

Software

# Visions of clouds: Intel® and AWS\*

Martin Kronberg – Technical Evangelist

# Outline

- Intro
- Intel IoT Overview
- AWS Overview
- CV at the Edge
- CV in the Cloud
- Demo!

# Computer Vision– Edge



- Basic Detection
- Programmatic
- Fast

# Computer Vision– Cloud



- Metadata
- Machine Learning
- Heavy Resource Requirements
- Slower
- Less fine control

# Put it Together!



# Intel IoT Overview

# Intel® Architecture Powers

## Endpoints, Gateways, Networks, and Cloud

### THINGS (ENDPOINTS)



### GATEWAY



### NETWORK (DATA CENTER)



### CLOUD-BASED ANALYTICS



The Intel® IoT Platform: A blueprint for connecting devices into the cloud for developers to better leverage data, customize, and scale

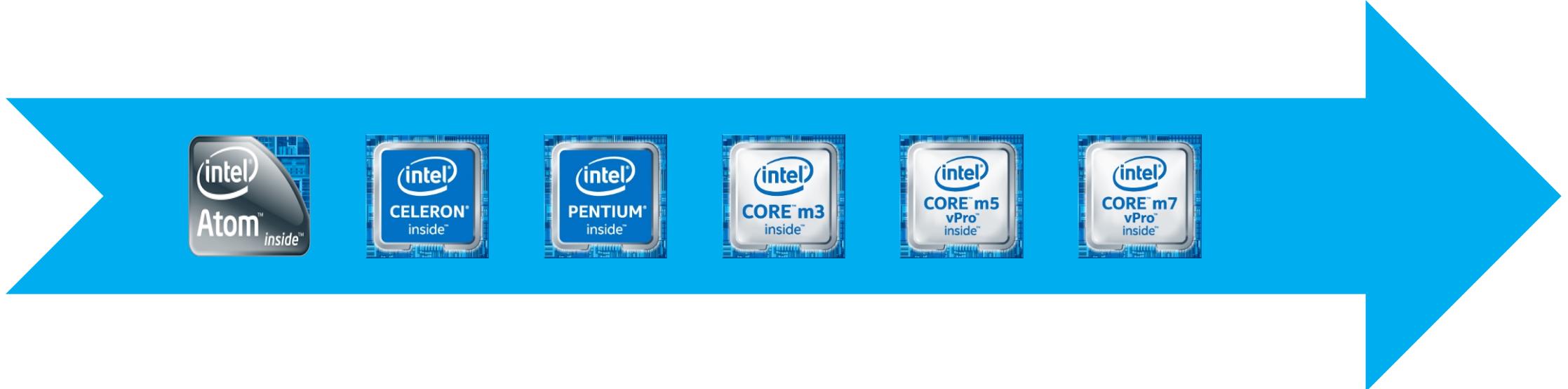
# Intel® NUC Gateway

- Scalable
- Open Source
- Wide Range of Specifications

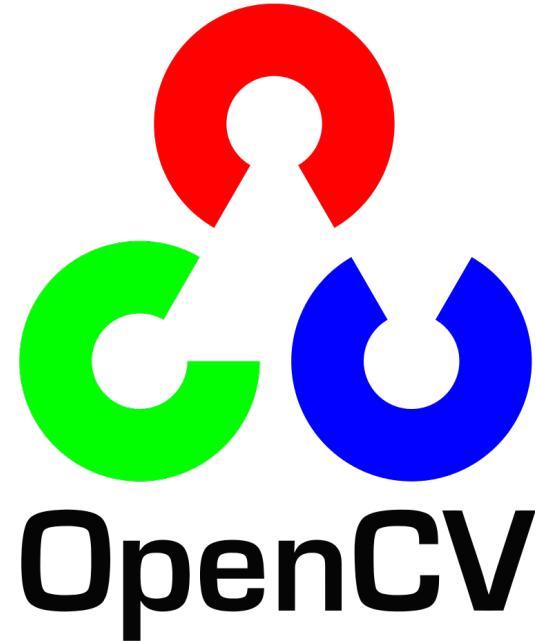


# Scalable Processing

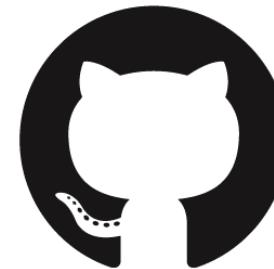
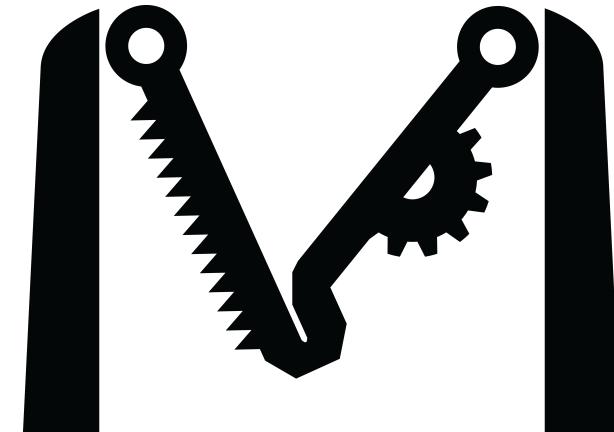
- From Intel® Atom® to Intel® Core™ i7
- Get just the right amount of compute power at the edge



# Open Source Technology



ubuntu



GitHub

# MRAA – I/O Library

Supports Intel, non-Intel (community added) MCU boards, UNIX boards and IoT Gateways

Simple I/O protocol control for:

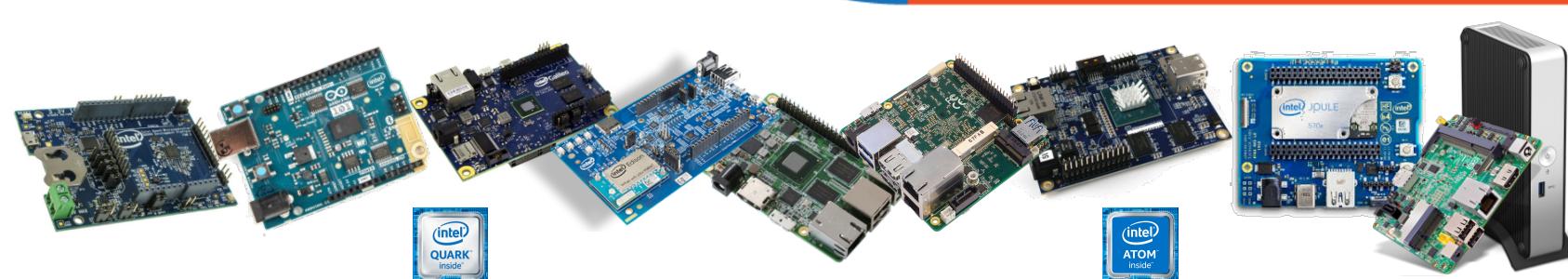
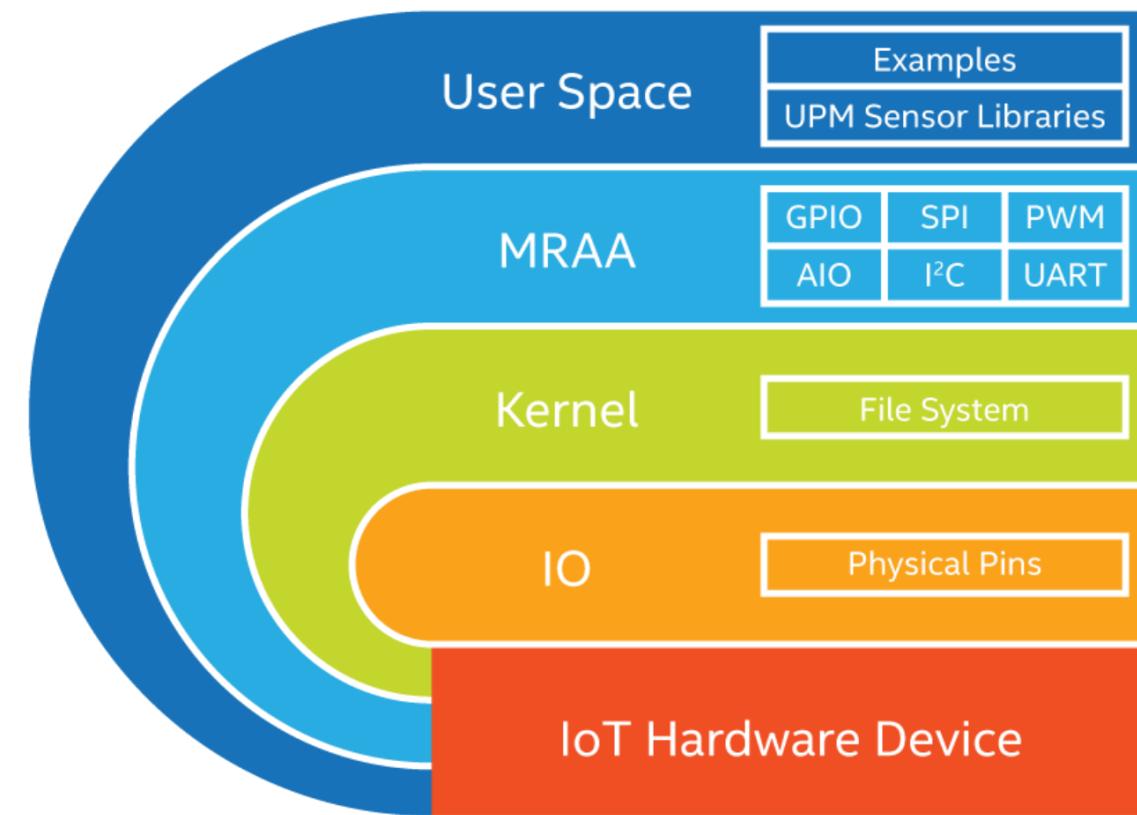
- GPIO
- Analog (AIO)
- PWM
- SPI
- I<sup>2</sup>C
- UART
- 1-Wire
- Firmata

Open source on GitHub:



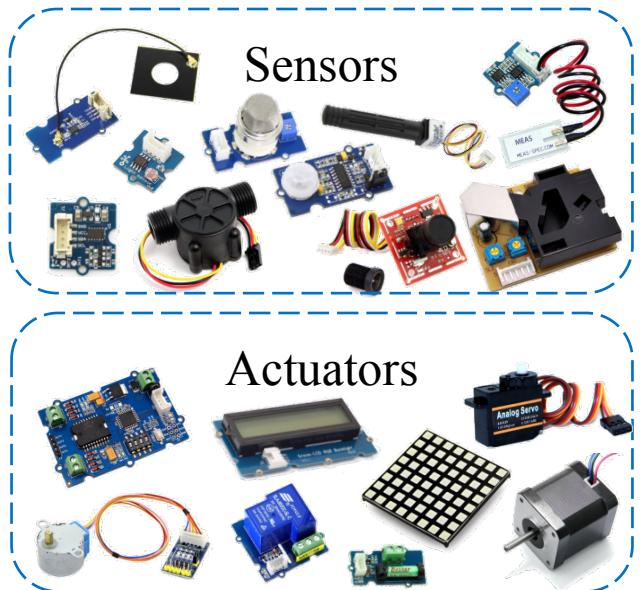
<http://mraa.io>

Typical stack on UNIX systems:



# UPM – Sensor Library

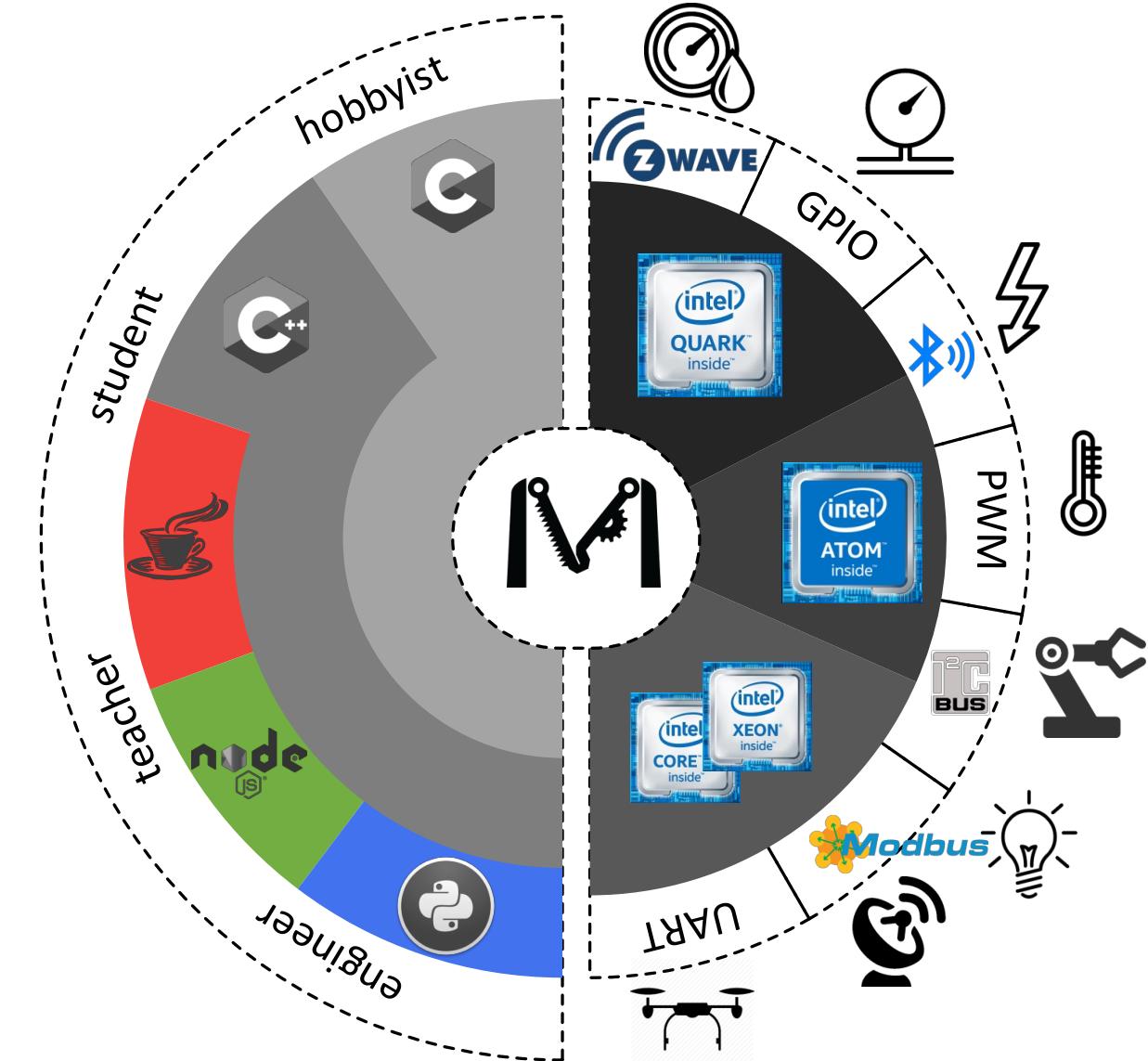
Software suite for sensors, actuators, radios and protocols



Open source, Intel maintained, community supported:



<http://upm.mraa.io>



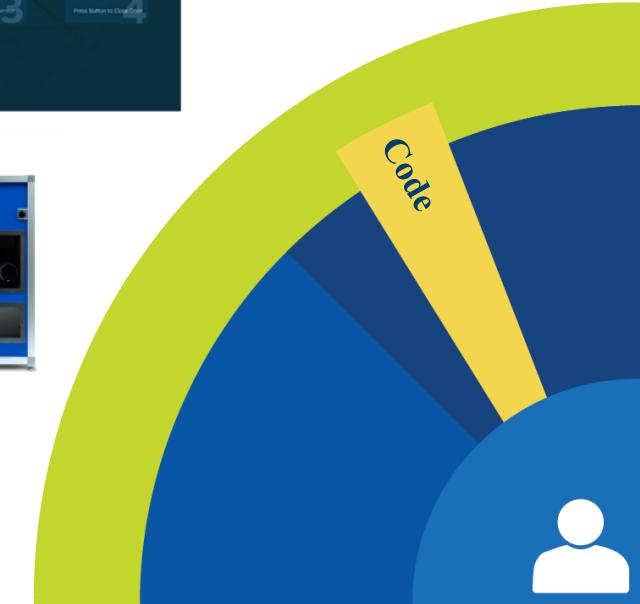
# Code Samples

Write your own or reuse sample code & packages for your own projects.

- Code snippets
- Mini projects
- End to end solutions

## End-to-end Reference Implementations

- Transportation
- Smart Home
- Retail



# Intel Developer Program: Software.intel.com/iot

The screenshot shows the homepage of the Intel Developer Zone for IoT. The top navigation bar includes the Intel Software logo, 'Developer Zone' text, a search bar ('Search our content library...'), a support link ('Support'), a profile icon ('Profile'), and a language selection ('English'). On the left, there's a sidebar with a menu icon and 'IOT Home' text. The main content area features a large image of a man in glasses working on a circuit board. Overlaid on the image is the text 'DEVELOPMENT AND DEPLOYMENT RESOURCES' in large white letters. Below this, three blue buttons offer 'Get Started', 'Digging Deeper', and 'Going to Production'. At the bottom, there's a search bar with the placeholder 'Enter your hardware, IDE, or topic:'.

# AWS Overview

# AWS\* - On Demand Cloud Computing

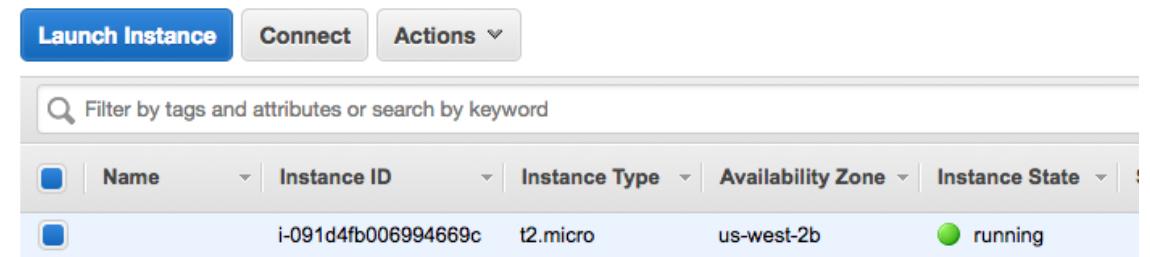
- Virtual Machines
- Compute
- Database
- Analytics
- AI
- IoT



# AWS EC2

Quickly deploy preconfigured or custom VM

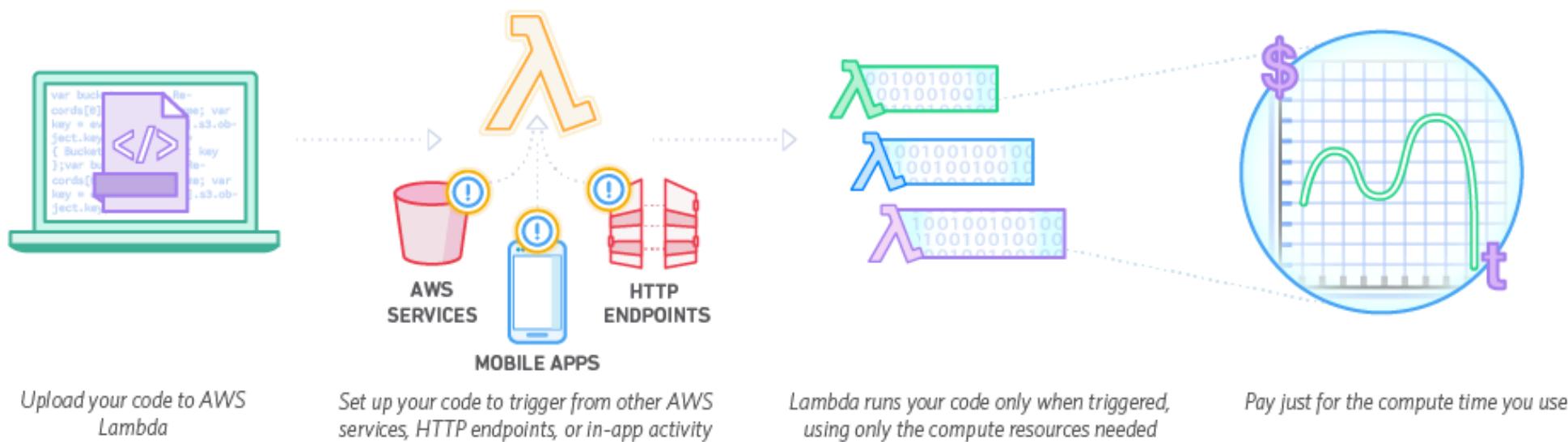
Scale with load



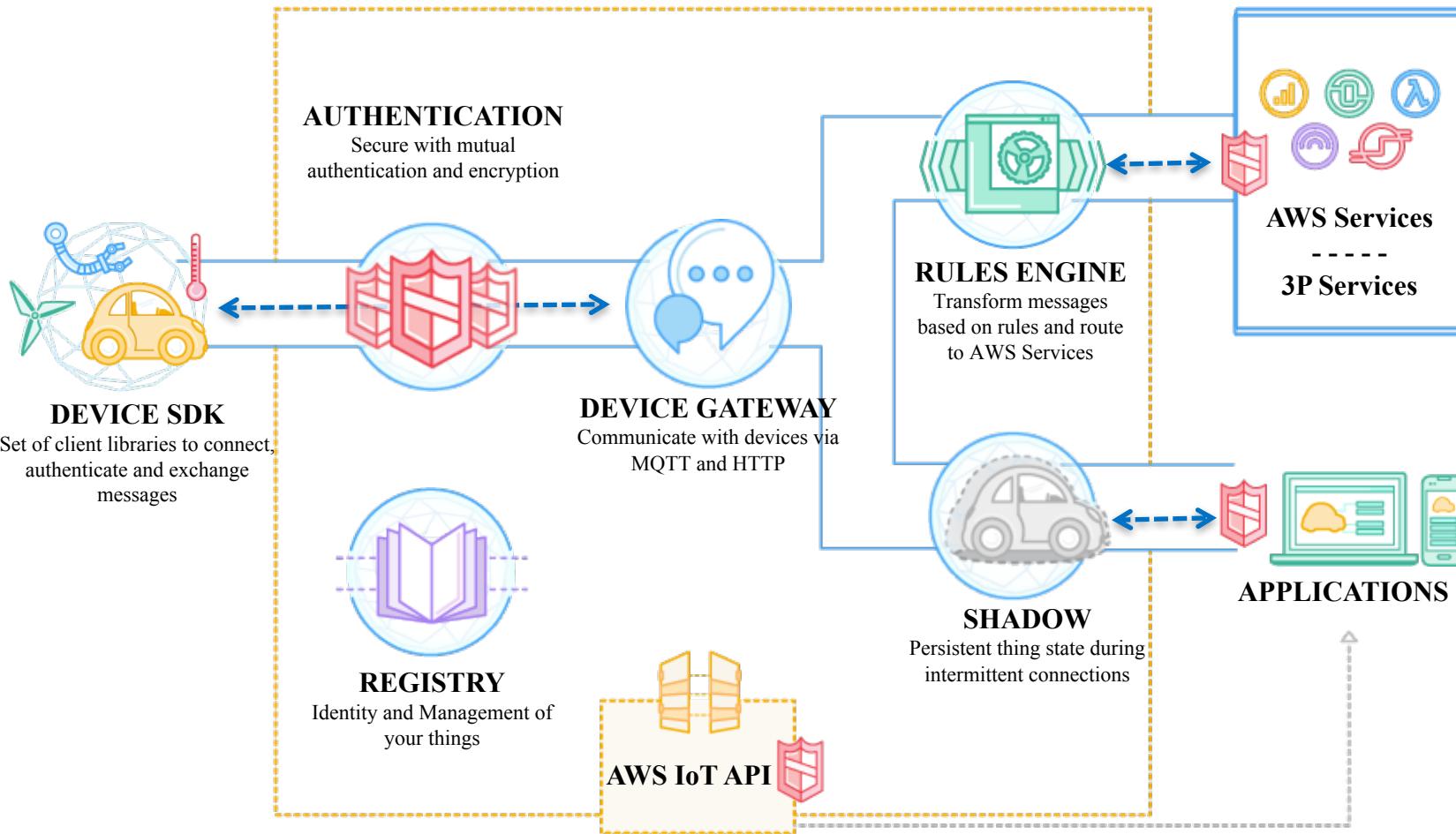
```
mkrone -- bash -- 80x45
inet 127.0.0.1 netmask 0xffffffff
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=1<PERFORMNUD>
    gif0: flags=8010<NOFORNINT,MULTICAST> mtu 1280
    stf0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1280
    en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 6c:40:08:b4:43:32
        inet6 fe80::6e40:8ff:fe04:4332%en0 prefixlen 64 scopeid 0x4
        inet 10.110.106.159 netmask 0xffffffff broadcast 10.110.127.255
        inet6 2620::1000:fd86:6e40:ffff:feb4:4332 prefixlen 64 autoconf
        inet6 2620::1000:fd86:1470:4229:7e92:bbc3 prefixlen 64 autoconf temporary
            nd6 options=1<PERFORMNUD>
            media: autoselect
            status: active
    en1: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
        options=60<TSO4,TSQ6>
        ether 72:00:06:d5:2:f:50
        media: autoselect <full-duplex>
        status: inactive
    en2: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
        options=60<TSQ6>
        ether 72:00:06:d5:2:f:51
        media: autoselect <full-duplex>
        status: inactive
    p2p0: flags=8043<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
        ether 0e:40:08:b4:43:32
        media: autoselect
        status: inactive
    awdl0: flags=1<NOFORNINT> mtu 1484
        ether 8a:dc:5e:83:00:15
        inet6 fe80::80dc:5eff:fe83:cc15%awdl0 prefixlen 64 scopeid 0x8
            nd6 options=1<PERFORMNUD>
            media: autoselect
            status: active
    bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=63<RXCSUM,TXCSUM,TSQ4,TSQ6>
        ether 6e:40:08:4b:06:00
        Configuration:
            id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
            maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
            root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
            ipfilter disabled flags 0x2
        member: en1 flags=3<LEARNING,DISCOVER>
            ifmaxaddr 0 port 5 priority 0 path cost 0
```

# AWS\* Lambda

Serverless compute in the cloud



# AWS IoT



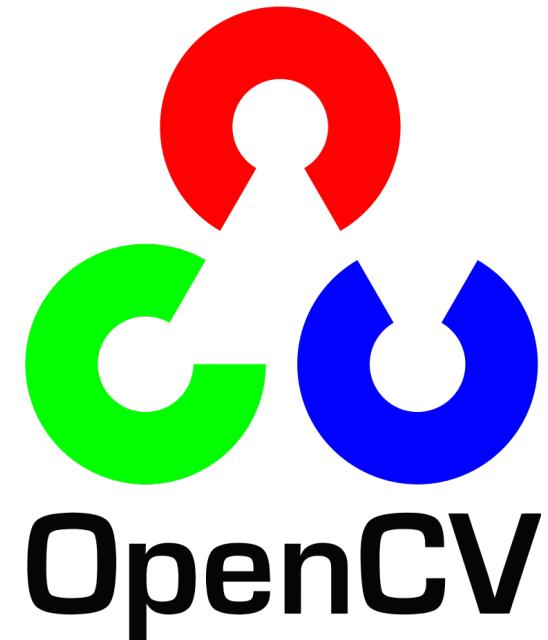
# Computer Vision at the edge – Open CV

# Overview

- Open Source Computer Vision C++ Framework
- Created by Intel in 1999
- Large and Active Developer Community

<https://www.learnopencv.com/install-opencv3-on-ubuntu/>

[http://docs.opencv.org/master/d7/d9f/tutorial\\_linux\\_install.html](http://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html)



# Open CV Uses



- Image Transformation
- Edge Detection
- Facial Recognition
- Gesture Recognition
- Mobile Robotics
- Motion Tracking

# Sample Code – Find Corner

```
import cv2
import numpy as np

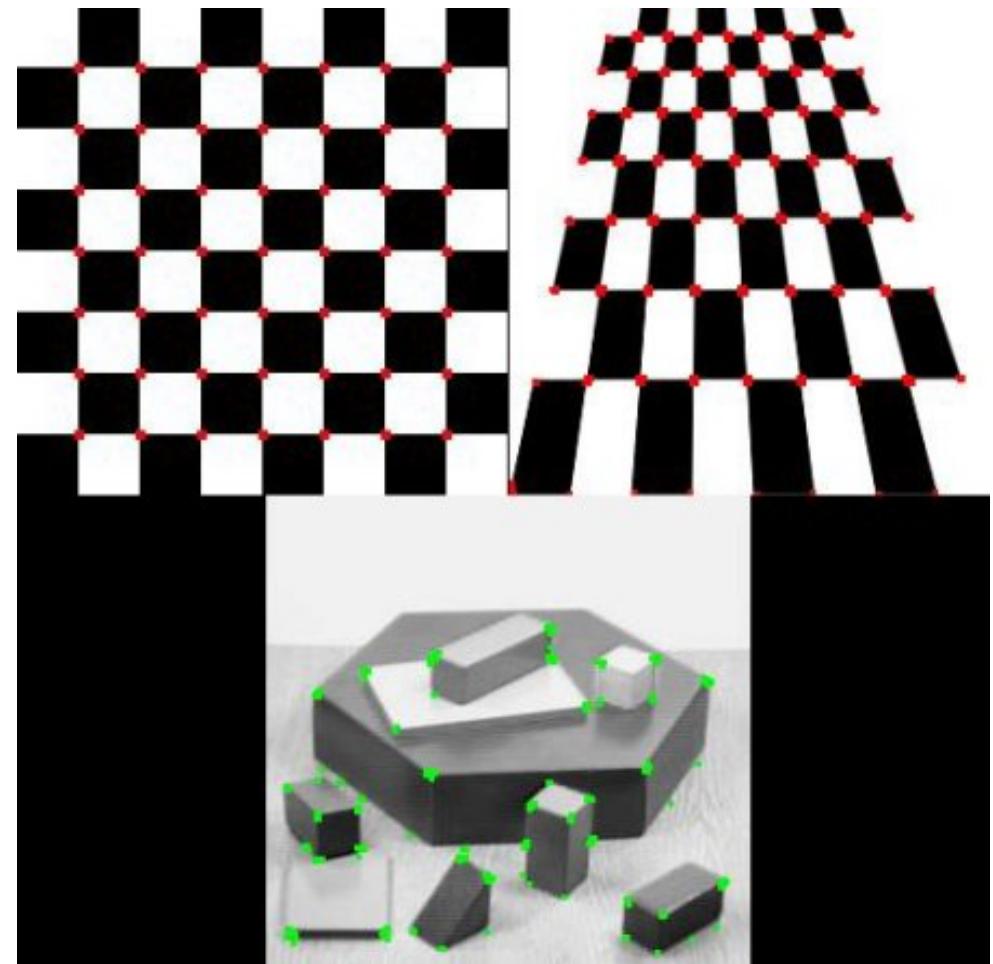
filename = 'chessboard.jpg'
img = cv2.imread(filename)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

gray = np.float32(gray)
dst = cv2.cornerHarris(gray, 2, 3, 0.04)

# result is dilated for marking the corners, not important
dst = cv2.dilate(dst, None)

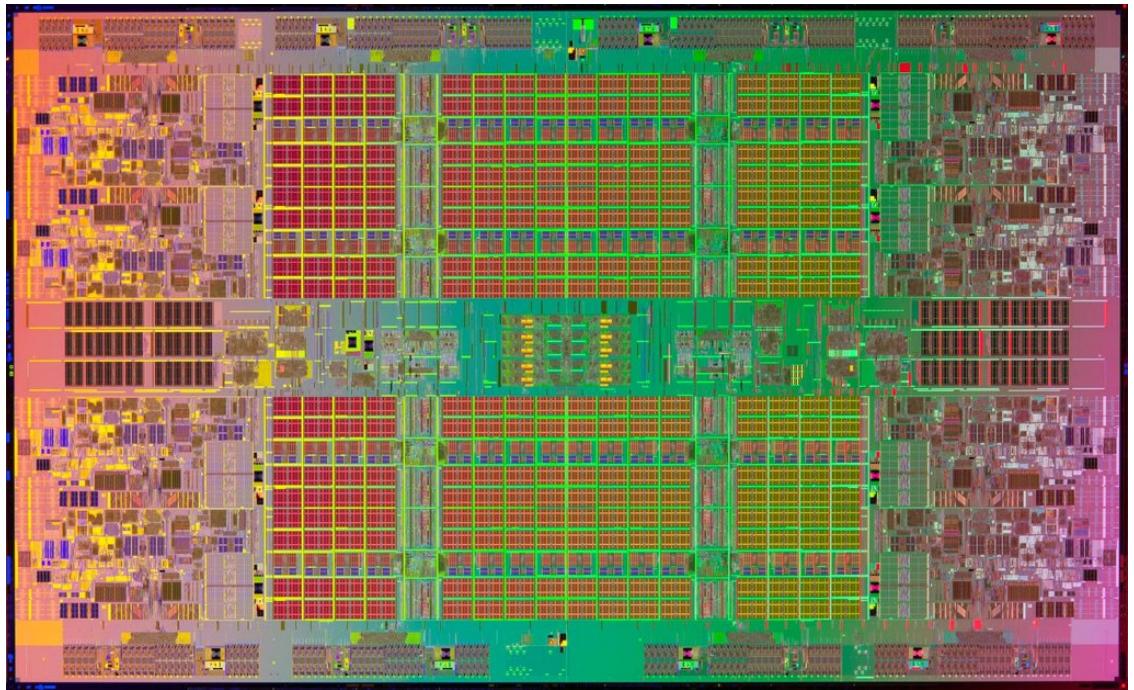
# Threshold for an optimal value, it may vary depending on the image.
img[dst > 0.01 * dst.max()] = [0, 0, 255]

cv2.imshow('dst', img)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```



# Hardware Requirements

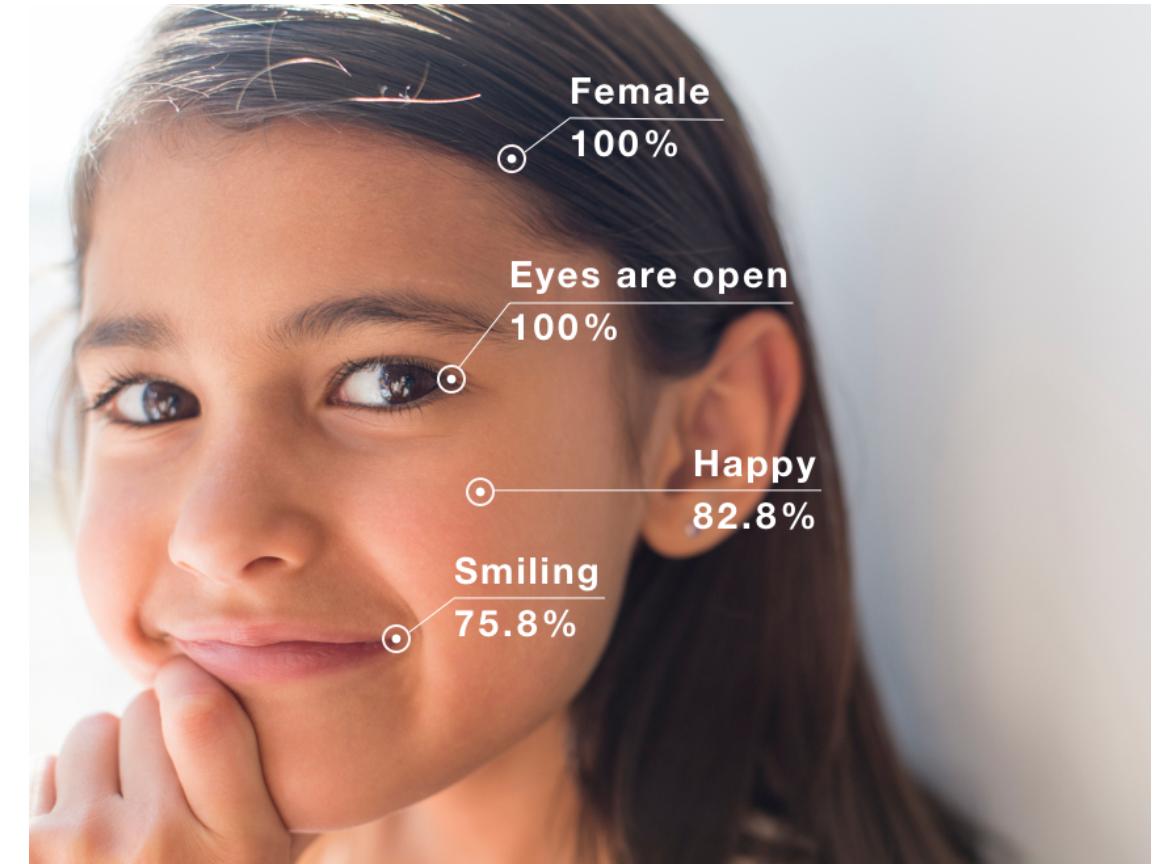
- Open CV can be CPU intensive
- Open CV can be RAM instensive



# Computer Vision in the Cloud – Rekognition

# Overview

- AWS image analysis service
- Introduced Q4 2016
- Uses deep learning algorithm



**ANALYSIS:**  
\*\*\*\*\*

234654 453 3  
654334 450 16  
245261 865 26  
453665 765 46  
382856 863 09  
  
356878 544 04  
664217 985 89  
254346 956 32

**MATCH** ■

**SCAN MODE 43894**  
**SIZE ASSESSMENT**

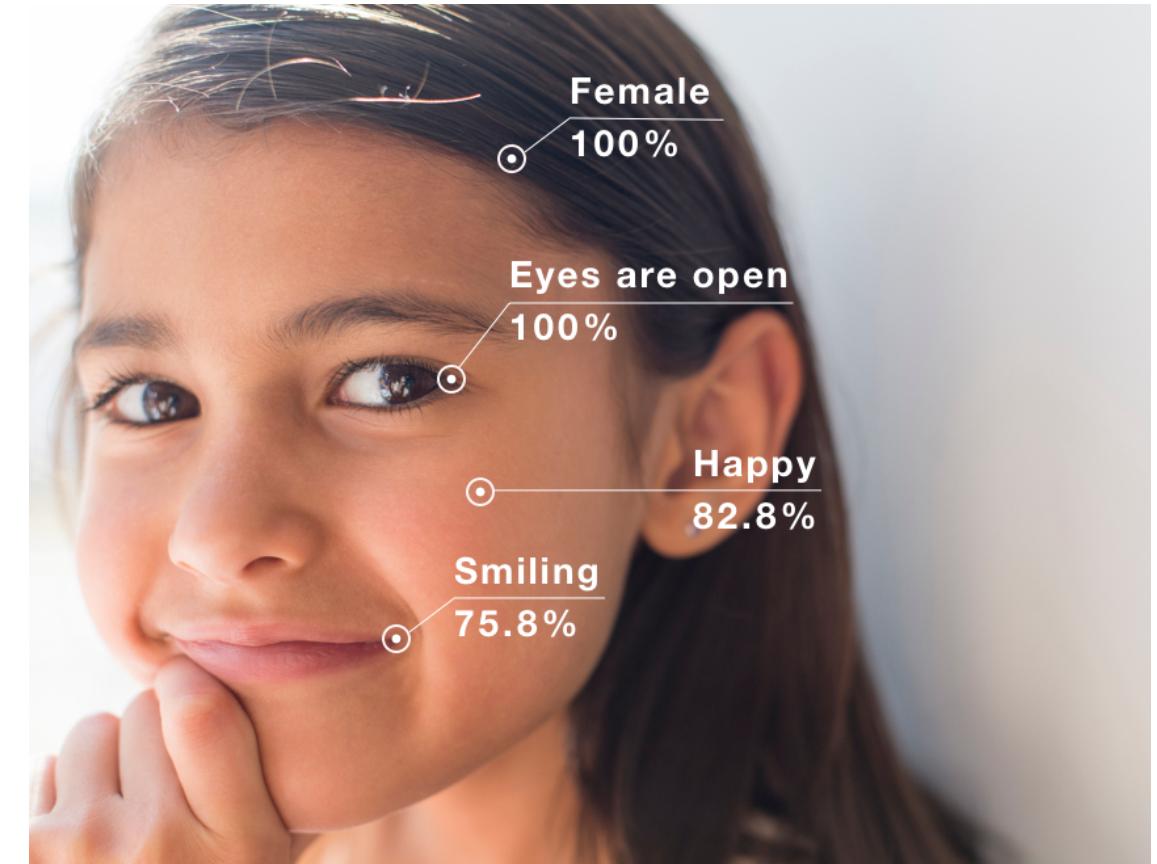
**ASSESSMENT COMPLETE**  
**FIT PROBABILITY 0.99**

**RESET TO ACQUISITION**  
**MODE SPEECH LEVEL 78**

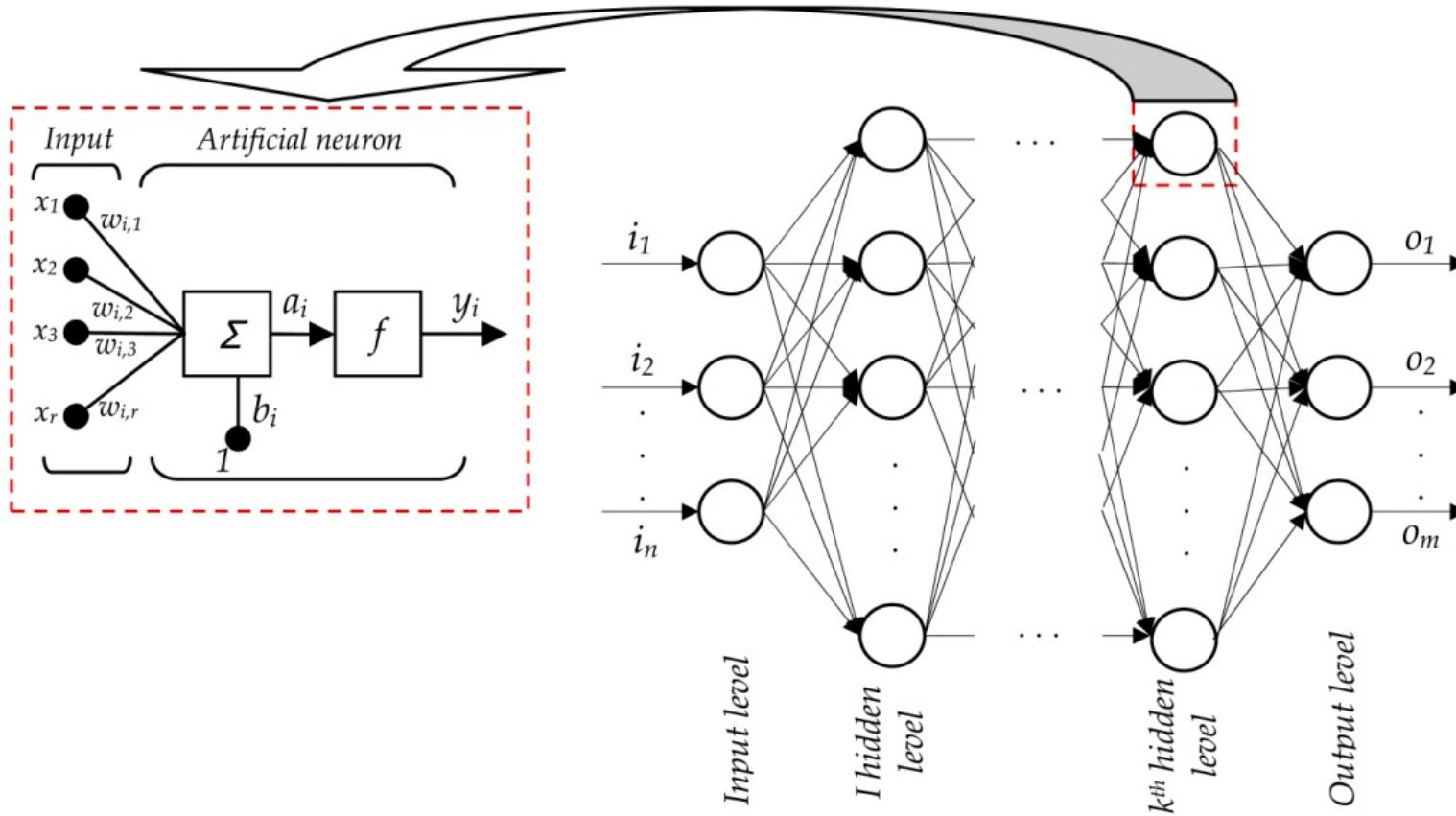
**PRIORITY OVERRIDE**  
**DEFENSE SYSTEMS SET**  
**ACTIVE STATUS**  
**LEVEL 23479 23 MAX**

# Overview

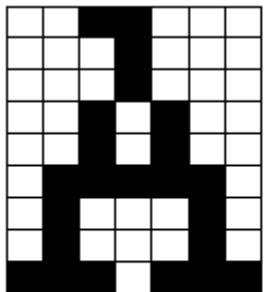
- AWS image analysis service
- Introduced Q4 2016
- Uses deep learning algorithm



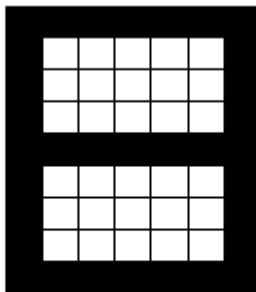
# Deep Learning / Neural Networks



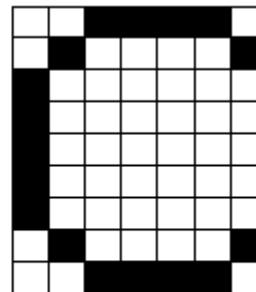
# ABCs



The Letter 'A'



The Letter 'B'



The Letter 'C'

- 7x9 pixel images of A B or C
- 63 input nodes – 1 for black pixel 0 for white pixel
- 3 output nodes – A, B, or C

# Training?

Math – Basically the chain rule with partial derivatives

**Summary: the equations of backpropagation**

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (\text{BP1})$$

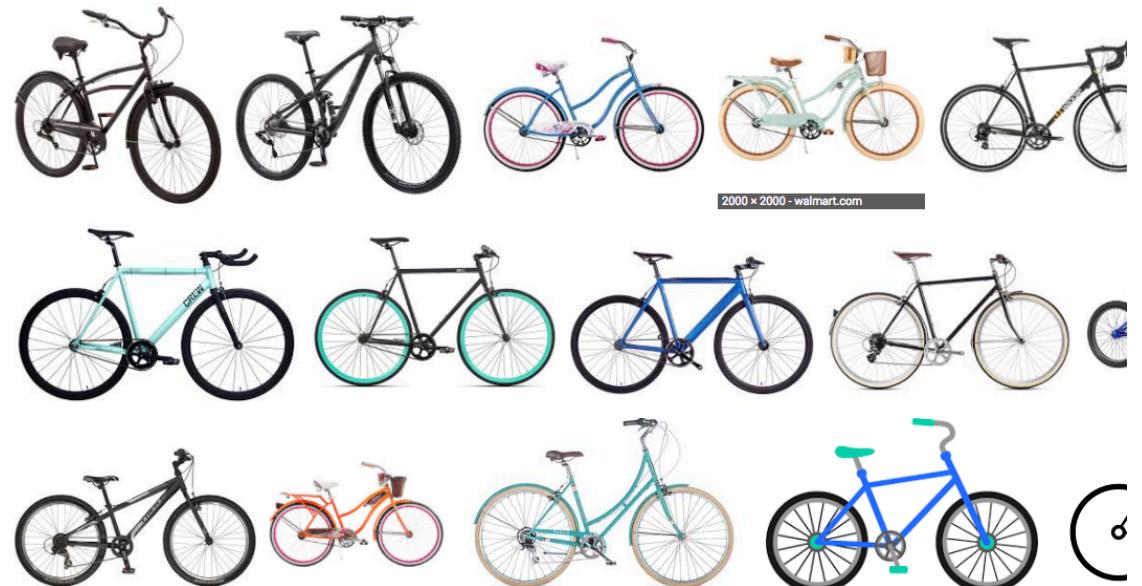
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad (\text{BP2})$$

$$\frac{\partial C}{\partial b_j^T} = \delta_j^l \quad (\text{BP3})$$

$$\frac{\partial C}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l \quad (\text{BP4})$$

# Trained Data

- To be successful a neural network needs a massive amount of *trained* data
- Trained Data = Image + Label
- ~10,000 images to train a network



“BIKE”

# Result



## ▼ Results

Human	99.2%
People	99.2%
Person	99.2%
Bicycle	97.6%
Bike	97.6%
Vehicle	97.6%
Plaid	59.3%

[Show less](#)

## ► Request

## ► Response

# Edge + Cloud demo!



Take Video



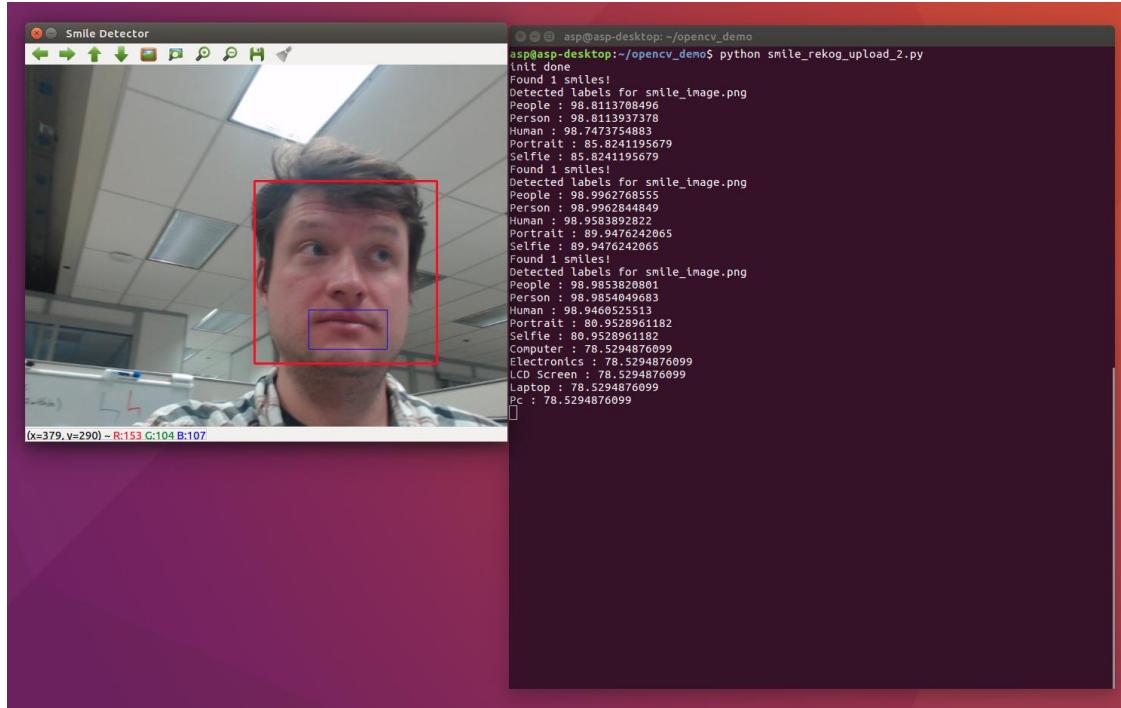
- Track Face
- Detect Smile
- Connect to AWS



### S3 + Rekognition

- Store Image
- Return Analysis

# Results!



- OpenCV can track faces in real time
- Rekognition can report back with what it thinks is in the image

# Questions?