Facial Recognition on Android Using Google Cloud AutoML

Andrew Kelly - Google Developer Expert for Android

@andrew_ke11y

https://medium.com/devnibbles

Steps

1. Detect a face
2. Classify the face
3. …
4. Profit

Actual Steps

1. Checkout base Android app - make sure it runs.
2. Setup Firebase and Google Cloud project
3. Upload photos to Google Cloud AutoML
4. Add face detection to Android app
5. Train model in Google Cloud AutoML
6. Use Google Cloud AutoML REST or SDK
   from the Android app
7. …
8. Profit

10 mins

Running the Android App

GitHub Project

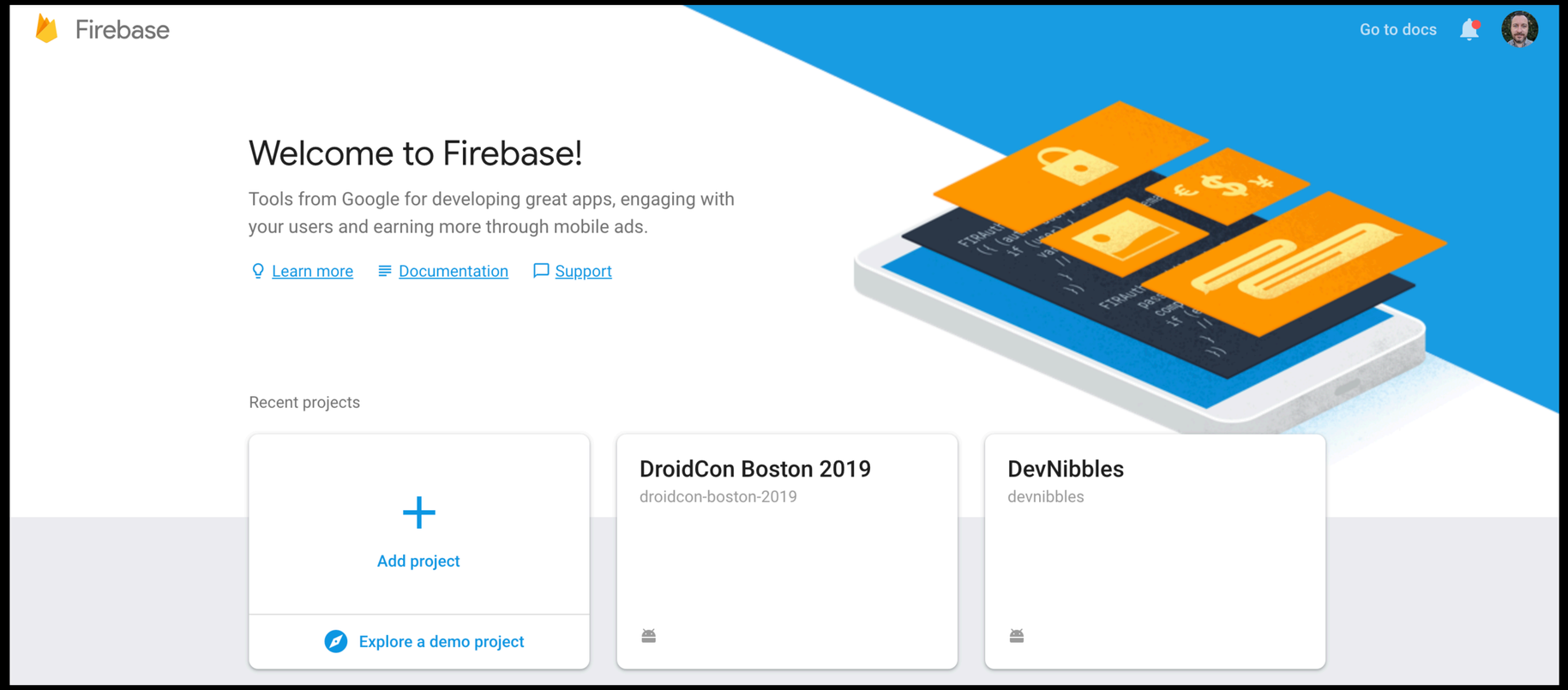
- We have a GitHub starter repository that’s done some of the boring work for you.
  - Requesting Permissions
  - Basic Architecture

- https://github.com/apkelly/DroidCon-Boston-2019
- Or point you git client to the repository on the USB stick

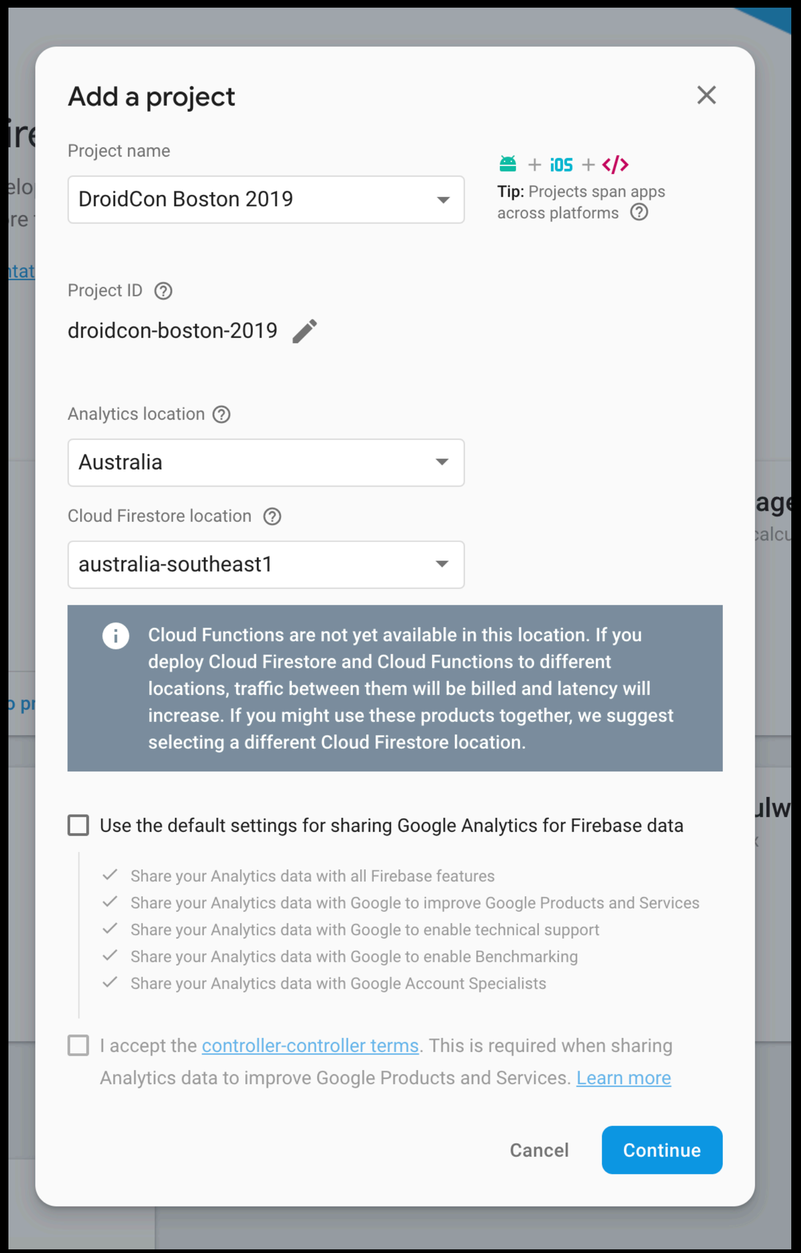
- Let’s open Android Studio and explore this starter project together.
- When setup successfully you should see a camera preview inside the app.

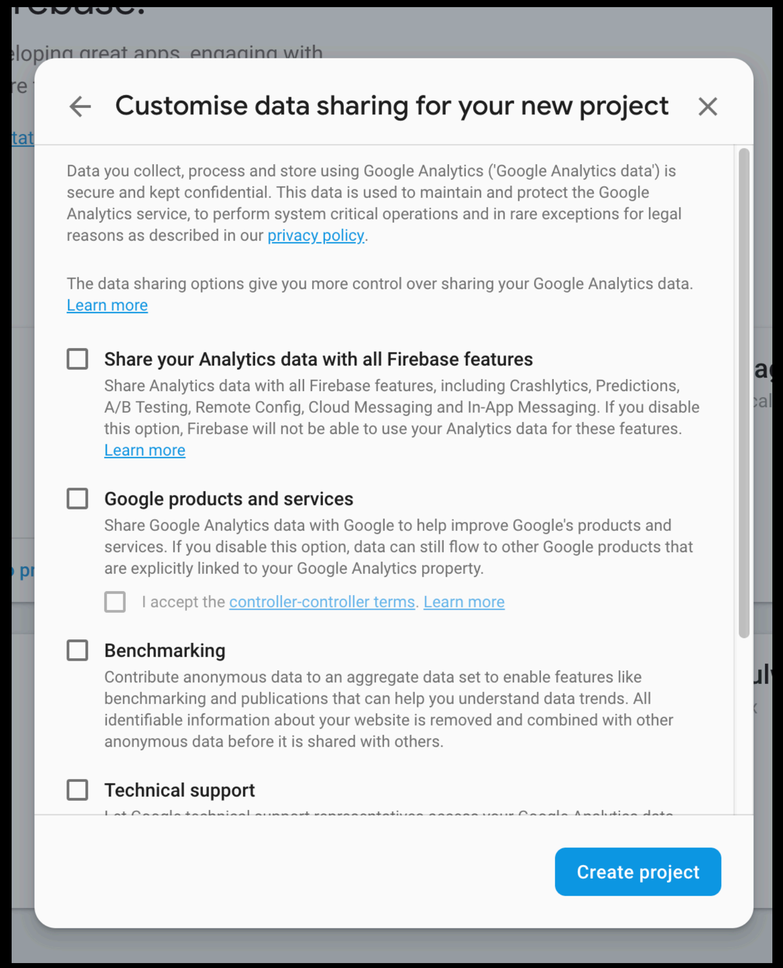
5 mins

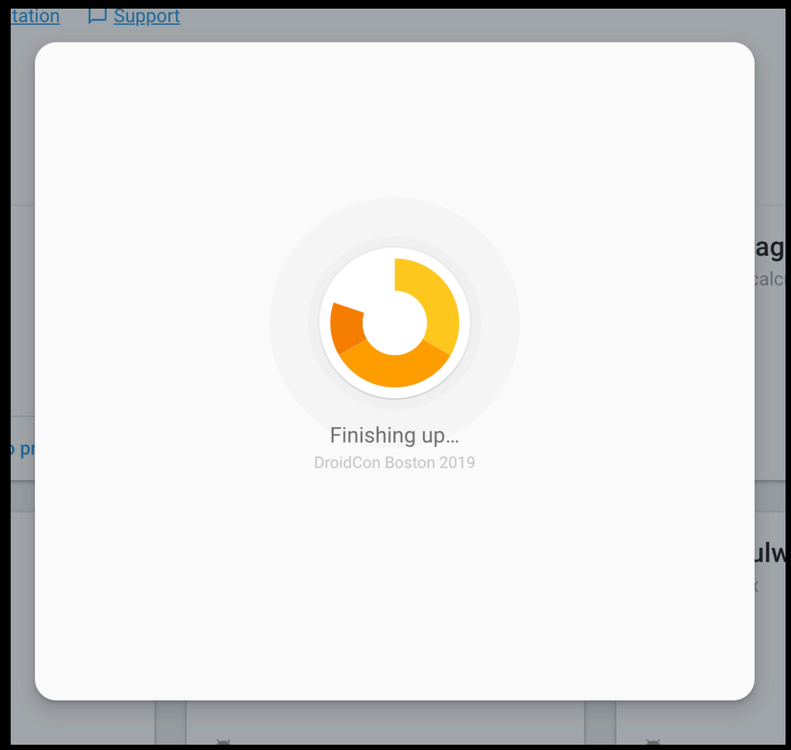
Setup Firebase

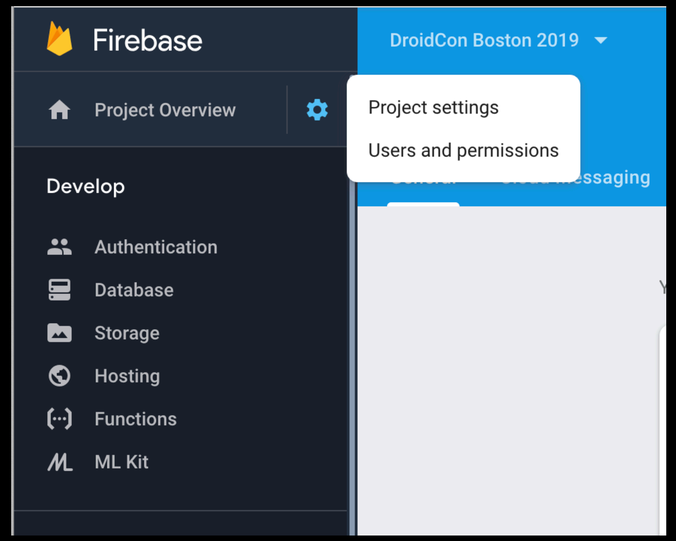


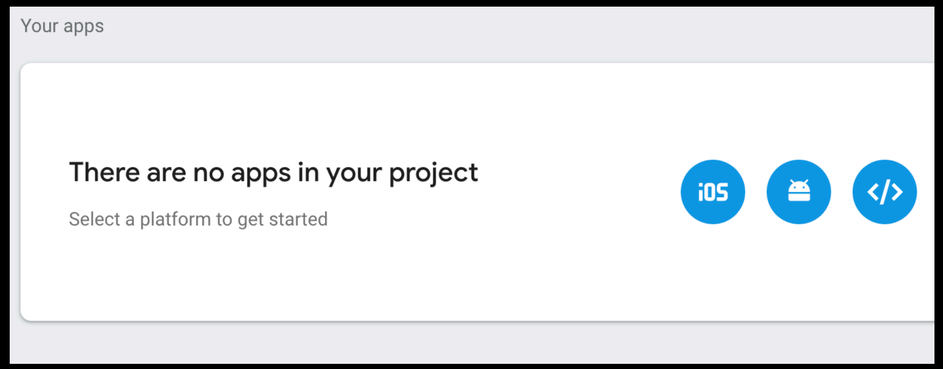
https://console.firebase.google.com/

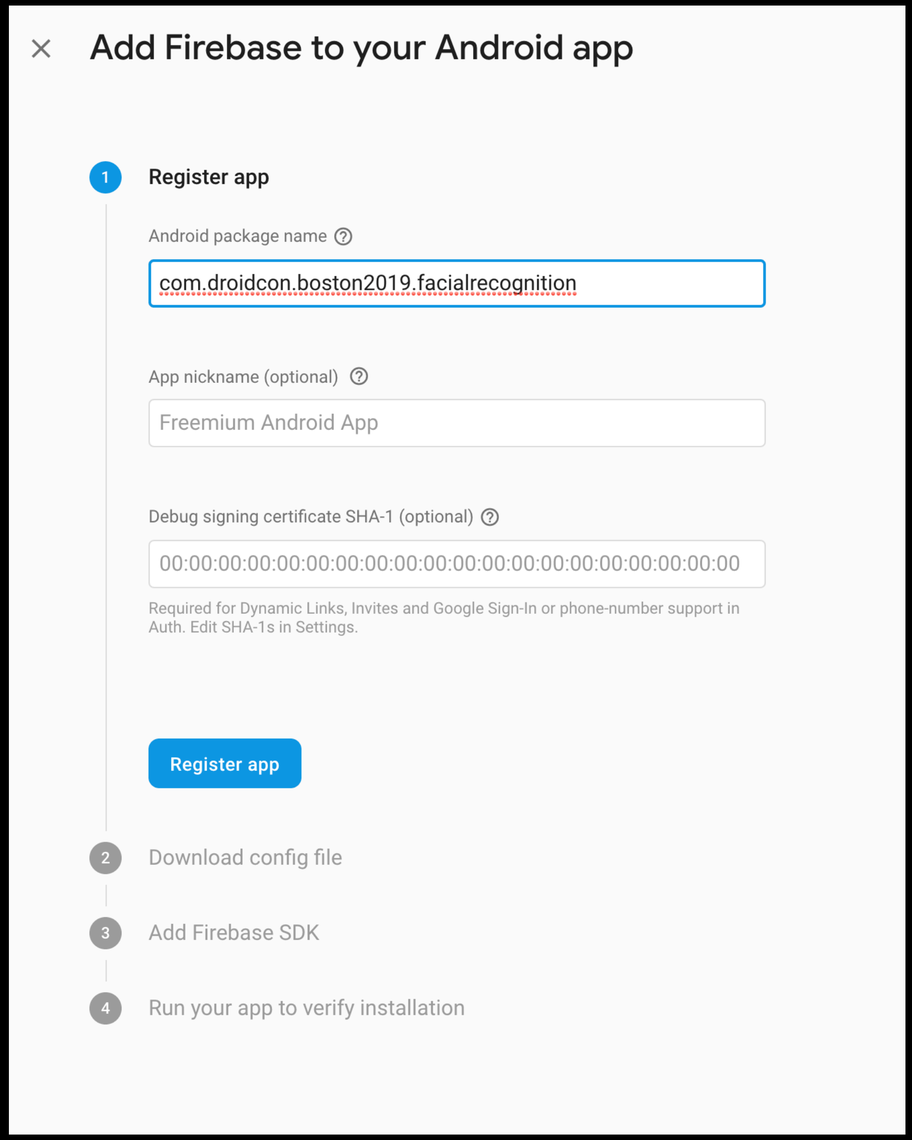

Add a project
Project name
DroidCon Boston 2019
Tip: Projects span apps across platforms
Project ID
droidcon-boston-2019
Analytics location
Australia
Cloud Firestore location
australia-southeast1
Cloud Functions are not yet available in this location. If you deploy Cloud Firestore and Cloud Functions to different locations, traffic between them will be billed and latency will increase. If you might use these products together, we suggest selecting a different Cloud Firestore location.
Use the default settings for sharing Google Analytics for Firebase data
Share your Analytics data with all Firebase features
Share your Analytics data with Google to improve Google Products and Services
Share your Analytics data with Google to enable technical support
Share your Analytics data with Google to enable Benchmarking
Share your Analytics data with Google Account Specialists
I accept the controller-controller terms. This is required when sharing Analytics data to improve Google Products and Services. Learn more
Cancel
Continue


Customise data sharing for your new project
Data you collect, process and store using Google Analytics ('Google Analytics data') is secure and kept confidential. This data is used to maintain and protect the Google Analytics service, to perform system critical operations and in rare exceptions for legal reasons as described in our privacy policy.
The data sharing options give you more control over sharing your Google Analytics data. Learn more
Share your Analytics data with all Firebase features
Share Analytics data with all Firebase features, including Crashlytics, Predictions, A/B Testing, Remote Config, Cloud Messaging and In-App Messaging. If you disable this option, Firebase will not be able to use your Analytics data for these features. Learn more
Google products and services
Share Google Analytics data with Google to help improve Google's products and services. If you disable this option, data can still flow to other Google products that are explicitly linked to your Google Analytics property.
I accept the controller-controller terms. Learn more
Benchmarking
Contribute anonymous data to an aggregate data set to enable features like benchmarking and publications that can help you understand data trends. All identifiable information about your website is removed and combined with other anonymous data before it is shared with others.
Technical support
Create project


Support
Finishing up...
DroidCon Boston 2019



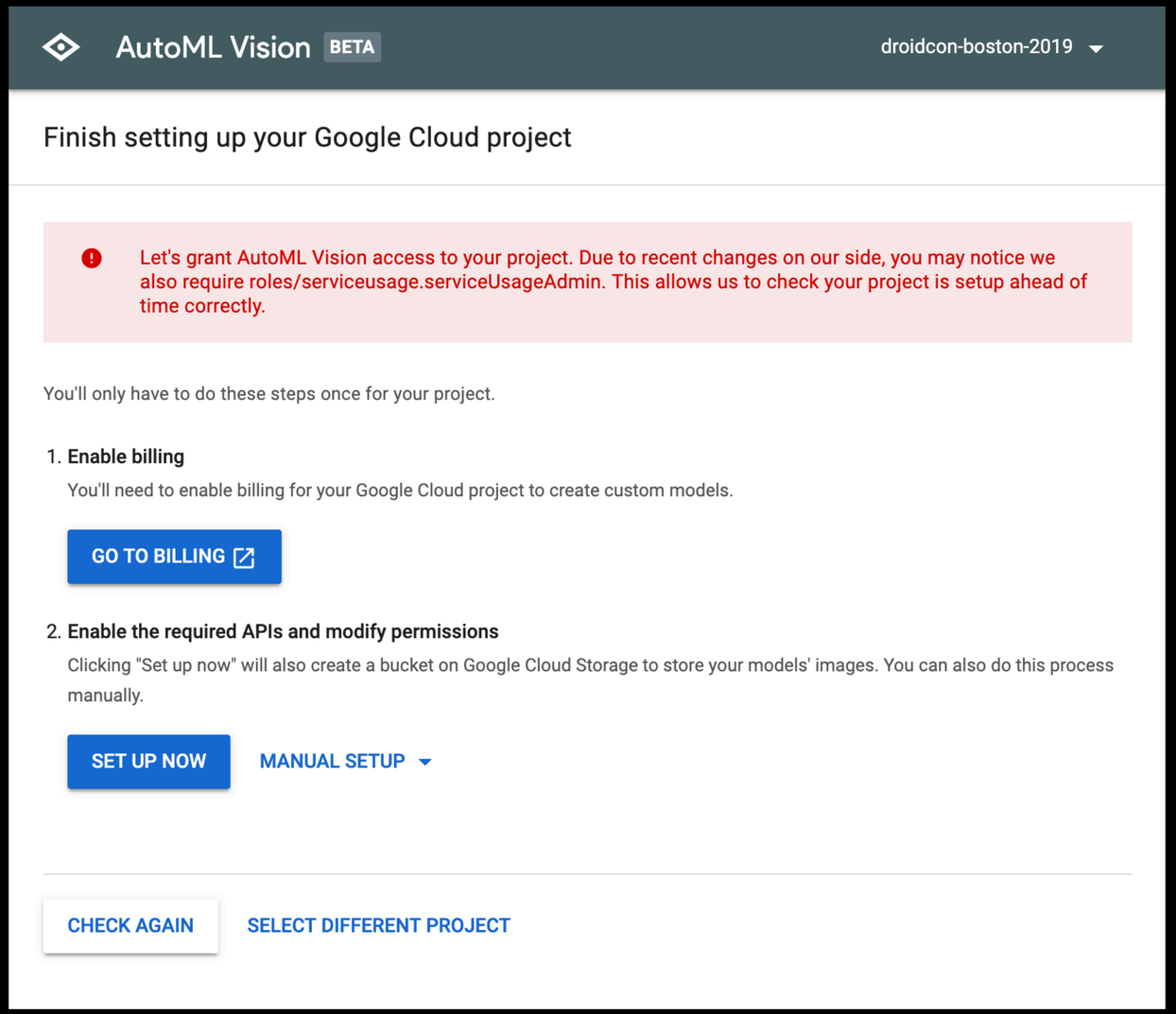




Package name is important - must match android app

5 mins

Setup Google Cloud

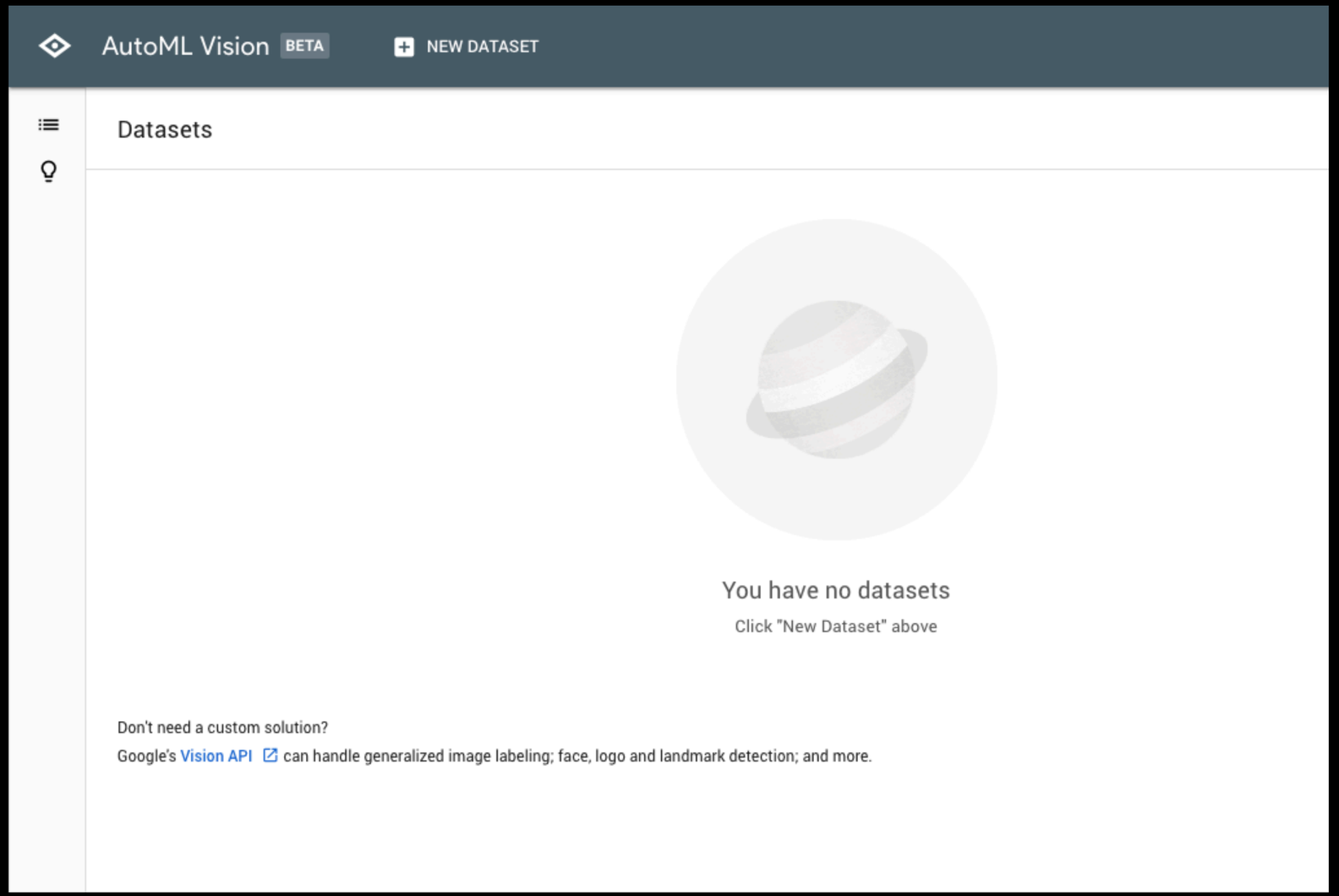


https://cloud.google.com/automl/ui/vision/

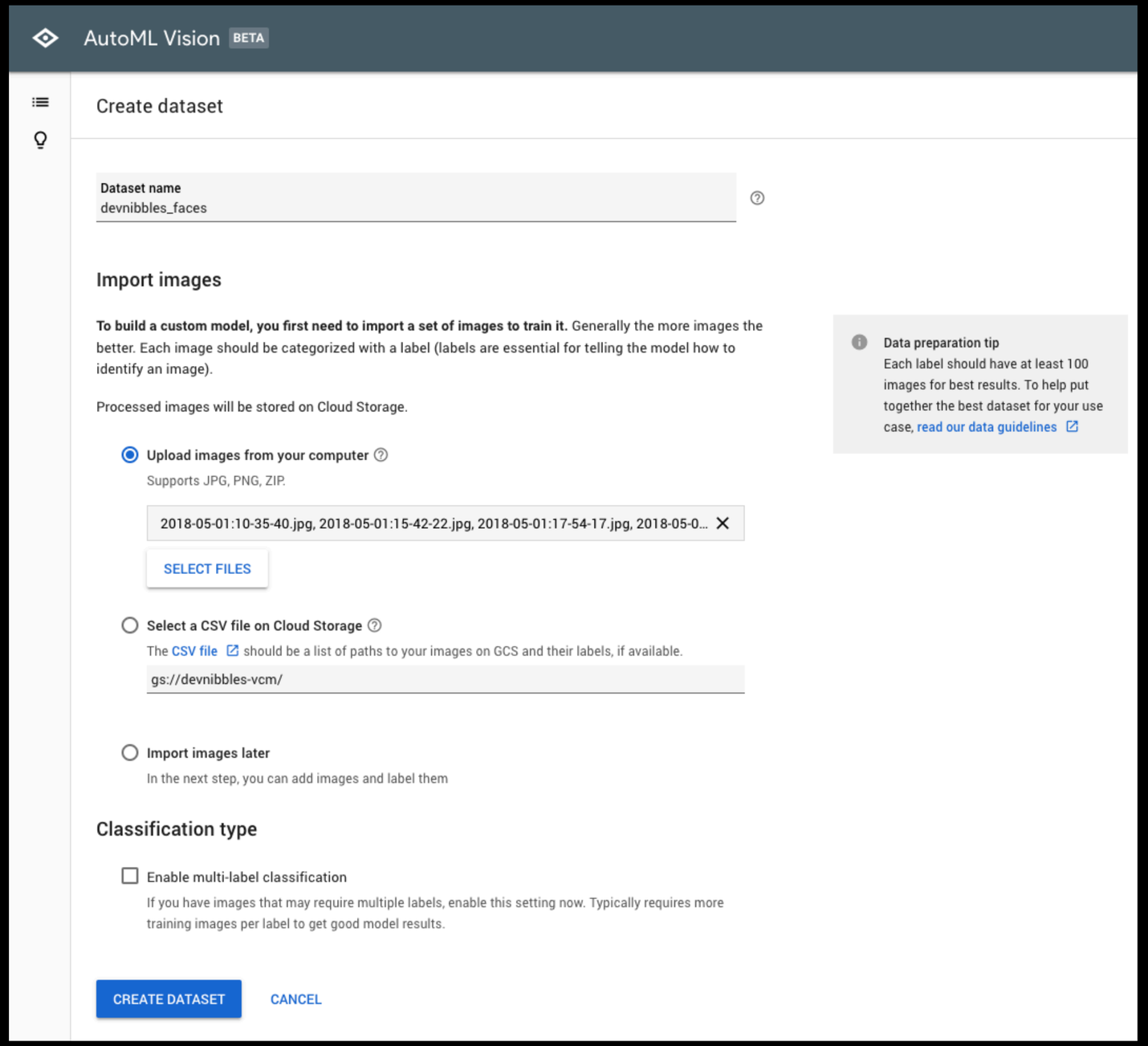
10 mins

Upload Images to Google Cloud AutoML

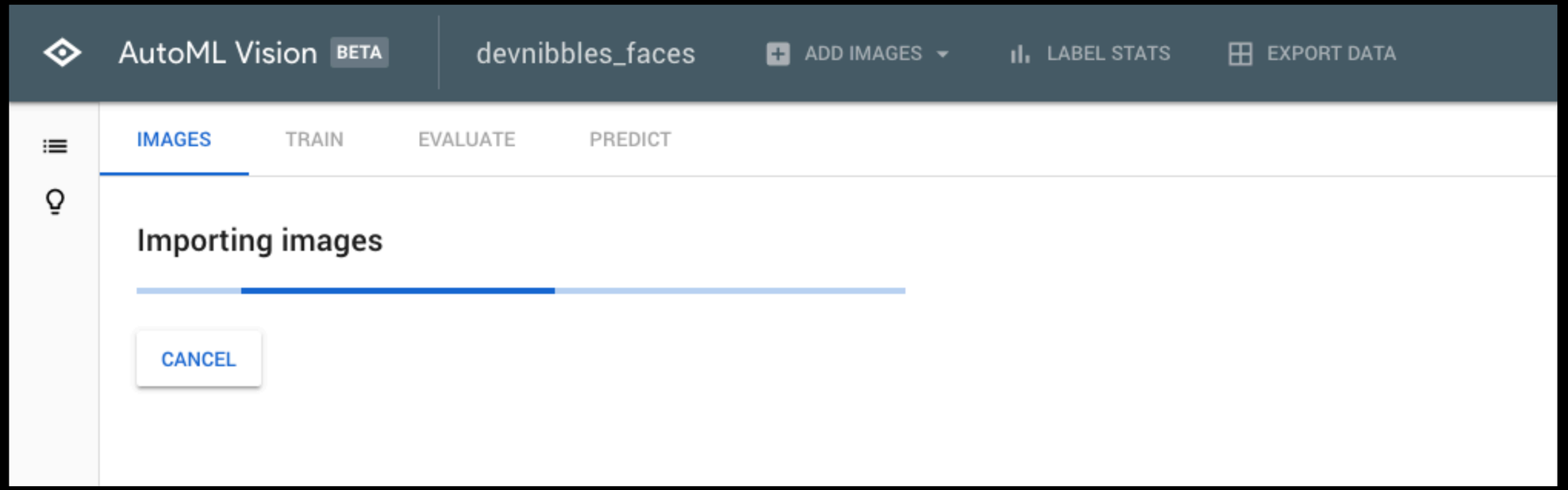
Create Dataset



https://cloud.google.com/automl/ui/vision/



If you have a large dataset, I'd recommend using the Cloud Storage option with a CSV of labels.



No need to watch this screen, go get a cup of tea, you'll be sent an email once the import is finished.

15 mins

Detect Faces in Android App

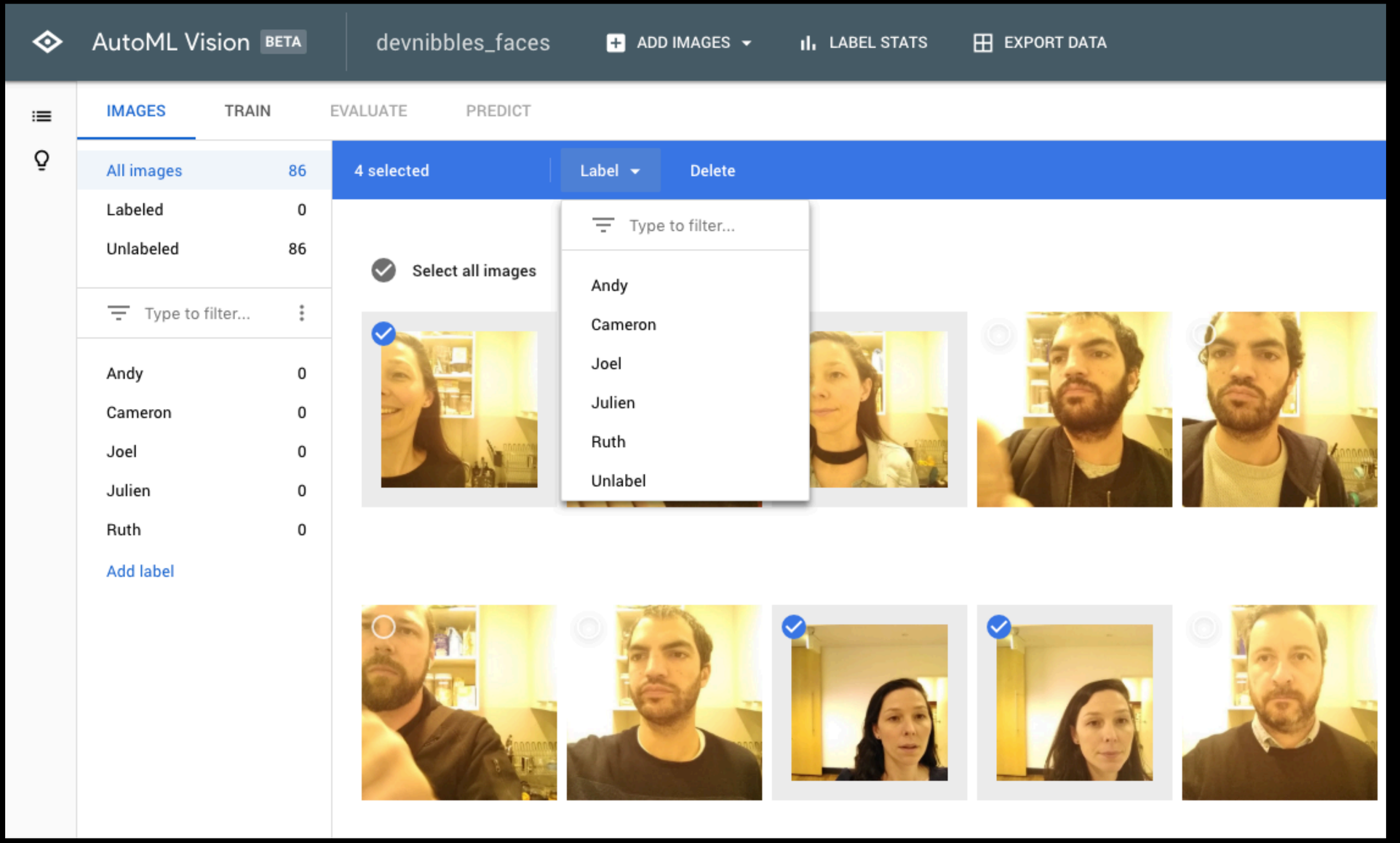
MLKitActivity.kt

```
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FaceDetector
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FaceGraphic
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FrameMetadata
import com.droidcon.boston2019.facialrecognition.detect.mlkit.MLCameraSource
import com.google.firebase.ml.vision.face.FirebaseVisionFace
import java.nio.ByteBuffer

    override fun createCameraSource() {
        mCameraSource = MLCameraSource(this, mGraphicOverlay)
        mCameraSource?.apply {
            setFrameDetector(
                FaceDetector(object : FaceDetector.DetectorCallback {
                    override fun onSuccess(
                        frameData: ByteBuffer,
                        results: List<FirebaseVisionFace>,
                        frameMetadata: FrameMetadata) {

                        if (results.isEmpty()) {
                            // No faces in frame, so clear frame of any previous faces.
                            mGraphicOverlay.clear()
                        } else {
                            // We have faces
                            results.forEach { face ->
                                val existingFace = mGraphicOverlay
                                    .find(face.trackingId) as FaceGraphic?

                                if (existingFace == null) {
                                    // A new face has been detected.
                                    val faceGraphic = FaceGraphic(
                                        face.trackingId,
                                        mGraphicOverlay
                                    )

                                    mGraphicOverlay.add(faceGraphic)

                                } else {
                                    // We have an existing face, update its position.
                                    existingFace.updateFace(face)
                                }
                            }

                            mGraphicOverlay.postInvalidate()

                        }
                    }

                    override fun onFailure(exception: Exception) {
                        exception.printStackTrace()
                    }
                })
            )
        }
    }
```

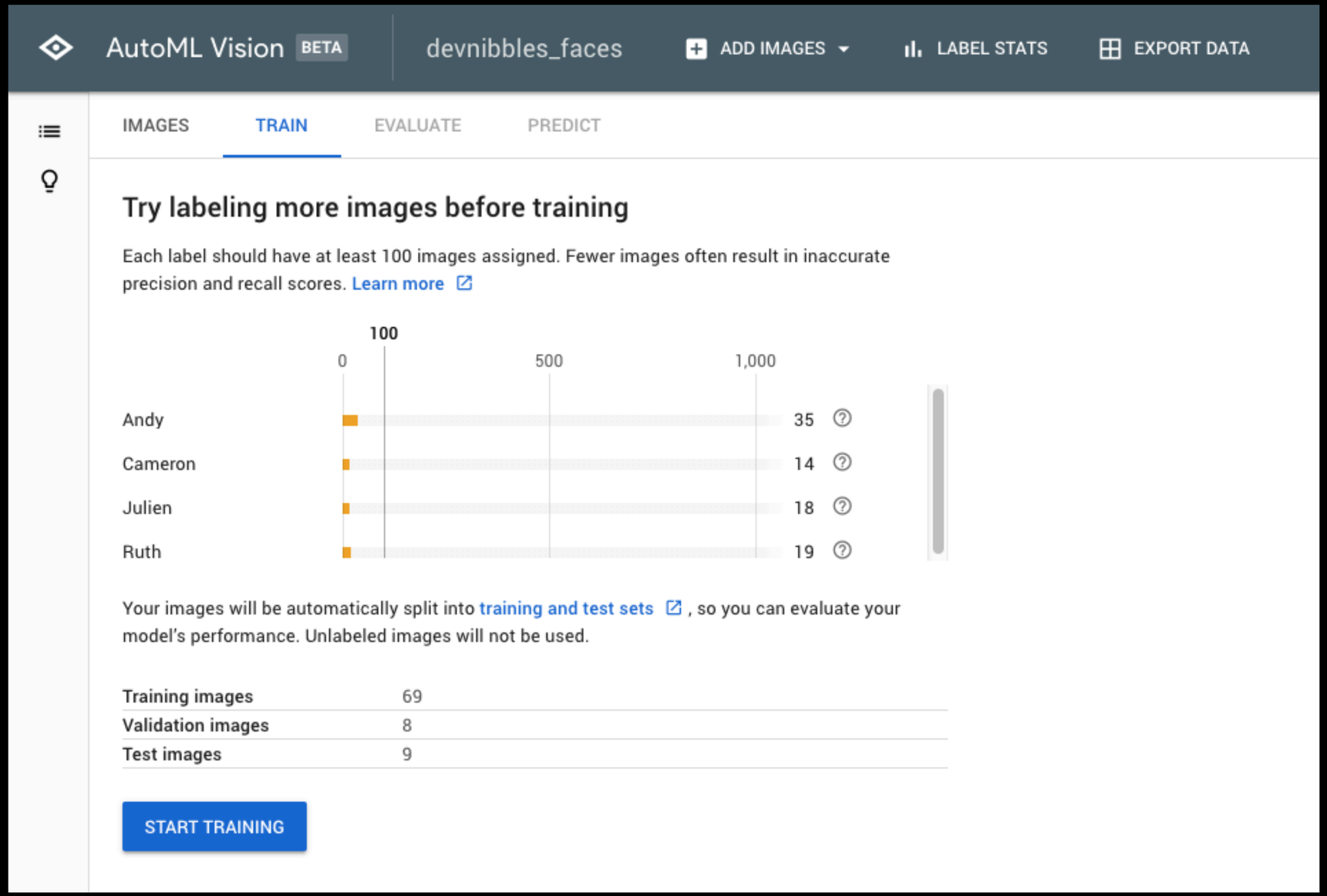
Enter this code and then run the app again.

15 mins

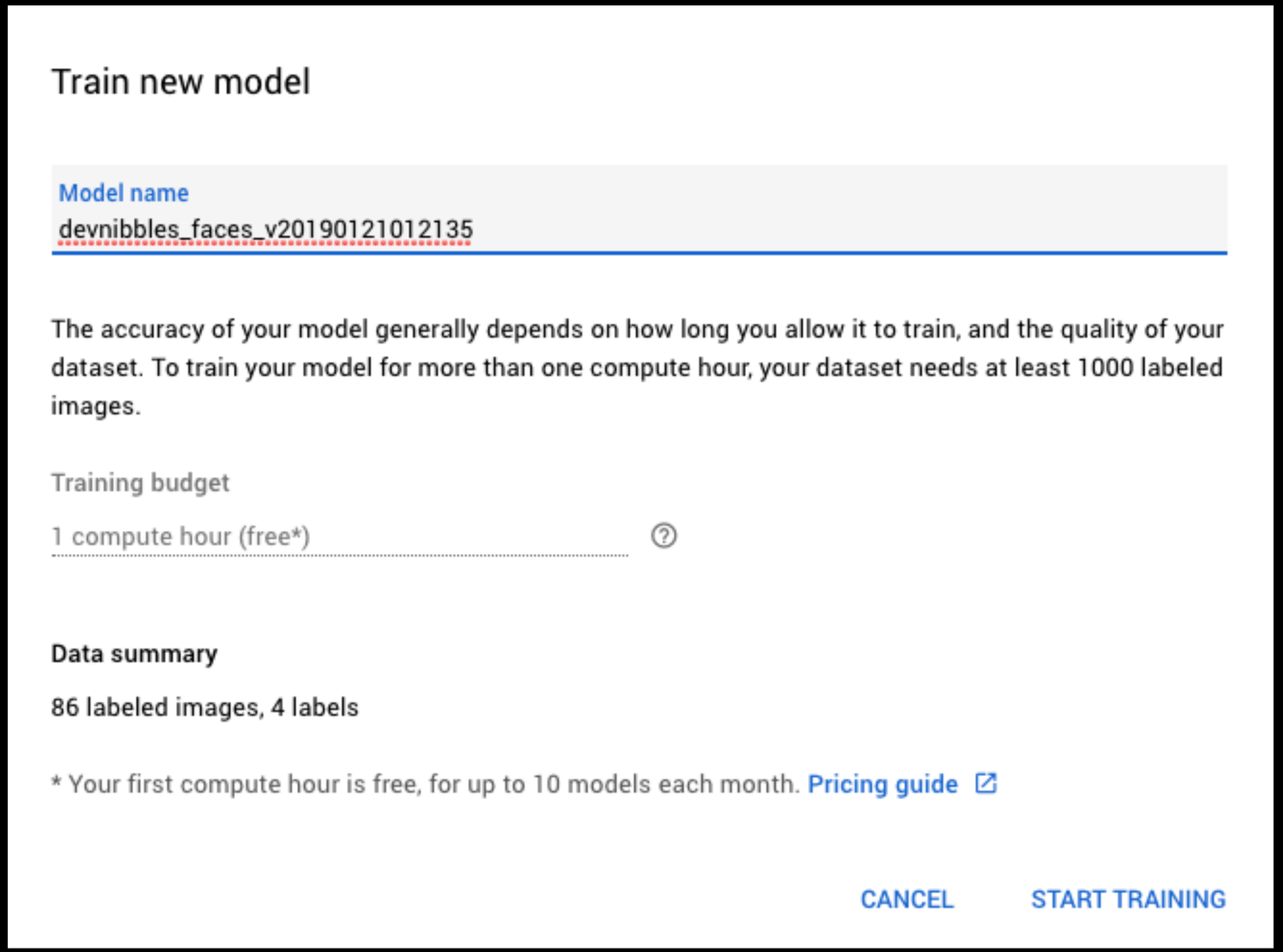
Train Facial Recognition Model



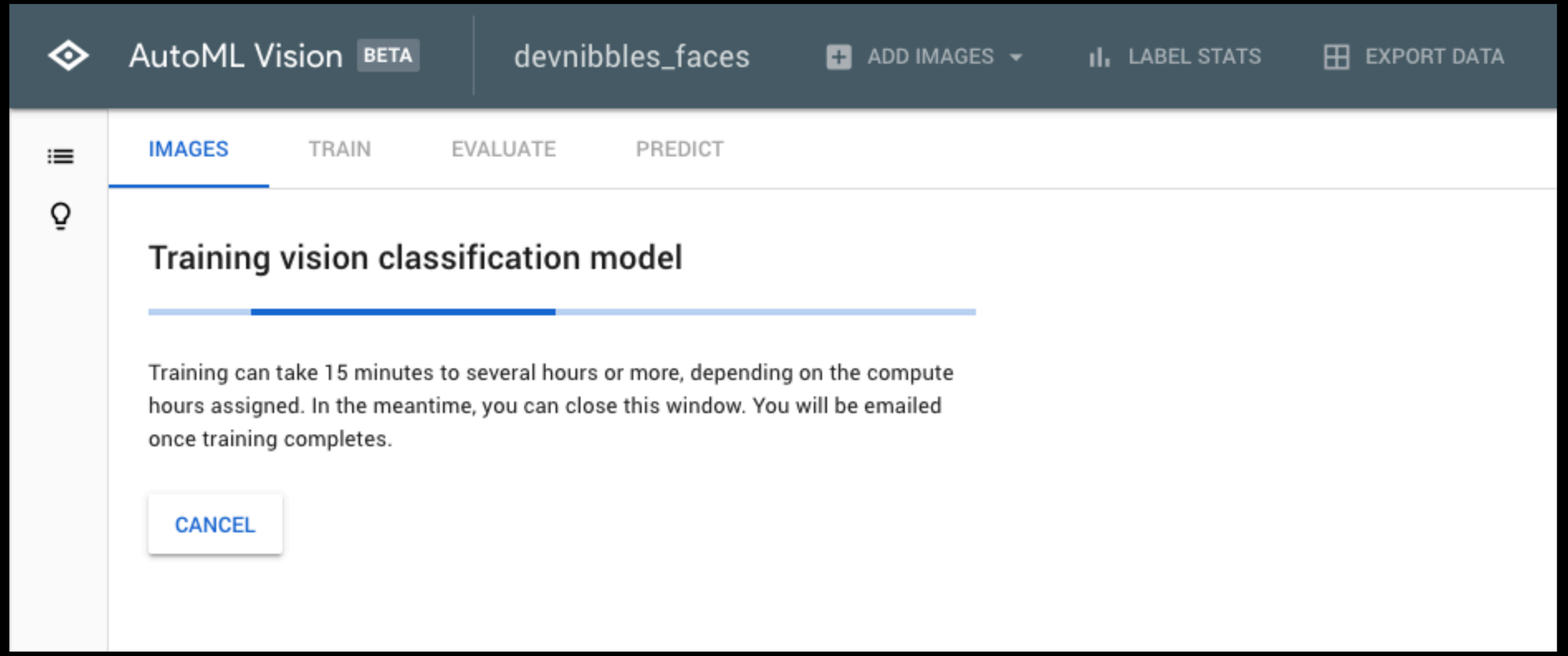
Create labels for each of the faces in your dataset using the “Add label” button, you can then apply labels on groups of images.



If your dataset is small, you'll be warned that you should import and label more images.



Train your model, small datasets should easily complete within the free hour, you'll need to enable Billing on your account though.



Go and get your second cup of tea, you'll receive an email once training is complete.

AutoML Vision BETA | devnibbles_faces | ADD IMAGES | LABEL STATS | EXPORT DATA

IMAGES | TRAIN | EVALUATE | PREDICT

Model

devnibbles_faces_v20190121012135

Created	Analyzed	Avg precision	Precision	Recall
1 compute hour	86 images 4 labels, 9 test images	1.0	100.0%	100.0%

Precision and recall are based on a score threshold of 0.5

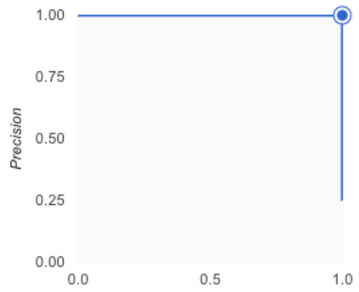
Type to filter labels...

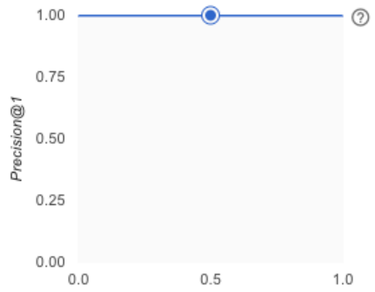
- All labels
- Andy
- Cameron
- Julien
- Ruth

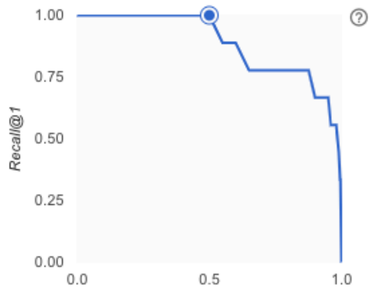
All labels

Score threshold	0.50
Total images	86
Precision	100.0%
Recall	100.0%

Use the slider to see which score threshold works best for your model on the precision-recall tradeoff curve. Learn more about these metrics and graphs



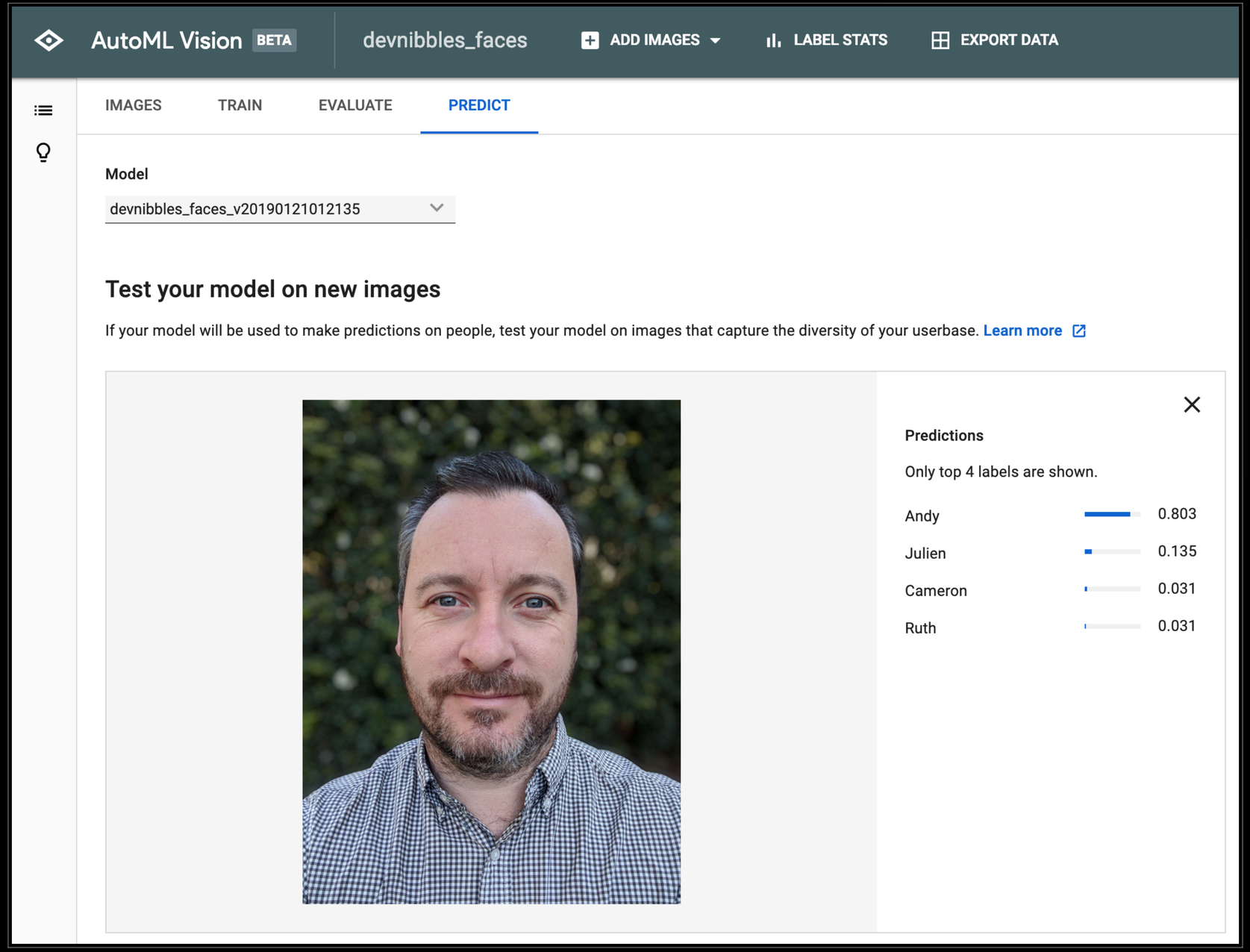




Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were m

True label \ Predicted label	Andy	Cameron	Julien	Ruth
Andy	100.0%	-	-	-
Cameron	-	100.0%	-	-
Julien	-	-	100.0%	-
Ruth	-	-	-	100.0%

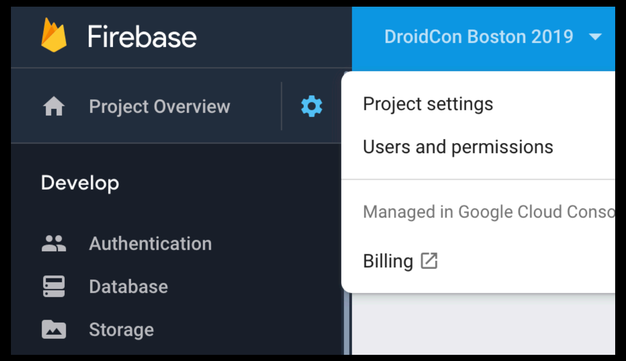


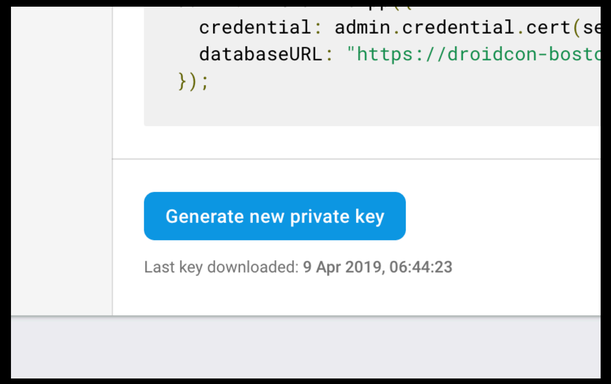
Your model is trained and ready to use, let's give it a quick test through the web interface.

20 mins

Classify Faces in Android App

CloudAutoMLViewModel.kt






```
private const val PROJECT = "droidcon-boston-2019"
private const val LOCATION = "us-central1"
private const val MODEL = "ICN1636466924499999978"
private const val SERVICE_ACCOUNT_JSON = "{\n" +
        "  \"type\": \"service_account\",\n" +
        "  \"project_id\": \"droidcon-boston-2019\",\n" +
        "  \"private_key_id\": \"7d0c363f95d2af49399999993f0ff9f3c615a36d\",\n"
        "  \"private_key\": \"\",\n" +
        "  \"client_email\": \"firebase-adminsdk@droidcon-boston-2019.iam.gservi
        "  \"client_id\": \"114199990381871398450\",\n" +
        "  \"auth_uri\": \"https://accounts.google.com/o/oauth2/auth\",\n" +
        "  \"token_uri\": \"https://oauth2.googleapis.com/token\",\n" +
        "  \"auth_provider_x509_cert_url\": \"https://www.googleapis.com/oauth2/
        "  \"client_x509_cert_url\": \"https://www.googleapis.com/robot/v1/metad
boston-2019.iam.gserviceaccount.com\"\n" +
        "}\n"
```

REST or SDK

```kotlin
private fun classifyUsingRetrofit(faceId: Int, imageBytes: ByteArray) {
    launch(errorHandler) {
        // Show loading indicator while we wait for the request.
        mResult.value = LoadingResource(FaceClassification(faceId, "", 0.0))

        // Build the body of our request, essentially the image to be classified.
        val body = CloudAutoMLModel(
            Payload(
                MlImage(
                    String(
                        Base64.encodeBase64(imageBytes)
                    )
                )
            )
        )

        // Define the authentication credentials and make the API request
        val response = getRESTService().classify(
            "Bearer ${accessToken?.tokenValue}",
            PROJECT, LOCATION, MODEL, body
        ).await()

        if (response.payload?.isNotEmpty() == true) {
            // We have a prediction!
            var predictedName: String? = null
            var predictedConfidence: Double? = null

            response.payload.forEach { entry ->
                if (entry.displayName != null) {
                    predictedName = entry.displayName
                    predictedConfidence = entry.classification?.score
                }
            }

            if (predictedName != null && predictedConfidence != null) {
                // We had an actual name returned
                mResult.postValue(
                    SuccessResource(
                        FaceClassification(
                            faceId,
                            predictedName!!,
                            predictedConfidence!!
                        )
                    )
                )
            } else {
                // No name was returned, this is an unknown face.
                mResult.postValue(ErrorResource(null))
            }
        } else {
            // There were no payloads returned, possible error or unknown face.
            mResult.postValue(ErrorResource(null))
        }
    }
}
```

```kotlin
private fun classifyUsingCloudSDK(faceId: Int, imageBytes: ByteArray) {
    launch(errorHandler) {
        // Show loading indicator while we wait for the request.
        mResult.value = LoadingResource(FaceClassification(faceId, "", 0.0))

        withContext(Dispatchers.IO) {
            // Define the authentication credentials
            val settings = PredictionServiceSettings.newBuilder()
                .setCredentialsProvider(FixedCredentialsProvider.create(mServiceCredentials)).build()

            val predictionServiceClient = PredictionServiceClient.create(settings)
            predictionServiceClient.use { client ->
                // Build the body of our request, essentially the image to be classified.
                val name = ModelName.of(PROJECT, LOCATION, MODEL)
                val image = Image.newBuilder().setImageBytes(ByteString.copyFrom(imageBytes)).build()
                val payload = ExamplePayload.newBuilder().setImage(image).build()
                val params = HashMap<String, String>()

                // Make the API request.
                val response = client.predict(name, payload, params)

                if (response.payloadCount > 0) {
                    // We have a prediction!
                    var predictedName: String? = null
                    var predictedConfidence: Double? = null

                    response.getPayload(0).allFields.entries.forEach { entry ->
                        if (entry.key.jsonName == "displayName") {
                            predictedName = entry.value as String
                        } else if (entry.key.jsonName == "classification") {
                            val classification = entry.value as ClassificationProto.ClassificationAnnotation
                            predictedConfidence= classification.score.toDouble()
                        }
                    }

                    if (predictedName != null && predictedConfidence != null) {
                        // We had an actual name returned
                        mResult.postValue(
                            SuccessResource(
                                FaceClassification(
                                    faceId,
                                    predictedName!!,
                                    predictedConfidence!!
                                )
                            )
                        )
                    } else {
                        // No name was returned, this is an unknown face.
                        mResult.postValue(ErrorResource(null))
                    }
                } else {
                    // There were no payloads returned, possible error or unknown face.
                    mResult.postValue(ErrorResource(null))
                }
            }
        }
    }
}
```


07:55
95%
DroidCon Boston ML-Kit
Andy (0.99)

Thank You

Andrew Kelly - Google Developer Expert for Android

@andrew_ke11y

https://medium.com/devnibbles