4. Create a scalable serverless architecture to detect objects from the images immediately after being uploaded to a GCS bucket. The extracted information about the objects should be logged [optional: stored to datastore database]. The designed architecture should also have error handling.

1. Create Cloud Storage bucket.
2. Create a Cloud Function with **trigger** as Cloud Storage and Event Type as Finalize/Create and then specify your bucket.
3. Set the runtime as Python 3.7 and add the following code:

```
def is_img(img_type):
    formats = ("image/jpeg", "image/png", "image/gif", "image/bmp", "image/webp", "image/x-icon", "application/pdf", "image/tiff")
    if img_type in formats:
        return True
    else:
        return False

def localize_objects_uri(uri):
    """Localize objects in the image on Google Cloud Storage

    Args:
    uri: The path to the file in Google Cloud Storage (gs://...)
    """
    from google.cloud import vision

    client = vision.ImageAnnotatorClient()

    image = vision.types.Image()
    image.source.image_uri = uri

    objects = client.object_localization(
        image=image).localized_object_annotations

    print('Number of objects found: {}'.format(len(objects)))
    for object_ in objects:
        print('\n{} (confidence: {})'.format(object_.name, object_.score))
        print('Normalized bounding polygon vertices: ')
        for vertex in object_.bounding_poly.normalized_vertices:
            print(' - ({}, {})'.format(vertex.x, vertex.y))

def hello_gcs(event, context):
    """Triggered by a change to a Cloud Storage bucket.
    Args:
         event (dict): Event payload.
         context (google.cloud.functions.Context): Metadata for the event.
    """
    #cloud_logger.setLevel(logging.INFO)

    file = event
    #print(file['name'])

    # checking for image
    img_type = file["contentType"]
    if is_img(img_type):
            path = "gs://av-a2-q4/" + file["name"]
            localize_objects_uri(path)
    else:
        print("unsupported file type")
```

4. Test the function by uploading an image in your bucket.
5. Now, to store the output in a Datastore database, modify your code as shown below:

```
from google.cloud import datastore

def is_img(img_type):
    # formats supported by cloud vision api
    formats = ("image/jpeg", "image/png", "image/gif", "image/bmp", "image/webp", "image/x-
icon", "application/pdf", "image/tiff")
    if img_type in formats:
        return True
    else:
        return False

def to_datastore(entity_dict):
    ds_client = datastore.Client()
    entity = datastore.Entity(key=ds_client.key('av_a2_q4 id:5706285722894336'))
    entity.update(entity_dict)
    ds_client.put(entity)

def localize_objects_uri(uri):
    """Localize objects in the image on Google Cloud Storage

    Args:
    uri: The path to the file in Google Cloud Storage (gs://...)
    """
    from google.cloud import vision

    client = vision.ImageAnnotatorClient()

    image = vision.types.Image()
    image.source.image_uri = uri

    objects = client.object_localization(
        image=image).localized_object_annotations

    print('Number of objects found: {}'.format(len(objects)))
    for object_ in objects:
        entity_dict = {
            'obj_name': object_.name,
            'obj_score': object_.score,
            }
        print('\n{} (confidence: {})'.format(object_.name, object_.score))
        print('Normalized bounding polygon vertices: ')
        i = 1
        for vertex in object_.bounding_poly.normalized_vertices:
            entity_dict['x_' + str(i)] = vertex.x
            entity_dict['y_' + str(i)] = vertex.y
            i+=1
            print(' - ({}, {})'.format(vertex.x, vertex.y))
        to_datastore(entity_dict)

def hello_gcs(event, context):
    """Triggered by a change to a Cloud Storage bucket.
    Args:
         event (dict): Event payload.
         context (google.cloud.functions.Context): Metadata for the event.
    """
    #cloud_logger.setLevel(logging.INFO)

    file = event
    #print(file['name'])

    # checking for image
    img_type = file["contentType"]
```

```
if is_img(img_type):
        path = "gs://av-a2-q4/" + file["name"]
        localize_objects_uri(path)
else:
   print("unsupported file type")
```