Auto Check-In App

AWS Implementation Guide

Yoshitaka Haribara

Shota Iizuka

Kazuya Iwami

Daisuke Miyamoto

Tomoya Okuno

January 2020

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Auto Check-In App in the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting in the AWS Cloud.

Overview

Event check-in is the first on-site touchpoint for most events, and the process is your first opportunity to make a good impression with your attendees. An optimized check-in process can help alleviate long lines and attendee confusion, which improves your attendee experience and helps set the tone of your event.

One way to optimize the check-in process is to automate it using self-check-in. Self-check-in kiosks and apps can enable attendees to check-in through a name search or QR code scan, print their badges, and enter your event quicker. But, self-check-in requires attendees to enter their name in a search interface, or present a QR code to be scanned which can slow the check-in process.

Facial comparison can make it even easier and quicker for attendees to check-in by eliminating the need to enter names or show QR codes. Attendees can have their picture taken and compared with a pre-registered attendee face collection in seconds.

Amazon Rekognition is a fully managed service that makes it easy to add deep learning powered visual analysis to your applications. With Amazon Rekognition, you can compare faces for a wide variety of use cases, including event attendee verification.

To help make it easier for customers to automate the check-in process, Amazon Web Services (AWS) offers the Auto Check-In App. This solution automatically provisions the products and services necessary to provide facial comparison for event attendee check-in.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of the date of publication, the cost for running this solution with default settings in the US West (Oregon) Region is approximately **$10 per hour**. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

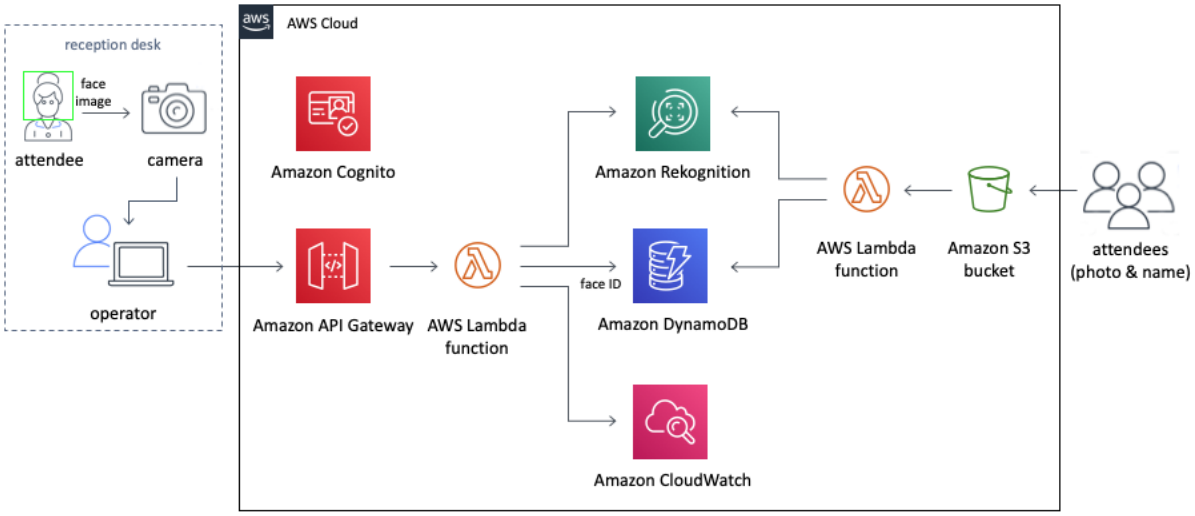

Figure 1: Auto Check-In App architecture

The AWS CloudFormation template deploys an Amazon API Gateway, an Amazon Cognito user pool, AWS Lambda functions, Amazon DynamoDB tables, Amazon Rekognition, Amazon CloudWatch, and an Amazon Simple Storage Service (Amazon S3) bucket.

When an attendee registers for your event, they can upload a photo which is stored in an Amazon S3 bucket. The upload to Amazon S3 triggers a Lambda function that calls the Amazon Rekognition IndexFaces API. Amazon Rekognition extracts facial features into a feature vector and creates a `face_id`. Then, the vector is stored in a face collection, and the `face_id` and corresponding user name are stored in a DynamoDB table.

At the event, an operator uses the solution's Python-based UI and a camera to take a picture of the attendee, crop the photo, and send the photo to Amazon API Gateway which triggers a

Lambda function that calls the Amazon Rekognition SearchFacesByImage API. Amazon Rekognition extracts the facial features from the image into a feature vector and compares the vector to the vectors in the face collection. When Amazon Rekognition finds a face with high similarity, the `face_id` is used to retrieve the user name. The user name is sent to the operator's laptop showing that the attendee has been authenticated.

Solution Components

Solution API

The Auto Check-In App configures Amazon API Gateway to host the solution's RESTful API. Operators can interact with data securely through the included UI and RESTful API. The API acts as a "front door" for access to data stored in Amazon DynamoDB. You can also use the APIs to access any extended functionality you build into the solution.

This solution takes advantage of the user authentication features of Amazon Cognito User Pools to manage operators of the solution. After successfully authenticating an operator, Amazon Cognito issues a JSON web token that is used to allow the console to submit requests to the solution's APIs (Amazon API Gateway endpoints). HTTPS requests are sent to the APIs with the authorization header that includes the token.

Based on the request, Amazon API Gateway invokes the appropriate AWS Lambda function to perform the necessary tasks on the data stored in the DynamoDB tables.

Amazon CloudWatch Dashboard

This solution provides an optional dashboard you can use to monitor the performance of your deployment. The dashboard displays custom operational Amazon CloudWatch metrics for your deployment, including the Amazon Rekognition response time and success count, the Amazon DynamoDB successful request latency, and the Amazon API Gateway latency.

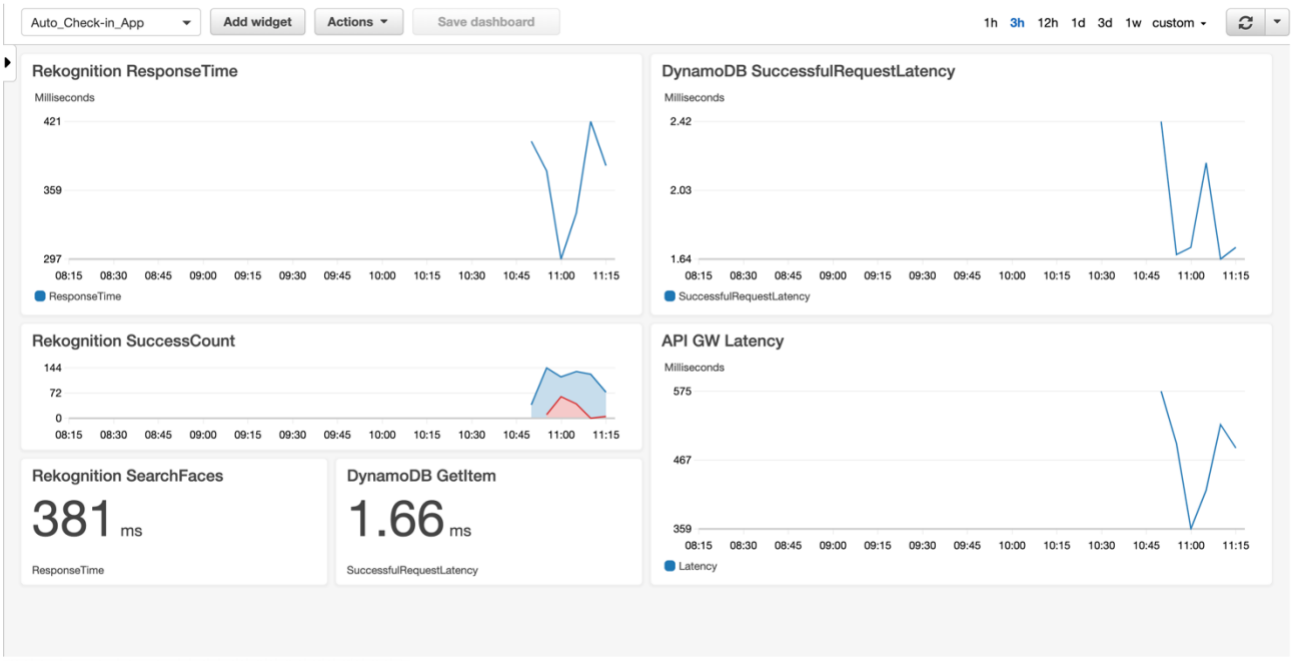

Figure 2: Amazon CloudWatch dashboard

Considerations

Notice

This guide focuses solely on architectural and technical issues related to developing and deploying the Auto Check-In App. It does not cover non-architectural or technical considerations that may be associated with this use case, such as legal and privacy issues. For example, in certain geographies there may be laws regarding the collection, storage, and processing of biometric information. You should review legal requirements in your jurisdiction and obtain legal advice where appropriate.

In addition, there are a number of privacy and transparency considerations around the use of facial comparison in commercial settings. You should educate yourself on these issues and take them into account when designing and deploying your application. We recommend that you provide information to end users about how the application works and how data collected from the application will be used, stored, and retained, and offer an alternate verification option for end users who do not wish to use the application. For more information, review the following additional resources and best practices:

- Facing Facts: Best Practices for Common Uses of Facial Recognition Technologies
- Privacy Best Practice Recommendations For Commercial Facial Recognition Use

- Privacy Principles for Facial Recognition Technology in Consumer Applications

Amazon Rekognition

Amazon Rekognition uses deep learning models to perform face detection and to search for faces in collections. It continues to improve the accuracy of its models based on customer feedback and advances in deep learning research. These improvements are shipped as model updates.

When you launch the Auto Check-In App solution, the face collection that is created is associated with the most recent version of the model. If a new version of the model is released, this solution's collection will continue to use the model that is already associated with the collection. For more information, see Model Versioning in the *Amazon Rekognition Developer Guide*.

Similarity Threshold

This solution uses a similarity threshold to determine whether two faces have high similarity. The threshold controls how many results are returned based on the similarity to the face being matched. Responses that are below the threshold are not returned.

The threshold is a value between 0 and 100. The default threshold is 99. For more information, see Searching for Faces Within a Collection in the *Amazon Rekognition Developer Guide*.

Registration Image Preparation

During event registration, allow event attendees who want to use facial comparison during check-in to send an image of their face. After an attendee sends the image, make sure that the image is formatted correctly. The image should be in .jpg, .jpeg, or .png format, and the file name should be the name that you expect to be displayed during the event. For example, `jeff.jpg` or `andy.png`. Also, make sure that the attendee's face is the only face in the image. For additional information, see Recommendation for Facial Recognition Input Images in the *Amazon Rekognition Developer Guide*. Then, upload the photo to the solution's Amazon Simple Storage Service (Amazon S3) bucket.

Event Image Preparation

To help ensure the most accurate results, make sure that your attendees' faces are sufficiently lit, and that your images are larger than 80x80 pixels. We recommend using an additional light to ensure the face is bright enough. For more information, see Limits in Amazon Rekognition in the *Amazon Rekognition Developer Guide*.

To reduce the latency from Amazon Rekognition and ensure the correct face is indexed, we also recommend cropping the facial images you capture at the event before sending them to Amazon API Gateway. By cropping the image, you can reduce the response time and remove unintended faces from the image. For more information, see Amazon Rekognition Image Operation Latency in the *Amazon Rekognition Developer Guide*.

To help prevent the solution from indexing unintended faces, this solution indexes the largest face in the image. All other faces in the image are not indexed.

Image Storage

After Amazon Rekognition extracts the facial features from a registration photo into a feature vector and adds the vector to the face collection, the registration photo is automatically deleted from Amazon S3. The photo taken at the event is not stored. The solution does not store any facial images after they are processed.

Processing Status

This solution includes the following status.

- `Stop here`: No face is detected
- `Checking`: A face is detected and being compared to faces in the face collection
- `Welcome`: A face is found with high similarity

While a facial image is being processed, the status will change from `Stop here` to `Checking`. If a face with similarity above the threshold you defined during initial deployment, the status will change to `Welcome`. If no faces with similarity above the threshold are found, the status will stay in the `Checking` state.

Event Security

For customers who use this solution for events with additional security requirements, we recommend using multi-factor authentication during check-in as an additional layer of security. For example, in addition to the face matching, ask attendees to show their government-issued ID.

This solution does not provide anti-spoofing or liveness detection. You can, however, customize the solution to add this functionality. For example, you can send attendees a temporary code to use in addition to the picture at the event. You can also build a challenge-based workflow where the attendee has to perform a series of randomized steps such as smiling, tilting their head, or closing their eyes to verify liveness.

Operator Management

The Auto Check-In App includes a Shell script to grant you access to the solution's UI. The UI does not provide operator administration. To add additional operators, you can use the included script. For more information, see Step 3.

Testing

We recommend testing the functionality of this solution before your event. We also recommend using the laptop and camera you plan to bring to the event.

Test Configuration

While you can use any laptop, camera, and monitor with the Auto Check-In App, we have tested this solution using an Apple MacBook Pro, a LogiTech C920S webcam, and a Dell P2419H monitor. We recommend changing the monitor settings to portrait mode and putting the camera on top of the monitor. We also recommend using an additional light to ensure the faces are well lit.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Auto Check-In App in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment:

View template

auto-check-in-app.template: Use this template to launch the Auto Check-In App and all associated components. The default configuration deploys an Amazon API Gateway, an Amazon Cognito user pool, AWS Lambda functions, Amazon DynamoDB tables, Amazon Rekognition, Amazon CloudWatch, and an Amazon Simple Storage Service (Amazon S3) bucket. But you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the Auto Check-In App into your account.

Time to deploy: Approximately five minutes

Prerequisites

Before you launch the Auto Check-In App, you must have the AWS Command Line Interface (AWS CLI) version 1.16.243 installed, Python 3.7, and OpenCV 4.1.0.

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the Stack

- Launch the AWS CloudFormation template into your AWS account.
- Review the template parameters, and adjust if necessary.

Step 2. Upload Images to Amazon S3

- Upload attendee face images to the solution's Amazon S3 bucket.

Step 3. Download and Configure the App

- Download and configure the included UI.

Step 4. Launch and Run the App

- Take the attendee's photo and let the solution compare the image to the face collection.

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the Auto Check-In App in the AWS Cloud. Make sure that you have reviewed the prerequisites and considerations before launching the stack.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `auto-check-in-app` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

   Launch Solution

2. The template is launched in the US West (Oregon) Region by default. To launch the solution in a different AWS Region, use the region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Log Level	`INFO`	Choose a log level for the AWS Lambda functions. Choose `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`, or `NOSET`.
Registration Bucket Name	`auto-check-in-app-register`	The name of the Amazon S3 bucket where face images are uploaded and stored during registration.
Rekognition Collection Name	`auto-check-in-app-face-collection`	The name of the Amazon Rekognition face collection.
Rekognition Face Similarity Threshold	`99`	Specify a value between 0 and 100 that determines how similar the captured face image must be to the pre-registered face image.

6. Choose **Next.**
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately five minutes.

> **Note:** In addition to the primary AWS Lambda function, this solution includes the `auto-check-in-app-CreateCollection` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
>
> When running this solution, the `auto-check-in-app-CreateCollection` function is inactive. However, do not delete the function as it is necessary to manage associated resources.

Step 2. Upload Images to Amazon S3

> **Note:** We recommend testing the solution before your event.

Before you start uploading images to the solution's Amazon Simple Storage Service (Amazon S3) bucket, make sure you review the image preparation guidelines.

You can use any tool to upload the images to the solution-created Amazon S3 bucket for registration photos. If you used the default setting, the bucket name is `auto-check-in-app-register-<region>-<account-id>`. If you changed the default name, you can find the bucket name in the stack **Outputs** tab. The bucket name is the value of **RegistrationBucketName** key.

After you successfully upload a photo, the facial features are automatically extracted into a feature vector and added to the Amazon Rekognition face collection. After the vector is added, the original photo is automatically deleted from the Amazon S3 bucket.

Step 3. Download and Configure the App

Use this procedure to download and configure the included UI.

1. On the laptop you will use during your event, run the following command in your preferred directory to clone the application source code from GitHub.

```
$ git clone https://github.com/awslabs/auto-check-in-app.git
```

2. After the code is cloned, navigate to the `auto-check-in-app/source/frontend` directory.
3. Run the following command.

```
chmod +x register-operator.sh
```

4. Run the following command

```
./register-operator.sh <operator E-mail address>
```

> **Note:** Specify the applicable AWS Region and AWS CloudFormation stack name in the script.

5. Enter a password.
6. Open the `default.env.json` file and copy it to `env.json`.
7. Modify the file with the applicable values.

> **Note:** You can find the applicable values in the stack **Outputs** tab.

The following table shows the applicable output keys and values.

Key	Value	Description
Region	`us-west-2`	The AWS Region where you launched the stack.
API Endpoint	*<rest-api-id>*`.execute-api.us-west-2.amazonaws.com/prod/recognize_face`	The Amazon API Gateway endpoint name with the REST API ID.
Cognito User Pool ID	*<region>_<ID>*	The Cognito user pool ID.

Key	Value	Description
Cognito User Pool Client ID	*<client ID>*	The Cognito user pool client ID.

Step 4. Launch and Run the App

Use this procedure to launch the Auto Check-In App and start comparing faces.

1. To launch the check-in app, run the following command.

```
$ python main.py
```

2. When prompted, enter your user name and password.
3. Use a built-in camera or a USB camera to take a picture of the attendee's face. While the image is being processed, the status will change from `Stop here` (no face is detected) to `Checking` (the face is detected and being compared to the face collection) to `Welcome` (a face is found with high similarity).

> **Note:** After the status changes to `Welcome`, it will change back to `Checking` for 10 seconds, by default, to avoid recurrent acceptance signals. To change the amount of time, modify the `VisibilityTimeout` parameter in the `env.json` file.

After the image is taken, a window opens to show the face detection and recognition results.

```
{'result': 'OK', 'name': '<username>', 'similarity':
99.71908569335938}
{'result': 'OK', 'name': '<username>', 'similarity':
99.79707336425781}
{'result': 'OK', 'name': '<username>', 'similarity':
99.7418441772461}
```

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

IAM Roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on AWS. This solution creates several

IAM roles, including roles that grant the solution's AWS Lambda functions access to other services used in this solution.

Encryption

The Auto Check-In App leverages HTTPS which uses SSL/TLS encryption for data in transit. By default, server-side encryption is enabled on the solution's Amazon Simple Storage Service (Amazon S3) bucket and Amazon DynamoDB table to encrypt data at rest. For more information, see Protecting Data Using Server-Side Encryption in the *Amazon S3 Developer Guide*, and DynamoDB Encryption at Rest in the *Amazon DynamoDB Developer Guide*.

Additional Resources

AWS services

- Amazon Rekognition
- AWS Lambda
- Amazon DynamoDB
- Amazon Simple Storage Service
- Amazon API Gateway
- Amazon CloudWatch
- Amazon Cognito
- AWS CloudFormation

Other resources

- Facing Facts: Best Practices for Common Uses of Facial Recognition Technologies
- Privacy Best Practice Recommendations For Commercial Facial Recognition Use
- Privacy Principles for Facial Recognition Technology in Consumer Applications

Appendix : Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Similarity Value:** The similarity value returned by the Amazon Rekognition `SearchFacesByImage` API
- **Request Image Size:** The size of the images used for registration and searching
- **Response Time and Request Latency:** The time it takes for the solution to respond to the request

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

```
Mappings:
   Metrics:
      Send-Data:
         SendAnonymousData: "Yes"
```

to

```
Mappings:
   Metrics:
      Send-Data:
         SendAnonymousData: "No"
```

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
January 2020	Initial release