About

Imagine being able to capture live video streams, identify objects using deep learning, and then trigger actions or notifications based on the identified objects -- all with low latency and without a single server to manage. This is exactly what this project is going to help you accomplish with AWS. You will be able to setup and run a live video capture, analysis, and alerting solution prototype.
In this post, we present a serverless solution that uses Amazon Rekognition and other AWS services for low-latency video frame analysis. The solution is a prototype that captures a live video, analyzes it contents, and sends an alert when it detects a certain object. We walk through the solution's architecture and explain how the AWS services are integrated. We then give the tools that are needed to configure, build, and run the prototype. Finally, We show you the prototype in action.

Use Case

The prototype addresses a specific use case:
alerting when a human appears in a live video feed from an IP security camera, Web cam on your Laptop, from mobile phone etc. At a high level, it works like this:

1. A camera surveils a particular area, streaming video over the network to a video capture client.
2. The client samples video frames and sends them to AWS services, where they are analyzed and stored with metadata.
3. If Amazon Rekognition detects a certain object—in this case, a human—in the analyzed video frames, an AWS Lambda function sends an Amazon Simple Message Service (Amazon SNS) alert.
4. After you receive an SMS alert, you will likely want to know what caused it. For that, the prototype displays sampled video frames with low latency in a web-based user interface.

How you define low latency depends on the nature of the application. Low latency can range from microseconds to a few seconds. If you use a camera for surveillance, as in our prototype, the time between the capture of unusual activity and the triggering of an alarm can be a few seconds and still be considered a low-latency response. That's without special performance tuning.

Architecture

WikiImage11.jpg

WikiImage2.png

WikiImage3.png

WikiImage4.png

Pre-requisites

Software

Python Pip

```
sudo yum install epel-release
sudo yum install python-pip
```

AWS CLI

```
sudo yum install epel-release
pip install awscli --upgrade --user
sudo pip install --upgrade pip
```

Open JDK Devel

```
sudo yum install epel-release
sudo yum install java-1.8.0-openjdk-devel
```

Implementation

1. Create Python Virtual Environment

Python35
Required for OpenCV Compilation. Since most stable Centos only has Python 2, we will need to use IUS (Inline Upstream Stable)

```
sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm

sudo yum install -y python35u

Test with: python3.5 -V

sudo yum -y install python35u-pip

Use the following commands to install python packages:
sudo pip3.6 install <package_name>

sudo yum -y install python35u-devel
```

Create Python35 Virtual Env

```
  python3.5 -m venv /data/0/python35
  ln -s /data/0/python35 ~/python35
  source ~/python35/bin/activate
```

Alternatively, following method could also be used for Python version 2.*

Follow Steps at https://virtualenv.pypa.io/en/stable/ To Install Python Virtual Environment

```
#/bin/bash

export VIRTUAL_PYTHON_DIR=/data/0/vpython
sudo mkdir -p /data/0
sudo chown -R ${USER}:${USER} /data/0
sudo pip install virtualenv

virtualenv $VIRTUAL_PYTHON_DIR
ln -fs  $VIRTUAL_PYTHON_DIR $HOME/vpython

source $HOME/vpython/bin/activate

Note: For some reason "source $HOME/vpython/bin/activate" needs to be run manually as well.
```

```
After activation you MUST see (vpython) as the very first word in your command prompt. If not plea
se check your $PATH.
```

2. AWS Policies

Make sure that access keys associated with the IAM user has the following permissions:

```
Amazon S3
Amazon DynamoDB
Amazon Kinesis
AWS Lambda
Amazon CloudWatch and CloudWatch Logs
AWS CloudFormation
Amazon Rekognition
Amazon SNS
Amazon API Gateway
Creating IAM Roles
```

3. OpenCV

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposite to the C-based OpenCV 1.x API. The latter is described in opencv1x.pdf.
OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image processing - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- features2d - salient feature detectors, descriptors, and descriptor matchers.
- objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- highgui - an easy-to-use interface to simple UI capabilities.
- videoio - an easy-to-use interface to video capturing and video codecs.
- gpu - GPU-accelerated algorithms from different OpenCV modules.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

Centos7

Ubuntu

1. Install RPMS

```
sudo apt-get install -y xfsprogs
sudo apt-get install -y python-pip
sudo apt-get install -y python
sudo apt-get install -y cmake
sudo apt-get install -y python-devel numpy
sudo apt-get install -y gcc gcc-c++
sudo apt-get install -y gtk2-devel
sudo apt-get install -y libdc1394-devel
sudo apt-get install -y libv4l-devel
sudo apt-get install -y gstreamer-plugins-base-devel
sudo apt-get install -y libpng-devel libjpeg-turbo-devel jasper-devel openexr-devel libtiff-deve
```

```
l libwebp-devel
  sudo apt-get install -y tbb-devel
  sudo apt-get install -y eigen3-devel
  sudo apt-get install -y python-sphinx texlive
```

1.1 Install FFMPEG-DEVEL RPM

```
    CENTOS / REDHAT
-------------------------
    sudo yum localinstall --nogpgcheck https://download1.rpmfusion.org/free/el/rpmfusion-free-rele
ase-7.noarch.rpm
    https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-7.noarch.rpm
    sudo yum install ffmpeg-devel

    UBUNTU
---------------------------
    sudo apt-get install ffmpeg
```

1.2 Update all packages and reboot

```
sudo yum update -y
sudo reboot
```

1.3 Actiavate python35 Virtual Env

```
 source ~/python35/bin/activate
```

2 Clone following GIT Projects:

```
mkdir -p ~/projects
cd ~/projects

git clone git@github.com:opencv/opencv.git
git clone git@github.com:opencv/opencv_contrib.git
```

3. Configuring and Installing

Now we have installed all the required dependencies, let's install OpenCV. Installation has to be configured with CMake. It specifies which modules are to be installed, installation path, which additional libraries to be used, whether documentation and examples to be compiled etc. Below command is normally used for configuration (executed from build folder).

```
cd $HOME
mkdir build
cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

```
Note: if the above step throws compilation errors like "Could NOT find PythonInterp: Found unsuita
ble version '1.4'", then
  1. Remove * from build dir.
  2. Re-Run the 'cmake' command above.
```

4. Threading Building Blocks (TBB) and Eigen (Optimized mathematical Operations)

Several OpenCV functions are parallelized with Intel's Threading Building Blocks (TBB). But if you want to enable it, you need to install TBB first. (Also while configuring installation with CMake, don't forget to pass -D WITH_TBB=ON. More details below.)

OpenCV uses another library Eigen for optimized mathematical operations. So if you have Eigen installed in your system, you can exploit it. (Also while configuring installation with CMake, don't forget to pass -D WITH_EIGEN=ON. More details below.)

```
cmake -D WITH_TBB=ON -D WITH_EIGEN=ON ..
```

5. Disable GPU related modules

Since we use OpenCV-Python, we don't need GPU related modules. It saves us some time

```
cmake -D WITH_OPENCL=OFF -D WITH_CUDA=OFF -D BUILD_opencv_gpu=OFF -D BUILD_opencv_gpuarithm=OFF -D
 BUILD_opencv_gpubgsegm=OFF -D BUILD_opencv_gpucodec=OFF -D BUILD_opencv_gpufeatures2d=OFF -D BUIL
D_opencv_gpufilters=OFF -D BUILD_opencv_gpuimgproc=OFF -D BUILD_opencv_gpulegacy=OFF -D BUILD_open
cv_gpuoptflow=OFF -D BUILD_opencv_gpustereo=OFF -D BUILD_opencv_gpuwarping=OFF ..
```

6. Output
Run the following:

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

Please verify all the relevant fields below before moving further:

```
--   GUI:
--     QT:                          NO
--     GTK+ 2.x:                    YES (ver 2.24.31)
--     GThread :                    YES (ver 2.50.3)
--     GtkGlExt:                    NO
--     OpenGL support:              NO
--     VTK support:                 NO
--
--   Media I/O:
--     ZLib:                        /lib64/libz.so (ver 1.2.7)
--     JPEG:                        /lib64/libjpeg.so (ver )
--     WEBP:                        /lib64/libwebp.so (ver encoder: 0x0201)
--     PNG:                         /lib64/libpng.so (ver 1.5.13)
--     TIFF:                        /lib64/libtiff.so (ver 42 - 4.0.3)
--     JPEG 2000:                   /lib64/libjasper.so (ver 1.900.1)
--     OpenEXR:                     /lib64/libImath.so /lib64/libIlmImf.so /lib64/libIex.so /lib64
/libHalf.so /lib64/libIlmThread.so (ver 1.7.1)
--     GDAL:                        NO
--     GDCM:                        NO
--
--   Video I/O:
--     DC1394 1.x:                  NO
--     DC1394 2.x:                  YES (ver 2.2.2)
--     FFMPEG:                      YES
--       avcodec:                   YES (ver 56.60.100)
--       avformat:                  YES (ver 56.40.101)
```

```
--       avutil:                     YES (ver 54.31.100)
--       swscale:                    YES (ver 3.1.101)
--       avresample:                 YES (ver 2.1.0)
--     GStreamer:
--       base:                       YES (ver 0.10.36)
--       video:                      YES (ver 0.10.36)
--       app:                        YES (ver 0.10.36)
--       riff:                       YES (ver 0.10.36)
--       pbutils:                    YES (ver 0.10.36)
--     OpenNI:                       NO
--     OpenNI PrimeSensor Modules:   NO
--     OpenNI2:                      NO
--     PvAPI:                        NO
--     GigEVisionSDK:                NO
--     Aravis SDK:                   NO
--     UniCap:                       NO
--     UniCap ucil:                  NO
--     V4L/V4L2:                     NO/YES
--     XIMEA:                        NO
--     Xine:                         NO
--     Intel Media SDK:              NO
--     gPhoto2:                      NO
--
--   Parallel framework:             TBB (ver 4.1 interface 6103)
--
--   Trace:                          YES (with Intel ITT)
--
--   Other third-party libraries:
--     Use Intel IPP:                2017.0.3 [2017.0.3]
--                at:                /home/upendern/projects/opencv/build/3rdparty/ippicv/ippicv_ln
x
--     Use Intel IPP IW:             sources (2017.0.3)
--                   at:             /home/upendern/projects/opencv/build/3rdparty/ippicv/ippiw_lnx
--     Use VA:                       NO
--     Use Intel VA-API/OpenCL:      NO
--     Use Lapack:                   NO
--     Use Eigen:                    YES (ver 3.2.5)
--     Use Cuda:                     NO
--     Use OpenCL:                   NO
--     Use OpenVX:                   NO
--     Use custom HAL:               NO
--
--   Python 2:
--     Interpreter:                  /bin/python2.7 (ver 2.7.5)
--     Libraries:                    /lib64/libpython2.7.so (ver 2.7.5)
--     numpy:                        /usr/lib64/python2.7/site-packages/numpy/core/include (ver 1.7
.1)
--     packages path:                lib/python2.7/site-packages
--
--   Python 3:
--     Interpreter:                  /data/0/python35/bin/python3 (ver 3.5.4)
--
--   Python (for build):             /bin/python2.7
--
--   Java:
--     ant:                          NO
--     JNI:                          /usr/lib/jvm/java/include /usr/lib/jvm/java/include/linux /usr
/lib/jvm/java/include
--     Java wrappers:                NO
--     Java tests:                   NO
--
--   Matlab:                         Matlab not found or implicitly disabled
--
--   Documentation:
--     Doxygen:                      NO
--
--   Tests and samples:
```

```
--     Tests:                       YES
--     Performance tests:           YES
--     C/C++ Examples:              NO
```

7. Build Files

Now you build the files using make command and install it using make install command. make install should be executed as root.

```
make
su
make install
```

8. OpenCV Module Installtion is Over

Installation is over. All files are installed in /usr/local/ folder. But to use it, your Python should be able to find OpenCV module. You have two options for that.

```
1. Move the module to any folder in Python Path
   cp ~/projects/opencv/build/lib/cv2.so ~/python35/lib64/python3.5/site-packages/cv2.so
   cp ~/projects/opencv/build/lib/cv2.so /usr/lib/python2.7/site-packages/
```

4. Boto3

What is Boto3

Boto is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. Boto provides an easy to use, object-oriented API as well as low-level direct service access.

How to Install

http://boto3.readthedocs.io/en/latest/guide/quickstart.html#installation

```
pip install boto3
```

Example

```
import boto3

# Let's use Amazon S3
s3 = boto3.resource('s3')
Now that you have an s3 resource, you can make requests and process responses from the service. Th
e following uses the buckets collection to print out all bucket names:
```

```
# Print out bucket names
for bucket in s3.buckets.all():
    print(bucket.name)
```

5. Pynt

What is Pynt

Pynt enables you to write project build scripts in Python
Easy to learn.
Build tasks are just python funtions.
Manages dependencies between tasks.
Automatically generates a command line interface.
Rake style param passing to tasks
Supports python 2.7 and python 3.x

How to Install

```
pip install pynt
```

Example

```
#!/usr/bin/python

import sys
from pynt import task

@task()
def clean():
    '''Clean build directory.'''
    print 'Cleaning build directory...'

@task(clean)
def html(target='.'):
    '''Generate HTML.'''
    print 'Generating HTML in directory "%s"' %  target

@task(clean, ignore=True)
def images():
    '''Prepare images.'''
    print 'Preparing images...'

@task(html,images)
def start_server(server='localhost', port = '80'):
    '''Start the server'''
    print 'Starting server at %s:%s' % (server, port)

@task(start_server) #Depends on task with all optional params
def stop_server():
    print 'Stopping server....'
```

```
@task()
def copy_file(src, dest):
    print 'Copying from %s to %s' % (src, dest)

@task()
def echo(*args,**kwargs):
    print args
    print kwargs

# Default task (if specified) is run when no task is specified in the command line
# make sure you define the variable __DEFAULT__ after the task is defined
# A good convention is to define it at the end of the module
# __DEFAULT__ is an optional member

__DEFAULT__=start_server
```

6. Pyzt

pip install pytz -t amazon-rekognition-video-analyzer/lambda/imageprocessor/

What is Pyzt

Pytz is needed for timezone calculations.
Pytz brings the Olson tz database into Python. This library allows accurate and cross platform timezone calculations using Python 2.4 or higher.
It also solves the issue of ambiguous times at the end of daylight saving time, which you can read more about in the Python Library Reference (datetime.tzinfo).

Example

```
>>> from datetime import datetime, timedelta
>>> from pytz import timezone
>>> import pytz
>>> utc = pytz.utc
>>> utc.zone
'UTC'
>>> eastern = timezone('US/Eastern')
>>> eastern.zone
'US/Eastern'
>>> amsterdam = timezone('Europe/Amsterdam')
>>> fmt = '%Y-%m-%d %H:%M:%S %Z%z'
```

How to Install Pytz

```
source ~/python35/bin/activate
pip install pytz (in virtual env)
pip install pytz -t amazon-rekognition-video-analyzer/lambda/imageprocessor/
```

Structure

```
amazon-rekognition-video-analyzer/
├── lambda
│   └── imageprocessor
│       ├── pytz
│       │   ├── exceptions.py
│       │   ├── __init__.py
│       │   ├── lazy.py
│       │   ├── __pycache__
│       │   │   ├── exceptions.cpython-35.pyc
│       │   │   ├── __init__.cpython-35.pyc
│       │   │   ├── lazy.cpython-35.pyc
│       │   │   ├── reference.cpython-35.pyc
│       │   │   ├── tzfile.cpython-35.pyc
│       │   │   └── tzinfo.cpython-35.pyc
│       │   ├── reference.py
│       │   ├── tzfile.py
│       │   ├── tzinfo.py
│       │   └── zoneinfo
│       │       ├── Africa
│       │       ├── America
│       │       ├── Antarctica
│       │       ├── Arctic
│       │       ├── Asia
│       │       ├── Atlantic
│       │       ├── Australia
│       │       ├── Brazil
│       │       ├── Canada
│       │       ├── CET
│       │       ├── Chile
│       │       ├── CST6CDT
│       │       ├── Cuba
│       │       ├── EET
│       │       ├── Egypt
│       │       ├── Eire
│       │       ├── EST
│       │       ├── EST5EDT
│       │       ├── Etc
│       │       ├── Europe
│       │       ├── Factory
│       │       ├── GB
│       │       ├── GB-Eire
│       │       ├── GMT
│       │       ├── GMT0
│       │       ├── GMT-0
│       │       ├── GMT+0
│       │       ├── Greenwich
│       │       ├── Hongkong
│       │       ├── HST
```

```
|       |       ├── Iceland
|       |       ├── Indian
|       |       ├── Iran
|       |       ├── iso3166.tab
|       |       ├── Israel
|       |       ├── Jamaica
|       |       ├── Japan
|       |       ├── Kwajalein
|       |       ├── Libya
|       |       ├── localtime
|       |       ├── MET
|       |       ├── Mexico
|       |       ├── MST
|       |       ├── MST7MDT
|       |       ├── Navajo
|       |       ├── NZ
|       |       ├── NZ-CHAT
|       |       ├── Pacific
|       |       ├── Poland
|       |       ├── Portugal
|       |       ├── posixrules
|       |       ├── PRC
|       |       ├── PST8PDT
|       |       ├── ROC
|       |       ├── ROK
|       |       ├── Singapore
|       |       ├── Turkey
|       |       ├── UCT
|       |       ├── Universal
|       |       ├── US
|       |       ├── UTC
|       |       ├── WET
|       |       ├── W-SU
|       |       ├── zone1970.tab
|       |       ├── zone.tab
|       |       └── Zulu
|       └── pytz-2017.2.dist-info
|           ├── DESCRIPTION.rst
|           ├── INSTALLER
|           ├── METADATA
|           ├── metadata.json
|           ├── RECORD
|           ├── top_level.txt
|           ├── WHEEL
|           └── zip-safe
└── README.md
```

7. Building Prototype

Common interactions with the project have been simplified for you. Using pynt, the following tasks are automated with simple commands:

```
1. Creating, deleting, and updating the AWS infrastructure stack with AWS CloudFormation
2. Packaging lambda code into .zip files and deploying them into an Amazon S3 bucket
3. Running the video capture client to stream from a built-in laptop webcam or a USB camera
4. Running the video capture client to stream from an IP camera (MJPEG stream)
5. Build a simple web user interface (Web UI)
6. Run a lightweight local HTTP server to serve Web UI for development and demo purposes

For a list of all available tasks, enter the following command in the root directory of this proje
ct:
```

8. Configuring the Project

aws-infra-cfn.yaml

```
AWSTemplateFormatVersion: 2010-09-09

Description: The CloudFormation template for AWS resources required by amazon rekognition video an
alyzer.

Parameters:

  SourceS3BucketParameter:
    Type: String
    MinLength: "1"
    Description: "Enter the name of the S3 bucket containing source .zip files."

  ImageProcessorSourceS3KeyParameter:
    Type: String
    MinLength: "1"
    Description: "Enter the name of the S3 key of Image Processor lambda function .zip file."

  FrameFetcherSourceS3KeyParameter:
    Type: String
    MinLength: "1"
    Description: "Enter the name of the S3 key of Frame Fetcher lambda function .zip file."

  S3ImageGetterSourceS3KeyParameter:
    Type: String
    MinLength: "1"
    Description: "Enter the name of the S3 key of Image Getter lambda function .zip file."

  FrameFetcherLambdaFunctionName:
    Type: String
    Default: "framefetcher"
    Description: "Name of the Lambda function that fetches frame metadata from DynamoDB."

  ImageProcessorLambdaFunctionName:
    Type: String
    Default: "imageprocessor"
    Description: "Name of the Lambda function that receives and processes frame images."

  ImageGetterLambdaFunctionName:
    Type: String
    Default: "s3"
    Description: "Name of the Lambda function that Gets Images from S3 and sends to Kinesis Stream
."

  FrameFetcherApiResourcePathPart:
    Type: String
    Default: "enrichedframe"
    Description: "Path part for the API Gateway resource to access FrameFetcher lambda function."
```

```
  KinesisStreamNameParameter:
    Type: String
    Default: "FrameStream"
    Description: "Name of the Kinesis stream to receive frames from video capture client."

  FrameS3BucketNameParameter:
    Type: String
    MinLength: "1"
    Description: "Name of the S3 bucket for storage of captured frames."

  ImageReKogS3BucketNameParameter:
    Type: String
    MinLength: "1"
    Description: "Name of the S3 bucket for storage of Images to be Sent to Kinesis for Image Reko
gnition."

  DDBTableNameParameter:
    Type: String
    Default: "EnrichedFrame"
    Description: "Name of the DynamoDB table for persistence & querying of captured frames metadat
a."

  DDBGlobalSecondaryIndexNameParameter:
    Type: String
    Default: "processed_year_month-processed_timestamp-index"
    Description: "Name of the DDB Global Secondary Index for querying of captured frames by Web UI
."

  ApiGatewayRestApiNameParameter:
    Type: String
    Default: "RtRekogRestApi"
    Description: "Name of the API Gateway Rest API."

  ApiGatewayStageNameParameter:
    Type: String
    Default: "development"
    Description: "Name of the API Gateway stage."

  ApiGatewayUsagePlanNameParameter:
    Type: String
    Default: "development-plan"
    Description: "Name of the API Gateway Usage Plan."

Resources:

  FrameS3Bucket:
    Type: "AWS::S3::Bucket"
    Properties:
      BucketName: !Ref FrameS3BucketNameParameter

  S3ImageGetterLambdaExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::721045248511:policy/AmazonKinesisFullAccess
        - arn:aws:iam::721045248511:policy/AmazonRekognitionFullAccess
        - arn:aws:iam::721045248511:policy/AmazonS3FullAccess
        - arn:aws:iam::721045248511:policy/AmazonDynamoDBFullAccess
```

```
      - arn:aws:iam::721045248511:policy/AmazonSNSFullAccess
      - arn:aws:iam::721045248511:policy/CloudWatchLogsFullAccess
      - arn:aws:iam::721045248511:policy/AWSLambdaFullAccess
    Path: "/"
    RoleName: "S3ImageGetterLambdaExecutionRole"

ImageProcessorLambdaExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service:
          - lambda.amazonaws.com
        Action:
        - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::721045248511:policy/AmazonKinesisFullAccess
      - arn:aws:iam::721045248511:policy/AmazonRekognitionFullAccess
      - arn:aws:iam::721045248511:policy/AmazonS3FullAccess
      - arn:aws:iam::721045248511:policy/AmazonDynamoDBFullAccess
      - arn:aws:iam::721045248511:policy/AmazonSNSFullAccess
      - arn:aws:iam::721045248511:policy/CloudWatchLogsFullAccess
    Path: "/"
    RoleName: "ImageProcessorLambdaExecutionRole"

FrameFetcherLambdaExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Principal:
          Service:
          - lambda.amazonaws.com
        Action:
        - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::721045248511:policy/AmazonS3FullAccess
      - arn:aws:iam::721045248511:policy/AmazonDynamoDBFullAccess
      - arn:aws:iam::721045248511:policy/CloudWatchLogsFullAccess
    Path: "/"
    RoleName: "FrameFetcherLambdaExecutionRole"

FrameStream:
  Type: "AWS::Kinesis::Stream"
  Properties:
    Name: !Ref KinesisStreamNameParameter
    ShardCount: 1

ImageRekogS3Bucket:
  Type: "AWS::S3::Bucket"
  Properties:
    BucketName: !Ref ImageReKogS3BucketNameParameter
    NotificationConfiguration:
      LambdaConfigurations:
        -
          Function: !GetAtt S3ImageGetterLambda.Arn
          Event: "s3:ObjectCreated:Put"
          Filter:
            S3Key:
              Rules:
                -
                  Name: suffix
```

```
                    Value: jpg
    DependsOn:
      - ImageRekogS3BucketPermission

  ImageRekogS3BucketPermission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !Ref S3ImageGetterLambda
      Principal: s3.amazonaws.com
      SourceAccount: !Ref "AWS::AccountId"
      SourceArn: !Sub "arn:aws:s3:::${ImageReKogS3BucketNameParameter}"

  S3ImageGetterLambda:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: "s3imagegetter"
      Description: "Function Gets the Posted Image from S3 Bucket And sends to Kinesis Stream"
      Handler: "s3imagegetter.handler"
      Role: !GetAtt S3ImageGetterLambdaExecutionRole.Arn
      Code:
        S3Bucket: !Ref SourceS3BucketParameter
        S3Key: !Ref S3ImageGetterSourceS3KeyParameter
      Timeout: 40 #seconds
      MemorySize: 256 #MB
      Runtime: python2.7
    DependsOn:
      - FrameStream
      - S3ImageGetterLambdaExecutionRole

  ImageProcessorLambda:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: "imageprocessor"
      Description: "Function processes frame images fetched from a Kinesis stream."
      Handler: "imageprocessor.handler"
      Role: !GetAtt ImageProcessorLambdaExecutionRole.Arn
      Code:
        S3Bucket: !Ref SourceS3BucketParameter
        S3Key: !Ref ImageProcessorSourceS3KeyParameter
      Timeout: 40 #seconds
      MemorySize: 128 #MB
      Runtime: python2.7
    DependsOn:
      - FrameStream
      - ImageProcessorLambdaExecutionRole

  EventSourceMapping:
    Type: "AWS::Lambda::EventSourceMapping"
    Properties:
      EventSourceArn: !GetAtt FrameStream.Arn
      FunctionName: !GetAtt ImageProcessorLambda.Arn
      StartingPosition: "TRIM_HORIZON"
    DependsOn:
      - FrameStream
      - ImageProcessorLambda

  FrameFetcherLambda:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: "framefetcher"
      Description: "Function responds to a GET request by returning a list of frames up to a certa
in fetch horizon."
      Handler: "framefetcher.handler"
      Role: !GetAtt FrameFetcherLambdaExecutionRole.Arn
      Code:
        S3Bucket: !Ref SourceS3BucketParameter
```

```
        S3Key: !Ref FrameFetcherSourceS3KeyParameter
      Timeout: 10 #seconds
      MemorySize: 128 #MB
      Runtime: python2.7
    DependsOn:
      - FrameFetcherLambdaExecutionRole

  EnrichedFrameTable:
    Type: "AWS::DynamoDB::Table"
    Properties:
      TableName: !Ref DDBTableNameParameter
      KeySchema:
        - KeyType: "HASH"
          AttributeName: "frame_id"
      AttributeDefinitions:
        - AttributeName: "frame_id"
          AttributeType: "S"
        - AttributeName: "processed_timestamp"
          AttributeType: "N"
        - AttributeName: "processed_year_month"
          AttributeType: "S"
      ProvisionedThroughput:
            WriteCapacityUnits: 10
            ReadCapacityUnits: 10
      GlobalSecondaryIndexes:
        - IndexName: !Ref DDBGlobalSecondaryIndexNameParameter
          Projection:
            ProjectionType: "ALL"
          ProvisionedThroughput:
            WriteCapacityUnits: 10
            ReadCapacityUnits: 10
          KeySchema:
          - KeyType: "HASH"
            AttributeName: "processed_year_month"
          - KeyType: "RANGE"
            AttributeName: "processed_timestamp"

  # API Gateway Resources
  VidAnalyzerRestApi:
    Type: "AWS::ApiGateway::RestApi"
    Properties:
      Description: "The amazon rekognition video analyzer public API."
      Name: !Ref ApiGatewayRestApiNameParameter
    DependsOn: FrameFetcherLambda

  EnrichedFrameResource:
    Type: "AWS::ApiGateway::Resource"
    Properties:
      RestApiId: !Ref VidAnalyzerRestApi
      ParentId: !GetAtt VidAnalyzerRestApi.RootResourceId
      PathPart: !Ref FrameFetcherApiResourcePathPart

  EnrichedFrameResourceGET:
    Type: AWS::ApiGateway::Method
    Properties:
      RestApiId: !Ref VidAnalyzerRestApi
      ResourceId: !Ref EnrichedFrameResource
      ApiKeyRequired: true
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: AWS_PROXY
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${FrameFetche
rLambda.Arn}/invocations
      MethodResponses:
        - ResponseModels:
```

```
            application/json: Empty
          StatusCode: 200
          ResponseParameters:
            "method.response.header.Access-Control-Allow-Origin": true
            "method.response.header.Access-Control-Allow-Methods": true
            "method.response.header.Access-Control-Allow-Headers": true

  # Mock integration to allow Cross-Origin Resource Sharing (CORS)
  # for Web UI to invoke API Gateway
  EnrichedFrameResourceOPTIONS:
    Type: AWS::ApiGateway::Method
    Properties:
      RestApiId: !Ref VidAnalyzerRestApi
      ResourceId: !Ref EnrichedFrameResource
      ApiKeyRequired: false
      HttpMethod: OPTIONS
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        IntegrationHttpMethod: OPTIONS
        PassthroughBehavior: WHEN_NO_MATCH
        RequestTemplates:
          "application/json": '{"statusCode": 200 }'
        IntegrationResponses:
          - StatusCode: 200
            ResponseParameters:
              "method.response.header.Access-Control-Allow-Origin": "'*'"
              "method.response.header.Access-Control-Allow-Methods": "'GET,OPTIONS'"
              "method.response.header.Access-Control-Allow-Headers": "'Content-Type,X-Amz-Date,Aut
horization,X-Api-Key,X-Amz-Security-Token'"
            ResponseTemplates:
              "application/json": ''
      MethodResponses:
        - ResponseModels:
            application/json: Empty
          StatusCode: 200
          ResponseParameters:
            "method.response.header.Access-Control-Allow-Origin": true
            "method.response.header.Access-Control-Allow-Methods": true
            "method.response.header.Access-Control-Allow-Headers": true

  VidAnalyzerApiDeployment:
    Type: "AWS::ApiGateway::Deployment"
    Properties:
      Description: "Public API endpoint of video analyzer."
      RestApiId: !Ref VidAnalyzerRestApi
    DependsOn:
      - EnrichedFrameResourceGET
      - EnrichedFrameResourceOPTIONS

  DevStage:
    Type: "AWS::ApiGateway::Stage"
    Properties:
      DeploymentId: !Ref VidAnalyzerApiDeployment
      Description: "API development stage of video analyzer."
      RestApiId: !Ref VidAnalyzerRestApi
      StageName: !Ref ApiGatewayStageNameParameter

  DevUsagePlan:
    Type: AWS::ApiGateway::UsagePlan
    Properties:
      ApiStages:
      - ApiId: !Ref VidAnalyzerRestApi
        Stage: !Ref DevStage
      Description: Development usage plan
      UsagePlanName: !Ref ApiGatewayUsagePlanNameParameter
    DeletionPolicy: Retain #Had to be added to avoid stack deletion failing due to association wit
```

```
h DevStage.

  VidAnalyzerApiKey:
    Type: "AWS::ApiGateway::ApiKey"
    Properties:
      Name: "DevApiKey"
      Description: "Video Analyzer Dev API Key"
      Enabled: true
    DependsOn:
      - VidAnalyzerApiDeployment

  DevUsagePlanKey:
    Type: "AWS::ApiGateway::UsagePlanKey"
    Properties :
      KeyId: !Ref VidAnalyzerApiKey
      KeyType: API_KEY
      UsagePlanId: !Ref DevUsagePlan

  #Give API Gateway permission to invoke FrameFetcher lambda function.
  LambdaInvokePermissionSTAR:
    Type: "AWS::Lambda::Permission"
    Properties:
      FunctionName: !GetAtt FrameFetcherLambda.Arn
      Action: "lambda:InvokeFunction"
      Principal: "apigateway.amazonaws.com"
      SourceArn: !Join [ "", ["arn:aws:execute-api:", !Ref "AWS::Region", ':', !Ref "AWS::AccountI
d", ':', !Ref VidAnalyzerRestApi, '/*/*/', !Ref FrameFetcherLambdaFunctionName]]
    DependsOn:
      - VidAnalyzerApiDeployment

  LambdaInvokePermissionGET:
    Type: "AWS::Lambda::Permission"
    Properties:
      FunctionName: !GetAtt FrameFetcherLambda.Arn
      Action: "lambda:InvokeFunction"
      Principal: "apigateway.amazonaws.com"
      SourceArn: !Join [ "", ["arn:aws:execute-api:", !Ref "AWS::Region", ':', !Ref "AWS::AccountI
d", ':', !Ref VidAnalyzerRestApi, '/*/GET/', !Ref FrameFetcherApiResourcePathPart]]
    DependsOn:
      - VidAnalyzerApiDeployment

Outputs:
  #API Gateway endpoint Id
  VidAnalyzerApiEndpoint:
    Description: "Endpoint for invoking video analyzer API."
    Value: !Ref VidAnalyzerApiDeployment

  #API Key Id
  VidAnalyzerApiKey:
    Description: "Key for invoking video analyzer API."
    Value: !Ref VidAnalyzerApiKey
```

Package Lambda Functions

Run these commands from Non-Virtual Env

```
   pynt packagelambda
   Output: framefetcher.zip and imageprocessor.zip files get created under BUILD directory
   Currently, only Image Processor requires an external dependency, pytz
```

Deploy Lambda Functions

Run these commands from Non-Virtual Env

```
   pynt deploylambda # Deploy both functions to Amazon S3.

Run this command before you run createstack.
The deploylambda command uploads Image Processor and Frame Fetcher .zip packages to Amazon S3 for
pickup by AWS CloudFormation while creating the prototype's stack.
This command will parse the deployment Amazon S3 bucket name and keys names from the cfn-params.js
on file. If the bucket does not exist, the script will create it.
This bucket must be in the same AWS region as the AWS CloudFormation stack, or else the stack crea
tion will fail.
Without parameters, the command will deploy the .zip packages of both Image Processor and Frame Fe
tcher.
You can specify either “imageprocessor” or “framefetcher” as a parameter between square brackets t
o deploy an individual function.
```

Create Stack

CFStack.png

```
pynt createstack

CREATE_COMPLETE AWS::CloudFormation::Stack           video-analyzer-stack
CREATE_COMPLETE AWS::DynamoDB::Table                EnrichedFrameTable
CREATE_COMPLETE AWS::Lambda::EventSourceMapping     EventSourceMapping
CREATE_COMPLETE AWS::Lambda::Permission             LambdaInvokePermissionGET
CREATE_COMPLETE AWS::Lambda::Permission             LambdaInvokePermissionSTAR
CREATE_COMPLETE AWS::ApiGateway::UsagePlanKey       DevUsagePlanKey
CREATE_COMPLETE AWS::ApiGateway::UsagePlan          DevUsagePlan
CREATE_COMPLETE AWS::Lambda::Function               ImageProcessorLambda
CREATE_COMPLETE AWS::ApiGateway::Stage              DevStage
CREATE_COMPLETE AWS::ApiGateway::ApiKey             VidAnalyzerApiKey
CREATE_COMPLETE AWS::Kinesis::Stream                FrameStream
CREATE_COMPLETE AWS::ApiGateway::Deployment         VidAnalyzerApiDeployment
CREATE_COMPLETE AWS::ApiGateway::Method             EnrichedFrameResourceGET
CREATE_COMPLETE AWS::ApiGateway::Method             EnrichedFrameResourceOPTIONS
CREATE_COMPLETE AWS::ApiGateway::Resource           EnrichedFrameResource
CREATE_COMPLETE AWS::S3::Bucket                     FrameS3Bucket
CREATE_COMPLETE AWS::ApiGateway::RestApi            VidAnalyzerRestApi
CREATE_COMPLETE AWS::Lambda::Function               FrameFetcherLambda
CREATE_COMPLETE AWS::IAM::Role                      ImageProcessorLambdaExecutionRole
CREATE_COMPLETE AWS::IAM::Role                      FrameFetcherLambdaExecutionRole
```

The webui build command

```
pynt webui

1. Run this command when the prototype's stack has been created (using createstack).
2. The webui command “builds” the Web UI through which you can monitor incoming captured video fra
mes.
   First, the script copies the webui/ directory verbatim into the project’s build/ directory.
   Next, the script generates an apigw.js file which contains the API Gateway base URL and the API
 key to be used by Web UI for invoking the Fetch Frames function deployed in AWS Lambda.
   This file is created in the Web UI build directory.

You can issue the Web UI build command as follows.
```

The webuiserver build command

```
pynt webuiserver # Starts lightweight HTTP Server on port 8080.

1. The webuiserver command starts a local, lightweight, Python-based HTTP server on your machine t
o serve Web UI from the build/web-ui/ directory.
2. Use this command to serve the prototype's Web UI for development and demonstration purposes.
3. You can specify the server's port as pynt task parameter, between square brackets.

Here's sample invocation of the command.
```

How to Run

```
cd ~/projects/deep-learning
1. Package and Deploy Lambda Functions
   pynt packagelambda
   pynt deploylambda

2. Create Video Analyzer Stack
   pynt createstack

3. Write the API key and API base URL to apigw.js
   pynt webui

4. Launch Python based web server
   pynt webuiserver
```

Image Rekognition

How To

Capture Video Frame from Kinesis Stream

def process_image(event, context): #Initialize clients
rekog_client = boto3.client('rekognition')
sns_client = boto3.client('sns')
s3_client = boto3.client('s3')
dynamodb = boto3.resource('dynamodb') #Iterate on frames fetched from Kinesis
for record in event['Records']: frame_package_b64 = record['kinesis']['data']
frame_package = cPickle.loads(base64.b64decode(frame_package_b64)) img_bytes = frame_package["ImageBytes"]

Sample Video Frames Capture

Human

upender_screen_shot.png

ImaheRekognitionDetectFace.png

Object

cups.png

DynamoDB

Item

dynamodb_item.png

Pricing

Image Analysis Tiers	Price per 1,000 Images Processed
First 1 million images processed* per month	$1.00
Next 9 million images processed* per month	$0.80
Next 90 million images processed* per month	$0.60
Over 100 million images processed* per month	$0.40

Face Metadata Storage	Price per 1,000 face metadata stored per month
Face metadata stored	$0.01

For Example:

> Pricing Example 1 for US East (when outside the free tier)
> An application that analyzes 1 million images per month that require label detection.
> You would use Amazon Rekognition's DetectLabels APIs to analyze these 1 million images.

Number of images processsed	Price per image up to 1M images	Price per 1,000 images	Total per month
1 Million	$0.001	$1.00	$1,000.00

Limits

The following is a list of limits in Amazon Rekognition:

1. Maximum image size stored as an Amazon S3 object is limited to 15 MB. The minimum pixel resolution for height and width is 80 pixels.
2. Maximum images size as raw bytes passed in as parameter to an API is 5 MB.
3. Amazon Rekognition supports the PNG and JPEG image formats. That is, the images you provide as input to various API operations, such as DetectLabels and IndexFaces must be in one of the supported formats.
4. Maximum number of faces you can store in a single face collection is 1 million.
5. The maximum matching faces the search API returns is 4096.

Amazon Kinesis Video Streaming

```
Amazon Kinesis Video Streams makes it easy to securely stream video from connected devices to AWS
for analytics, machine learning (ML),
and other processing. Kinesis Video Streams automatically provisions and elastically scales all th
e infrastructure needed to ingest streaming video data
from millions of devices. It also durably stores, encrypts, and indexes video data in your streams
, and allows you to access your data through
easy-to-use APIs. Kinesis Video Streams enables you to quickly build computer vision and ML applic
ations through integration with Amazon Rekognition Video
and libraries for ML frameworks such as Apache MxNet, TensorFlow, and OpenCV.
```

Amazon Kinesis Video Streams Producer SDK Java

```
The Amazon Kinesis Video Streams Producer SDK Java makes it easy to build an on-device application
 that securely connects to a video stream, and reliably
publishes video and other media data to Kinesis Video Streams. It takes care of all the underlying
 tasks required to package the frames and fragments
```

```
generated by the device's media pipeline. The SDK also handles stream creation, token rotation for
 secure and uninterrupted streaming, processing acknowledgements returned by Kinesis Video Streams
, and other tasks.
```

How to Run Kinesis Producer and Consumer

Producer

```
1. Download jar from s3://sage-media-bucket/jars/kinesisVideoProducer.jar

2. Download demo/mkv/clusters.mkv from sage-media-bucket on S3.

3. Create new Kinesis Video Stream as "my-mkv-stream"

4. Run producer as following to send movie "demo/mkv/clusters.mkv" to "my-mkv-stream" Kinesis Vide
o Stream
   export JARS=<path to jars>
   java -Dstream.name="my-mkv-stream" \
        -Ds3.bucket="sage-media-bucket" \
        -Ds3.key="demo/mkv/clusters.mkv" \
        -Ds3.download.dir="/tmp/mymovies" \
        -jar $JARS/kinesisVideoProducer.jar

    Or (in Single Line)

java -Dstream.name="my-mkv-stream" -Ds3.bucket="sage-media-bucket" -Ds3.key="demo/mkv/clusters.mkv
"  -Ds3.download.dir="/tmp/mymovies" -jar $JARS/kinesisVideoProducer.jar
```

Consumer

```
  1. Kinesis Consumer application fetches frames off of the Kinesis Video Stream, and sends them t
o rekog-images-bucket.
  2. This triggers an S3 event Lambda function which packages the frame with image bytes and some
other metadata.
  3. This is then sent to Kinesis Data stream "Framestream".
  4. A new Lambda function named "imageprocessor" gets triggered which picks up the frame, unpickl
es it and send to Image Rekognition.
```

```
  export JARS=<path to jars>
  java -Dstream.name="my-mkv-stream" -jar $JARS/kinesisVideoConsumerToS3Sender.jar
```

Troubleshooting

Error: There are no faces in the Image

Description:

```
n error occurred (InvalidParameterException) when calling the SearchFacesByImage operation: There
are no faces in the image. Should be at least 1.: InvalidParameterException
Traceback (most recent call last):
File "/var/task/imageprocessor.py", line 209, in handler
```

```
return process_image(event, context)
File "/var/task/imageprocessor.py", line 110, in process_image
person_found = person_of_interest_finder(rekog_client, img_bytes, config)
File "/var/task/imageprocessor.py", line 188, in person_of_interest_finder
FaceMatchThreshold=config['face_match_threshold']
File "/var/runtime/botocore/client.py", line 314, in _api_call
return self._make_api_call(operation_name, kwargs)
File "/var/runtime/botocore/client.py", line 612, in _make_api_call
raise error_class(parsed_response, operation_name)
InvalidParameterException: An error occurred (InvalidParameterException) when calling the SearchFa
cesByImage operation: There are no faces in the image. Should be at least 1.
```

Solution:

Sample Commands

DynamoDB

Get the Items Count

```
aws dynamodb scan --table-name EnrichedFrame --select "COUNT"
```

Delete All Entries from a Table

```
export KEY="frame_id"
export TABLE="EnrichedFrame"

aws dynamodb scan --table-name "$TABLE" --attributes-to-get "$KEY" \
--query "Items[].$KEY.S" --output text | \
tr "\t" "\n" | \
xargs -t -I keyvalue aws dynamodb delete-item --table-name "$TABLE" \
--key "{\"$KEY\": {\"S\": \"keyvalue\"}}"
```

Further Reads

FFMPEG

http://ffmpeg.org/ffmpeg.html.
ffmpeg is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter. We require this to be able to capture AUDIO with VIDEO Frames and store it in S3 bucket.

Install nasm2.0.13

```
1. wget http://www.nasm.us/pub/nasm/releasebuilds/2.13.01/nasm-2.13.01.tar.gz
2. tar xzvf nasm-2.13.01.tar.gz
3. cd nasm-2.13.01
4. ./configure --prefix=/opt/nasm
5. make
6. sudo make install
7. export PATH=/opt/nasm/bin/:$PATH
```

Install x264 Codec

```
1. git clone http://git.videolan.org/git/x264.git
```

```
  2. cd x264
  2. ./configure --enable-shared
  3. make
  4. sudo make install
```

Install ffmpeg

```
   1. sudo apt-get install nasm yasm libvpx. libx264.
   2. git clone git@github.com:FFmpeg/FFmpeg.git
   3. ./configure --enable-shared --enable-libx264 --enable-gpl --extra-cflags="-fPIC"

      _Note: If you get error "libavcodec/mqc.o: error adding symbols: Bad value", then do_
      make distclean
      Run step 3 again.
   4. make
   5. sudo make install
   6. edit /etc/ls.so.conf and add the following line in the end:
      /usr/local/lib/
   7. run --> sudo ldconfig
```

ffmpeg Sample Commands

Get the JPG Frames from MP4

```
ffmpeg -ss 00:00:25 -t 00:00:00.04 -i YOURMOVIE.MP4 -r 25.0 YOURIMAGE%4d.jpg
  -- will extract frames beginning at second 25 [-ss 00:00:25] stopping after 0.04 second [-t 00:0
0:00.04] reading from input file YOURMOVIE.MP4
  -- using only 25.0 frames per second, i. e. one frame every 1/25 seconds [-r 25.0]
  -- as JPEG images with the names YOURIMAGE%04d.jpg, where %4d is a 4-digit autoincrement number
with leading zeros

  example: ffmpeg -ss 00:00:01 -t 00:00:00.1 -i jellyfish-25-mbps-hd-hevc.mp4 -r 60.0 frames/jelly
-%4d.jp

            Or use the following if above command throws some timing errors etc.

           ffmpeg -i mp4/jellyfish-25-mbps-hd-hevc.mp4 -r 25.0 jellyfish/jelly-%4d.jpg
```

Convert MP4 to MKV

```
  1. ffmpeg -i input.mp4 -b:v 10M -minrate 10M -maxrate 10M -bufsize 10M -bf 0 input.mkv

  2. Lossless
     ffmpeg \
     -i input.mp4 \
     -vcodec copy \
     -acodec copy \
     output.mkv
```

Remove Audio Track from MP4 and convert to MKV

```
ffmpeg -i <Path to Input Movie> -map 0 -map -0:a:0 -c copy <Output Path>

For example
ffmpeg -i /home/upen/aws/s3/sage-media-bucket/demo/mp4/BigBuckBunny_320x180.mp4 -map 0 -map -0:a:0
 -c copy /tmp/test.mkv
```

Remove Audio Track from MP4 and convert to MKV using vcodec

```
ffmpeg -i <Path to Input Movie> -vcodec libx264 -map 0 -map -0:a:0 -c copy <Output Path>

For example
ffmpeg -i /data/1/aws/sage-media-bucket/demo/mp4/BigBuckBunny_320x180.mp4 -vcodec libx264 -map 0 -
map -0:a:0 -c copy /tmp/BigBunny2.mkv
```