AI in libraries on $5 a day: image matching system with Koha: a case study

Dr Edmund Balnaves

Prosentient Systems

ebalnaves@prosentient.com.au

Abstract

This article takes the reader on a journey through AI integration with a well know library management system, Koha. By leveraging open source and cloud services, an image AI integration is demonstrated on a budget of less than $5/day. This article explores the strengths and opportunities that lie in open-source AI projects with an example of integrating these services with a widely implemented open-source library management system.

Introduction

Image matching as great potential value for a wide range of library applications, from patron authentication to machine learning.

Many AI toolkits have their origins in open source. Python has been a development language of choice in this area, and tool kits for textual and image recognition about. **Examples include SCIKit, ….**

An open-source implementation of AI tools has a range of privacy benefits to the library. Image and digital content are not sent to external services where the national hosting and privacy of this content may be uncertain. Once of the ethical considerations in the adoption of AI is the degree to which the algorithmic elements of the solution used can be scrutinised, tested and understood. In the case of open source, the library has agency over the algorithmic design of the AI implementation.

In this case study we explore the process of integrating an open-source AI toolkit with an open-source library management system. In the process with explore the use of cloud services in order to provide the proof-of-concept platform for the whole solution at low cost.

Choosing the platform and toolkit

Cloud platforms provide an environment that allow for low-cost proof of concept design of system without having to invest in infrastructure in the first instance. In this case study we will illustrate the use of cloud infrastructure in Amazon Web services to implement the proof-of-concept integration demonstrated in this article, with installation of a Koha library management system and the AI tool kits, with use of Koha plugings to bring the two solutions together.

The final implementation of the solution can be within the normal infrastructure of the institution. There are several advantages to implementing the initial trial solution within a cloud infrastructure. This allows the formulation of the final technical architecture required without having to commit to this architecture during the design stages. The solution sets can therefore be created and "torn down" quite easily while the project is in evolution. The cloud infrastructure also allows experimentation with solution options without large upfront investment. The security architecture around the solution can also be safely tested outside the target implementation environment.

The use of cloud platforms for initial proof of concept testing has many benefits. During the initial exploratory stage with software experimentation the final target hardware environment may be unknown as different packages are tested for their suitability. In this case we explored several open-source image recognition packages before we arrived at the CodeProject AI instance. We settled on the CodeProject AI because:

- It installed very easily
- It passed initial image testing well
- It provided a REST Application Programming Interface for integration with other applications. This was the key requirement for our cross-platform integration.

Image recognition toolkit.

Docker is a platform that allows lightweight installation of software based on "containers" that are pre-baked with the solution you are installing. The nicety of docker is it is platform-agnostic. You can install Linux docker images on Windows systems for instance, or Windows docker images on Linux systems.

In our business we use docker images to rapidly run in and test solutions. We also use docker images for some production solutions (when thoroughly tested).

Docker lives "on top" of the host platform you are running it on, and contains within it sufficient of the operating system environment that your application uses in order to run the application. Because it is so "lightweight" it is very easy to deploy.

Docker provides an excellent method for undertaking software "throw away" trials in order to test the suitability of software, and also for production deployment of software in scalable solutions.

For the purpose of this proof of concept we were looking for an image recognition toolkit that had a good Application Programming Interface and a web interface for testing purposes, and facial recognition features. Fortunately, in the AI space, there are many open-source solution in both machine learning and image recognition.

We explored this project with a recent kit released by CodeProject. CodeProject release a range of solution kits for developers to experiment with new technologies. One of the recent solution toolkits was CodeProject AI, an implementation of Python open-source image recognition technologies within a

windows or Linux environment. The CodeProject AI solution has a docker install. There are many others – e.g. https://github.com/YashNita/FaceNet-Object-Detection-Net- and projects like https://realpython.com/face-recognition-with-python/ .

The deployment of the CodeProject AI server was accomplished with a single command:

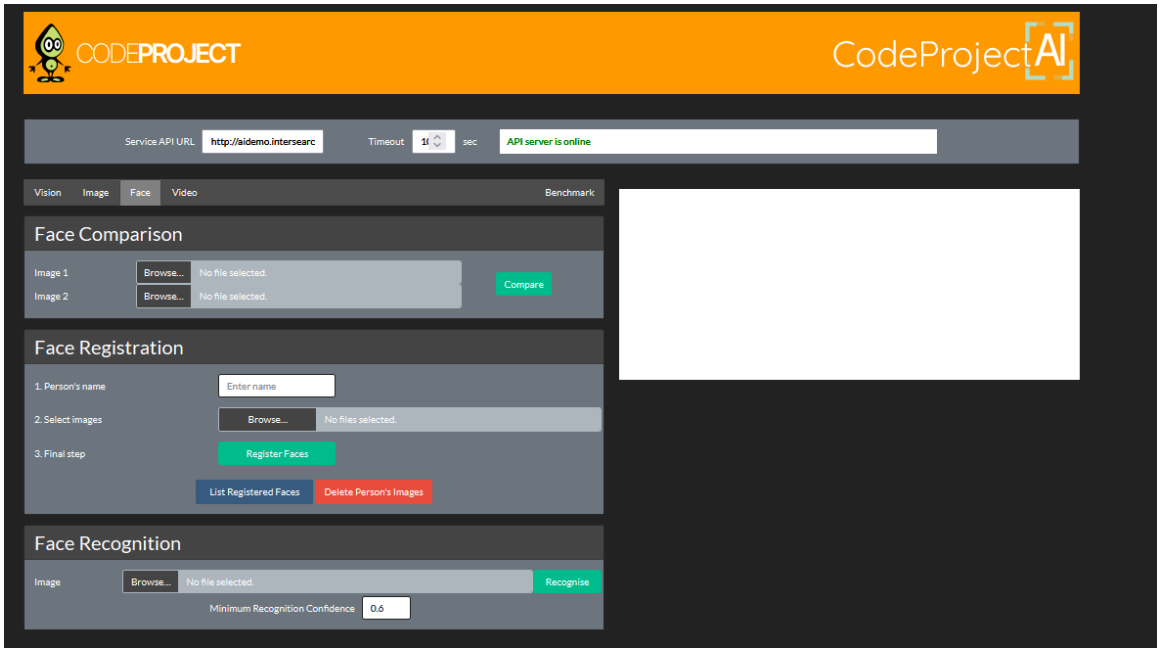
```
docker run -p 32168:32168  --name CodeProject.AI-Server -d -v
/usr/share/CodeProject/AI:/usr/share/CodeProject/AI codeproject/ai-server:1.6.8.0
```

In one command this:

- Downloads the docker project
- Unpacks the docker images and launches the docker processes

Code project makes its toolkit by default available via web port 32168 as a web server (i.e. http://localhost:32168). They also kindly provide a web testing interface at http://localhost:32168/vision.html . In our testing we exposed this interface as:

http://aidemo.intersearch.com.au:32168/vision.html



We used this to trial some initial face recognition. The function allows upload of an image and testing recognition of image variants.

The web interface above has corresponding Application Programming Interfaces (API) which are the methods for integrating with other systems – in this case Koha.

Cloud server setup

Cloud servers are available from many vendors, and are increasingly positioned globally across most continents and larger industrial centres. They provide a platform to create and use computing resources

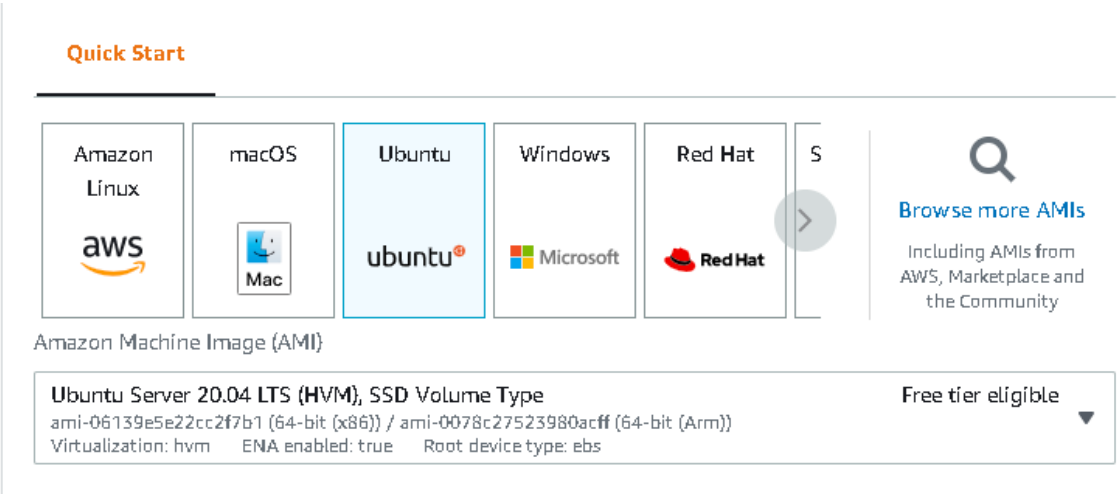
on a fractional basis with micro-billing tied to the amount of resources used. For this reason they provide a nice environment for concept testing software architectures. In this way a proof of concept of a new software solution can be tested in a "throw away" environment without large upfront hardware expenditure. The final deployment may or may not be in a cloud platform.

In this article this concept is used to demonstrate the implementation of Koha version 21.11 on an AWS Ubuntu 20.04 server with integration to AI capabilities of CodeProject's AI docker images.

It is easy to create an AWS server for throw-away testing. Creating the AWS instance requires that you:

- Register an AWS account (free but linked to a credit card)
- Create a server ("EC2" in AWS terminology). We created an "EC2" instance using the standard AWS ubuntu 20.04 server open source base. However, there are many flavours of computer operating system that can be deployed (at differing costs).

We installed Koha and Code Project initially on a *free tier* eligible server following the standard installation guidelines for Koha at https://wiki.koha-community.org/wiki/Koha_on_Debian - using the Ubuntu 21.11 pathway.



In this project we created a small instance in the free tier layer. However, in our testing we soon found that we could not install the image recognition software in the smallest instance. The target installation required at least 4GB of memory. We discovered this when the server "hung" when launching the CodeProject docker image at the same time as the Koha server. So, we "threw away" the server we had just created and started again with a larger image.

We finally deployed on an 8GB (EC2 type "T3 large:) server after running up and throwing away several sever instances. This cloud server has 2 x CPU and 8GB x memory.

In this way we arrived at a minimum architecture for the deployment environment. This is also how we arrived at our $5/day cost.

Implementation of Koha

Koha is one of the most widely implemented library management systems in the world. It is an open-source library management solution supported by a rich community of developers, service providers and librarians around the globe.

Koha is an open-source library management system that runs on Debian, Ubuntu or other flavours of Linux. It has a well-documented installation process (see https://wiki.koha-community.org/wiki/Koha_on_Debian). We installed Koha on our EC2 instance alongside our docker CodeProject instance. It took 20 minutes to install and deploy Koha on the server (with the advantage of familiarity).

Integrating with Koha

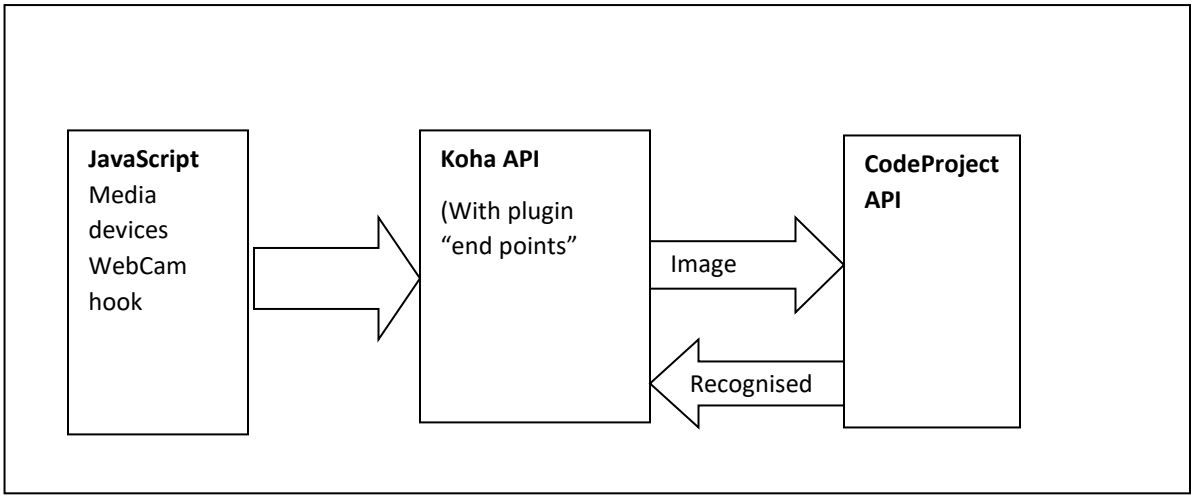
Koha provides a "plugin" system that allows integration of features within the application. The plugin system means that the developer does not have to touch the code base of the application. The plugin system itself has API hooks and hooks to place code within the application. In the case of this project, we developed a plugin based on existing community published plugins.

This plugin does two minimal functions for the project integration task:

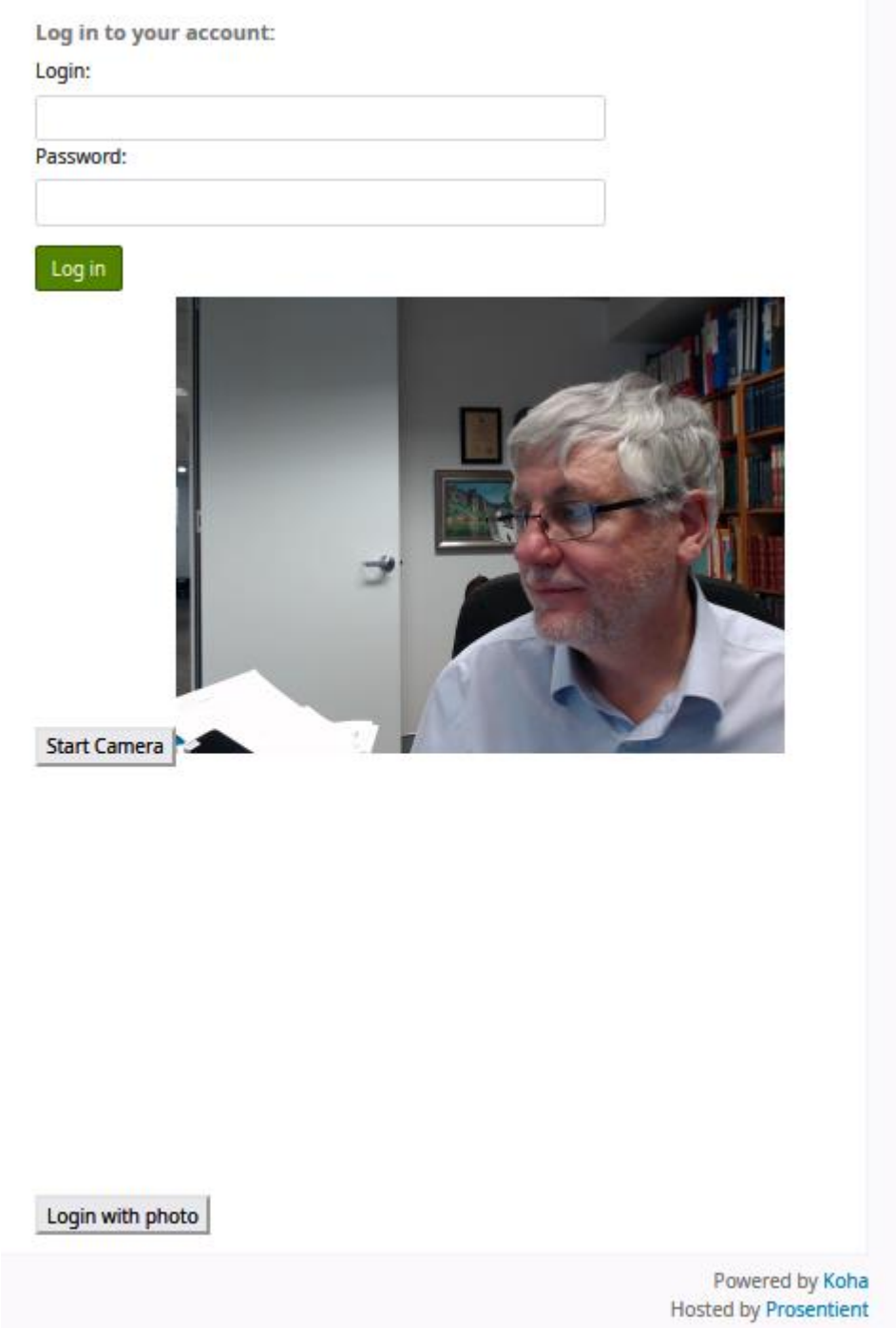
- An integration hook between the OPAC login page and the AI server to
    - Scan an image
    - Send the image to CodeProject AI
    - Authenticate the use on Koha based on a successful match
- An integration hook on the Patron page to allow transfer of an uploaded patron image to CodeProject AI

Both development tasks essentially involved "injecting" JavaScript into the login page and librarian's patron editing page to upload images using the Koha Plugin API integration as follows:

- The OPAC login page shows JavaScript to capture an image using the webcam
- The JavaScript's sends to image to Koha using the Koha Plugin integration with the Koha API
- The Koha plugin sends the image to the Code Project AI server for validation or registration



By relying heavily on existing plugin examples, and using the APIs in JavaScript, Koha and CodeProject, the proof-of-concept development took two days to completed. This was the most complex programmatic task in this exercise. It involved software development focussed primarily on interacting with the Koha plugin functions and the CodeForLibAI functions.



The Online Public Access Catalogue (OPAC) login page offers the end user the *choice* to use the camera to sign in.

Testing the AI toolkit – Confidence levels

Artificial Intelligence systems rely on models of the real world of one sort or another. Image recognition for instance can use many instances of sample images to "train" a model for subsequent recognition of similar instances. The more samples the better the matching. When a applying a real-world instance against a model, the process is always an approximation: this image matches *this much*. This is the level of confidence.

The matching confidence levels are crucial in the context of an authentication facial image recognition. Too low a confidence level accepted from the matching algorithm and any face is happily accepted. Too high a confidence level and it is functionally impossible to log in. Our initial explorations indicated a confidence level of > 0.75 as a minimum confidence.

The important thing in using AI of any sort is the algorithms used and methodology for use, and whether it provides levels of confidence at any of its modelling or processing layers. The benefit of open source is the whole process is open to scrutiny. The CodeProject AI also yields a confidence level on its image and facial matching – this can be used to good effect in the overall testing.

CodeProject AI does not store the original image once it has captured and processed that image. It simply stores a vector representing the face captured. Exploring the source reveals an SQLITE database. The project is a wrapper for a well-known open-source projects – in this case ObjectDetectionYolo which itself uses OpenCV.

How does this toolkit store the face? No image is saved by the toolkit. Instead, it stores a vector defining the unique points defining the face, using the underlying image recognition software. It stores this vector in a database table, which can be explored as follows:

```
apt install sqlite

sqlite3 ./faceembedding.db

.tables

List the table

Select * from TB_EMBEDDINGS
```

We see a set a vector array that describe the unique points for a face.

```
0.026341065674227185, 0.03604118525981903, 0.05735778045518771, -0.04988788140183555, -
0.001674302271567285, 0.028918949887155787, -0.006237712223018781, 0.008357822452184116,
0.08690883219222096, 0.03714309023022652, 0.03508520871400833, 0.016281576706784135,

……

0.05015251785516739, 0.008849352598190308, 0.06125432997941971, -0.00867899414151907,
0.042281255453185, -0.010691865347325802, -0.02570171467959881, -0.01565711200237274, -
0.03459269553222928, -0.011747458018362522, -0.03084883838891983, -0.05788338112831116, -
0.09049146135759354, -0.020388029515743256, -0.061122243144919586, 0.026703915790077972,
0.0673513263463974, 0.04039162260029793, 0.018179498612880707]
```

(Note – this is not a real vector).

The API call returns matched face ids and the confidence level for the match.

The application using the toolkit is left to make a decision on what to do based on the confidence levels of the match.

WebCam integration, Secure Server and image scanning embedded in a website

To be useful within the Koha application, it was necessary to allow Koha to take an image snapshot to send to the server.

This could have been the trickiest part of the project: accessing the web cam from the browser to scan facial images. Fortunately, this is a problem solved several years ago with the browser introduction of HTML5 and media functions for the browser. This allows JavaScript functions to access and manipulate images from the Web Cam.

One of the first integration issues with the web interface for Koha was that accessing the Web Cam. For good security reasons, you cannot access the webcam from a non-SSL website. The reason for this is sound: anyone on the local network could watch or listen to your video/audio stream if the stream is not through a secure connection. Firefox and Chrome disable to WebCam functions if the web page itself is not secure.

Secure Socket Layer (SSL) is the technology used to create a secure information interchange between your browser (Chrome/Firefox, etc) and the server you are connecting to. Web sites use public certificates shared with your browser to create this secure connection.

Creating an SSL (https) protected website is an additional step required for this proof of concept. While there are options for what are called "self-signed" certificates, they do not always overcome the obstacle of using secured components of the browser.

To create an SSL-protected website there are two components.

- Creating a domain name. This will be the public web domain of the website on which our Koha will be hosted. We use aidemo.intersearch.com.au. Creating a domain name requires the use of a domain registration company (like GODADDY) or your own organisation if they have that capability.
- Installing a secure certificate for the domain. An SSL certificate needs to be installed on your server. This certificate is time limited (these days a maximum of a year) and must be created through a well-known registrar (one that is widely recognised on client computers and devices). There can be a fee associated with this certificate creation, but some sites provide the certificate for free as part of the hosting.

In this project we had control of an existing domain, intersearch.com.au so we registered two sub-domains for Koha: aidemo.intersearch.com.au and aidemoadmin.intersearch.com.au. These domains translated to our cloud web server.

We created the following domains and registered certificates for them:

https://aidemo.intersearch.com.au

https://aidemoadmin.intersearch.com.au

Fortunately, there is a free certificate service called "Lets Encrypt" (https://letsencrypt.org/). This initiative provides an open access method for registering certificates at no charge. This terrific service provides a solid and robust means for encryption of your website. It has been responsible for a much

greater level of systematic website encryption. Let's Encrypt is a non-profit authority. It takes sponsorships and donations. Support for this wonderful service is greatly to be encouraged.

LetsEncrypt can be integrated with the servers very easily with a tool "certbot".

https://www.inmotionhosting.com/support/website/ssl/lets-encrypt-ssl-ubuntu-with-certbot/

Install snapd:

- sudo apt install snapd
- Ensure you have the latest snapd version installed:
- sudo snap install core; sudo snap refresh core
- Install Certbot with snapd:
- sudo snap install --classic certbot
- Create a symlink to ensure Certbot runs:

sudo ln -s /snap/bin/certbot /usr/bin/certbot

Once installed, the certbot tool can be used to create certificates that are automatically renewed on this server.

Ethical review

Any project that captures biometric data, or data that could be interpreted as biometric, raises some considerations.

Retention: how long is the image data to be retained – is it removed after it's relevant usage or when the client is no longer a library subscriber

Access: who can access the library subscriber data? For instance, in our production implementations for Koha we added two factor identification requirements for all reports that contain library subscriber data, in addition to the login security for the system.

Appropriate use: is the captured image data used appropriately and only for allowed use. In this case its purpose is to provide a simple, hands free, face-based login. There are evident weaknesses with such a system that might not make it suitable in, for instance, a public library setting. It might, however , be very useful in a 24x7 professional/research library setting.

Consent: the use of the face recognition login should not be activated automatically – in this case it is triggered by the end user, should they wish to use this method for authentication. Traditional password authentication is still also enabled.

Wrapping up the proof of concept

The purpose of this exploration was to determine the feasibility and complexity of integrating an image recognition toolkit with the well-known open-source library system Koha. On the journey we demonstrated the use of a range of methodologies for building such a tool kit with little infrastructure cost, indeed on a budget of $5/day.

From here to production

There is long journey from a “proof of concept” to a “production implementation”. Considerations for a production implementation include:

- Testing across different devices, environments
- Testing across different facial profiles/population/gender sets
- A security review of the solution. How safe are all the components used. What vulnerabilities apply and can they be mitigated
- A project ethical review