EC2 instance also abstracts away the complexity of the system architecture from the end user, so that the end user is not concerned about the plurality of buckets, or the AWS credentials needed to access those buckets.

To setup parts of this architecture, two reference guides were relied upon:

https://github.com/aws-samples/aws-media-services-vod-automation/blob/master/MediaConvert-WorkflowWatchFolderAndNotification/README-tutorial.md

and,
https://github.com/aws-samples/aws-media-services-vod-automation/tree/master/MediaConvert-WorkflowWatchFolderAndNotification

The former guide is detailed in nature, and walks the reader through every step, from the setup of the buckets, to the setup of the notification service. Steps includes creating the buckets, setting permissions and access to the buckets, setting IAM roles for MediaConvert, setting IAM roles for the Lambda function, writing the Lambda function in Python, creating a S3 trigger for the Lambda function, creating a SNS topic for notifications, creating a CloudWatch Event Rule, and permitting CloudWatch to add to SNS topics.

However, this setup can be greatly simplified, if the reader refers to the latter guide. In the latter guide, clicking on 'Launch Stack' would cause CloudFormation, a template service, to generate the aforementioned stacks. The 'Launch Stack' link is reproduced below:

https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/create/template?stackName=emc-watchfolder&templateURL=https://s3.amazonaws.com/rodeolabz-us-east-1/vodtk/1a-mediaconvert-watchfolder/WatchFolder.yaml

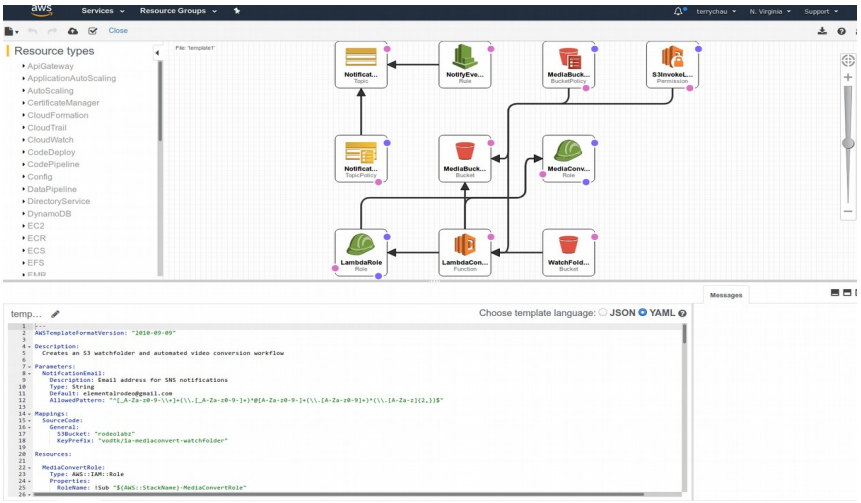


Fig. 3. View of Template Designer in AWS for the VOD Automation Stack

It is noted that the template only works for the US EAST and US West regions. Once the template has started, the only parameter that needs to change is the Notification Email parameter. The email parameter should be set to an email address that the user can confirm.

Once the CloudFormation template is ready, the user can change to output format of AWS MediaConverter by changing the settings in 'jobs.json'. Examples of jobs.json can be found here:

https://docs.aws.amazon.com/mediaconvert/latest/apireference/sample-json.html

The 'jobs.json' used for this project can be found on the github page stated on page 1 of this document. To use this json file, place the file in the S3 bucket emc-watchfolder-watchfolder/jobs. Editing 'jobs.json' in AWS>Lambda>'Function code' is not recommended as the changes do not appear to be saved.

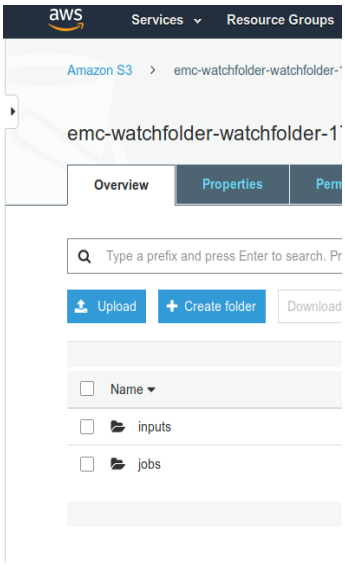


Fig 4. Placement and creation of jobs folder in S3 for emc-watchfolder-watchfolder-...

Next we must create an EC2 instance. The Amazon Linux 2 machine image, t2.micro instance is selected.

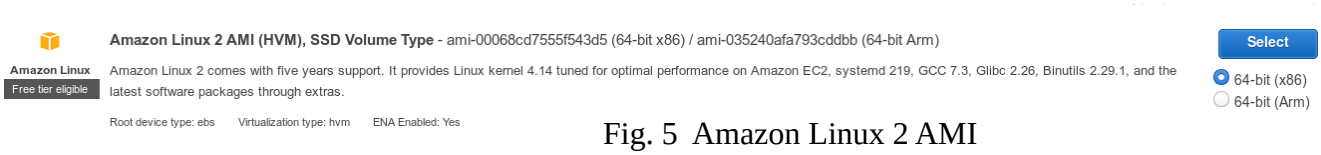
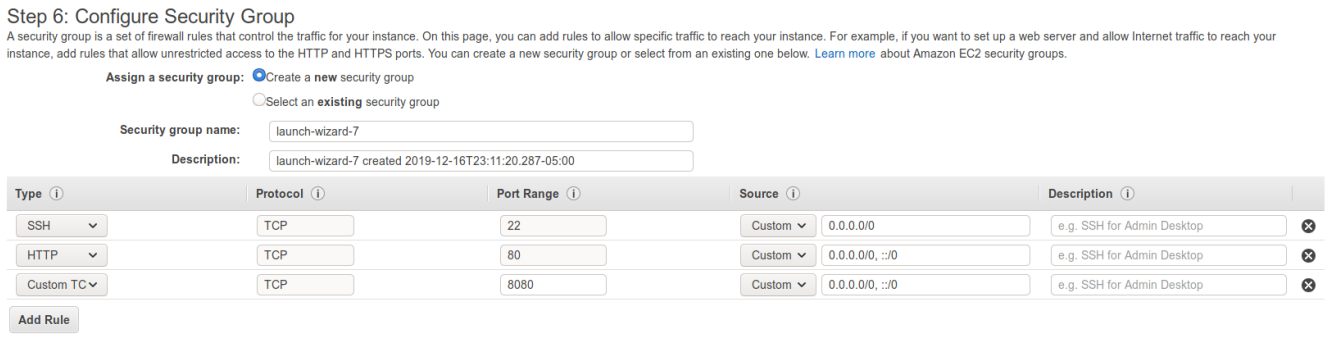


Fig. 5 Amazon Linux 2 AMI

For development purposes in Node.js, security group should include custom TCP port 8080. Alternatively, server.js in the github link on page 1 of this document should be changed to replace port 8080 with 80, and package.json should be changed so that *"start": "ssh node server.js"* is replaced with *"start": "ssh node server.js"*, if these steps are not yet performed. Node.js cannot open ports below 1024 without root.



Now login to the EC2 instance using SSH. See AWS>Services>EC2>Action>Connect> for instructions. Then perform these command in succession, pressing 'y' if asked:

1) sudo yum update
2) sudo yum install git
3) curl -sL https://rpm.nodesource.com/setup_10.x | sudo bash -
4) sudo yum install nodejs

5) git clone https://github.com/TerryChau/FINALPROJECT_E90.git
6) cd FINALPROJECT_E90/server
7) npm install
8) {vim | nano | ed} config.json

Then enter the following:

```
{
   "AWS_ACCESS_KEY": "XXXXXXXXXXXX",
   "AWS_SECRET_ACCESS_KEY": "XXXXXXXXXXXXXXXXXXXXXXXX",
   "AWS_REGION": "XXXXXXXXXXXX",
   "AWS_BUCKET_WATCH": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
   "AWS_BUCKET_MEDIA": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

All values with XXX needs to be filled with appropriate information. The AWS_ACCESS_KEY and AWS_SECRET_ACCESS_KEY can be gotten from AWS>Services>'Add user'. AWS_ACCESS_KEY should be 'us-east-1'. The AWS_BUCKET_WATCH and AWS_BUCKET_MEDIA are the S3 media and watch bucket names created using CloudFormation.

9) npm start
10) Back in AWS>Services, take note of the Public DNS (IPv4) address. Goto this address with a browser, and upload a video file. The video file must be less than 25 MB. This is a security precaution added to Multer in file 'server.js', and can be removed by deleting lines 60-62 of said file.
11) See a result similar to figure 1.

This concludes the instructions for running the 'Video Transcoder With AWS MediaConvert'. The following is a brief discussion of the code base.

From the packages imported into server.js, express and express-handlebars are responsibly for view and routing of files. Aws-sdk, multer, multer-S3 are responsible for the file upload and monitoring of the S3 buckets. Multer-S3 extends the ability of multer so that it can upload to S3 buckets directly without passing through the EC2 server. Additional code was added to the POST to /uploadToAWS route to test for the presence or absence of files in the S3 bucket downstream of MediaConvert. Nested 's2.waitFor('objectExists...' delay the redirect of the browser until MediaConvert finishes the transcoding.

4-5 minute Youtube Summary
https://youtu.be/R72TvCuXqa0

10+ minute Youtube Presentation
https://youtu.be/ntKAx2iR2ig