This tutorial is designed to let you quickly start exploring and developing applications with the Video Intelligence API. It is designed for people familiar with basic programming, though even without much programming knowledge, you should be able to follow along. Having walked through this tutorial, you should be able to use the Reference documentation (/video-intelligence/docs/reference/rest/) to create your own basic applications.

This tutorial steps through a Video Intelligence API application using Python code. The purpose here is not to explain the Python client libraries, but to explain how to make calls to the Video Intelligence API. Applications in Java and Node.js are essentially similar.

If you're looking for a code-only example or an example in another language, check out the companion how-to guide (/video-intelligence/docs/analyze-labels).

This tutorial has several prerequisites:

- You've set up a Cloud Video Intelligence API project (/video-intelligence/docs/before-you-begin) in the Google Cloud Console.
- You've set up your environment using a service account and Application Default Credentials (/video-intelligence/docs/common/auth#set_up_a_service_account).
- You have basic familiarity with Python (https://www.python.org/) programming.
- Set up your Python development environment. It is recommended that you have the latest version of Python, `pip`, and `virtualenv` installed on your system. For instructions, refer to the Python Development Environment Setup Guide (/python/setup) for Google Cloud Platform.
- You've installed the Google Cloud client library (https://github.com/GoogleCloudPlatform/google-cloud-python/tree/master/videointelligence)

This tutorial walks you through a basic Video API application, using a `LABEL_DETECTION` request. A `LABEL_DETECTION` request annotates a video with labels (or "tags") that are selected based on the image content. For example, a video of a train at a crossing may produce labels such as "train", "transportation", "railroad crossing", etc.

We'll show the entire code first. Note that we have removed most comments from this code in order to show you how brief it is. We'll provide more comments as we walk through the code.

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

This simple application performs the following tasks:

- Imports the libraries necessary to run the application
- Takes a video file stored in Google Cloud Storage URI as an argument and passes it to the `main()` function
- Gets credentials to run the Video Intelligence API service
- Creates a video annotation request to send to the video service
- Sends the request and returns a long running operation
- Loops over the long running operation until the video is processed and return values are available
- Parses the response for the service and displays it to the user

We'll go over these steps in more detail below.

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

We import some standard libraries: `argparse` to allow the application to accept input filenames as arguments and `sys` for formatting output while waiting for API responses. We also import `time` to run some simple wait loops.

For using the Cloud Video Intelligence API, we'll also want to import the `google.cloud.videointelligence_v1` and its enumeration class, which holds the directory of our API calls.

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

Here, we simply parse the passed argument for the Google Cloud Storage URI of the video filename and pass it to the `main()` function.

Before communicating with the Video Intelligence API service, you need to authenticate your service using previously acquired credentials. Within an application, the simplest way to obtain credentials is to use Application Default Credentials (https://developers.google.com/identity/protocols/application-default-credentials) (ADC). By default, ADC will attempt to obtain credentials from the `GOOGLE_APPLICATION_CREDENTIALS` environment file, which should be set to point to your service account's JSON key file. (You should have set up your service account and environment to use ADC in the Quickstart (/video-intelligence/docs/getting-started).

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

Now that our Video Intelligence API service is ready, we can construct a request to that service. Requests to the Video Intelligence API are provided as JSON objects. See the Video Intelligence API Reference (/video-intelligence/docs/reference/rest) for complete information on the specific structure of such a request.

This code snippet performs the following tasks:

1. Constructs the JSON for a POST request to the `annotate_video()` method.
2. Injects the Google Cloud Storage location of our passed video filename into the request.
3. Indicates that the `annotate` method should perform `LABEL_DETECTION`.

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

Using the existing operation request for our existing operation, we construct a `while` loop to periodically check the state of that operation. Once our operation has indicated that the operation is `done`, we break out of the loop and can parse the response.

video/cloud-client/labels/labels.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/video/cloud-client/labels/labels.py)

Once the operation has been completed, our response will contain result within an AnnotateVideoResponse (/video-intelligence/docs/reference/rest/Shared.Types/AnnotateVideoResponse), which consists of a list of `annotationResults`, one for each video sent in the request. Because we sent only one video in the request, we take the first `segmentLabelAnnotations` of the results. We then loop through all the labels in `segmentLabelAnnotations`. For the purpose of this tutorial, we only display video-level annotations. To identify video-level annotations, we pull `segment_label_annotations` data from the results. Each segment label annotation includes a description (`segment_label.description`), a list of entity categories (`category_entity.description`) and where they occur in segments by start and end time offsets from the beginning of the video.

Because we sent only one video in the request, we will take the first `description` of the first result and print that `description`.

To run your application, simply pass it the Google Cloud Storage URI of a video:

Congratulations! You've performed an annotation task using the Video Intelligence API!