In order for our videos to play via HTML5 video, we need to first convert them. We want to do this when we upload our videos via the uploader.

Before we start coding our transcoding job we will need to do some setup on AWS. Our first task is to set up out Simple Notification Service (SNS) Topic.

After logging into AWS we will open the services menu and under the messaging category and click **Simple Notification Service**.

To create a new SNS topic we will click the **Create topic** link.

We'll give our topic a name. In this case we'll just name our topic **TranscodingSNS**.

Now that we have our topic we can setup our Transcoding Pipeline, but we will need to comeback to add our webhook.

Our next task is to create two S3 buckets, one for our pre-transcoding videos and one for our post-transcoding videos.

Let's get started by heading over to the S3 service via the service menu.

Click on the **Create Bucket** button

We'll name our first bucket **vids-drop.mytube.com**

Click the **Next** Button

Leave default settings and click the **Next** Button

Leave default settings and click the **Next** Button

Click the **Create Bucket** Button

We will follow the previous steps to create our second bucket named **vids.mytube.com**

Once **vids.mytube.com** is created, click on it, then click on **Permission**

Click on **Bucket Policy** and paste the following policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
                "Sid": "AddPerm",
                "Effect": "Allow",
                "Principal": "*",
                "Action": "s3:GetObject",
                "Resource":
"arn:aws:s3:::vids.mytube.com/*"
          }
     ]
}
```

Click **Save**

Next up we will go ahead and setup our Elastic Transcoder Pipeline

Go back to the Services menu and then click on Elastic Transcoder under the Application Services category.

Then click on the **Create New Pipeline.**

We'll give our pipeline the name **MyTube**

Set **vids-drop.mytube.com** as the input bucket

For our output bucket we'll use **vids.mytube.com** and set the Storage Class to **Standard**

The S3 bucket we'll use for our thumbnails will be **assets.mytube.com** with the Storage Class set to **Standard**

We will need to know when our video has finished transcoding so we'll need to set up a notification.

Expand the Notification tab.

Set the **On Complete Event** to **Use an existing SNS Topic**

In the dropdown select the **TranscodingSNS** we created earlier

Finally click the **Create Pipeline** Button

Now we can setup our webhook

Open up the console again and generate a new webhook controller

```
php artisan make:controller WebhookController
```

Open up the newly created WebhookController and include our required libraries

```
use Aws\Sns\Message;
use Aws\Sns\MessageValidator;
use
Aws\Sns\Exception\InvalidSnsMessageException;
use App\Video;
```

Then we’ll define our index function

```
public function index(){
           $message =
Message::fromRawPostData();
                      $validator = new
MessageValidator();

                      try {
                         $validator-
>validate($message);
                      } catch
(InvalidSnsMessageException $e) {
                         // Pretend we're not
here if the message is invalid.

http_response_code(404);
                         error_log('SNS Message
Validation Error: ' . $e->getMessage());
                         die();
                      }

                      // Check the type of the
message and handle the subscription.
                      if ($message['Type'] ===
'SubscriptionConfirmation') {
                         // Confirm the
subscription by sending a GET request to the
SubscribeURL

file_get_contents($message['SubscribeURL']);
                      }

                      if ($message['Type'] ===
'Notification') {

                                 $payload =
json_decode($message['Message']);

                                 if($payload-
>state == "COMPLETED"){


    $video = Video::where('job_id', $payload-
>jobId)->firstOrFail();
```

```
    $video->update([

    'processed' => true,

    'status' => "Completed"

    ]);

                                        }

                              }
    }
```

In our web.php routes file add the post route to our webhook

Route::post('/webhook', 'WebhookController@index');

And to ensure that we don’t hit a CSRF error we need to add the /webhook route to our VerifyCsrfToken.php middleware located under App > Http > Middleware

To the except array we’ll add

protected $except = [

```
        '/webhook'
    ];
```

Now we can add this webhook to our SNS topic but first we'll need to use ngrok to make our local server available to the outside world.

First serve the application

```
php artisan serve
```

Then we can use ngrok to expose http://localhost:8000 to the world using the following command

```
ngrok http -subdomain=mytube 8000
```

Now we'll have a unique URL that we can use to access our local server.

Open up the TranscodingSNS Topic on AWS

Click **Create Subscription**

Inside the Endpoint Field we'll add our webhook address

https://mytube.ngrok.io/webhook

Then click **Create Subscription**

Now check the new url from the list and click **Request Confirmations**

After a second or two click the refresh icon and you should see that the webhook has been confirmed

Now we need to create a upload video job using artisan.

```
php artisan make:job UploadVideo
```

Now we can open up new job under App > Jobs

We will be making use of the File and Storage Facades so will need to include them, as well as include the Elastic Transcoder Client and Video model.

```
use File;
use Storage;
use Aws\ElasticTranscoder\ElasticTranscoderClient;
use App\Video;
```

Next in out class right before the constructor function we'll define two public variables. One to store the file name and the other to store the instance of the video.

```
public $filename;
public $video;
```

In our constructor we need to pass the video and filename that we'll store in our two variables;

```
public function __construct(Video $video,
$filename)
    {
        $this->filename = $filename;
        $this->video = $video;
    }
```

We will then need to populate our handle function. Our handle function will upload the to S3 and then create a Transcoding job with AWS' Elastic Transcoder.

```
public function handle()
    {
        $file = storage_path() . "/uploads/" .
$this->filename;

        if($s3Url = Storage::disk('videosS3')-
>put($this->filename, fopen($file, 'r+'))){
            File::delete($file);

            $elasticTranscoder =
ElasticTranscoderClient::factory([
                'credentials' => [
                    'key' => env('AWS_KEY'),
                    'secret' => env('AWS_SECRET')
                ],
                'region' => 'us-east-1',
                'version' => 'latest'
            ]);

            $job = $elasticTranscoder-
>createJob([
                'PipelineId' => ‘{PIPELINEID}',
                'Input' => array(
                    'Key' => $this->filename,
                    'FrameRate' => 'auto',
                    'Resolution' => 'auto',
                    'AspectRatio' => 'auto',
                    'Interlaced' => 'auto',
                    'Container' => 'auto',
                ),
                'Outputs' => array(
                    array(
                        'Key' => $this->video-
>uid . '.mp4',
                        'ThumbnailPattern' =>
'thumbs-' . $this->video->uid . '-{count}',
                        'Rotate' => 'auto',
                        'PresetId' =>
'1351620000001-000010',
                    ),
                ),
            ]);

            $jobInfo = $job->get('Job');

            $this->video->update([
                'job_id' => $jobInfo['Id'],
                'status' => 'Transcoding'
            ]);

        }
    }
}
```