With CameraX, developing camera based Android views is a more streamlined process with some standout features as compared to its predecessor (Camera2):

* Abstracts away the complexities of differences in OEM sensor hardware drivers.
* Supports Android API 21 and up.
* As with other Jetpack components, it is lifecycle aware, and takes care of bringing up and breaking down the required resources in response to app lifecycle events.
* **To me, the most exciting feature is the introduction of what can be called the “Use Case Pipeline”**

Base Use Cases

As of now, the CameraX API has three distinct base Use Cases, namely:

Preview: accepts a surface for displaying a preview — Preview

Image analysis: provides CPU-accessible buffers for analysis, such as for machine learning inference :) — ImageAnalysis

Image capture: captures and saves a photo — ImageCapture

It is safe to assume most of the time that thePreview Use Case will be placed in the pipeline. This will take the received image frame and render it to a Surface (androidx.camera.view.PreviewView) for the user to see.

Similarly, ImageCapture provides the functionality to save the received frame as a photo.

The most exciting is obviouslyImageAnalyses — this gives you the raw framebuffer to run inference on with an ML model of your choosing.

For the purpose of this article, I will use the BarcodeScanning API (available as part of Google’s MLKit framework) to run inference on the image frame to recognize and decode different barcode standards found in the image