TASK 4.1 - P

Computer Vision

Detecting objects with Coordinates using Azure computer vision

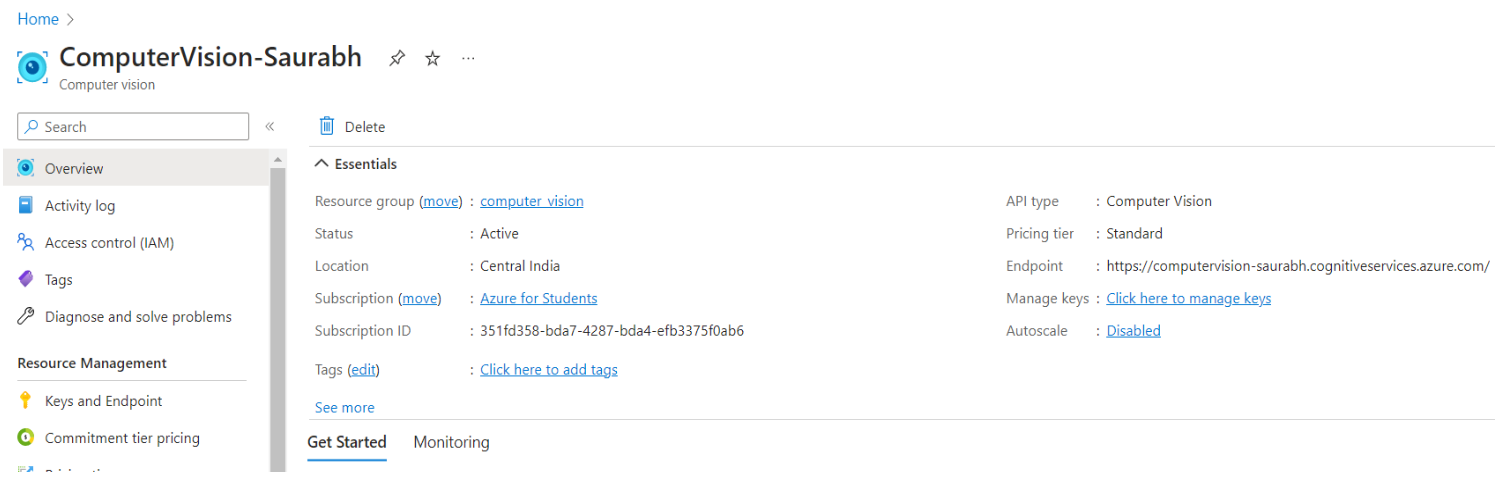
Saurabh Dharmadhikari
Saurabh.dharma01@gmail.com

Table of Content:

1. **Please explain cell by cell of your code from reading a local image to object detection, drawing a bounding box around different object. To complete this task, you need to provide the screenshot of your code and explain cell by cell of the code and explain what sort of API is being used.**

We first create a computer vision resource on Azure.

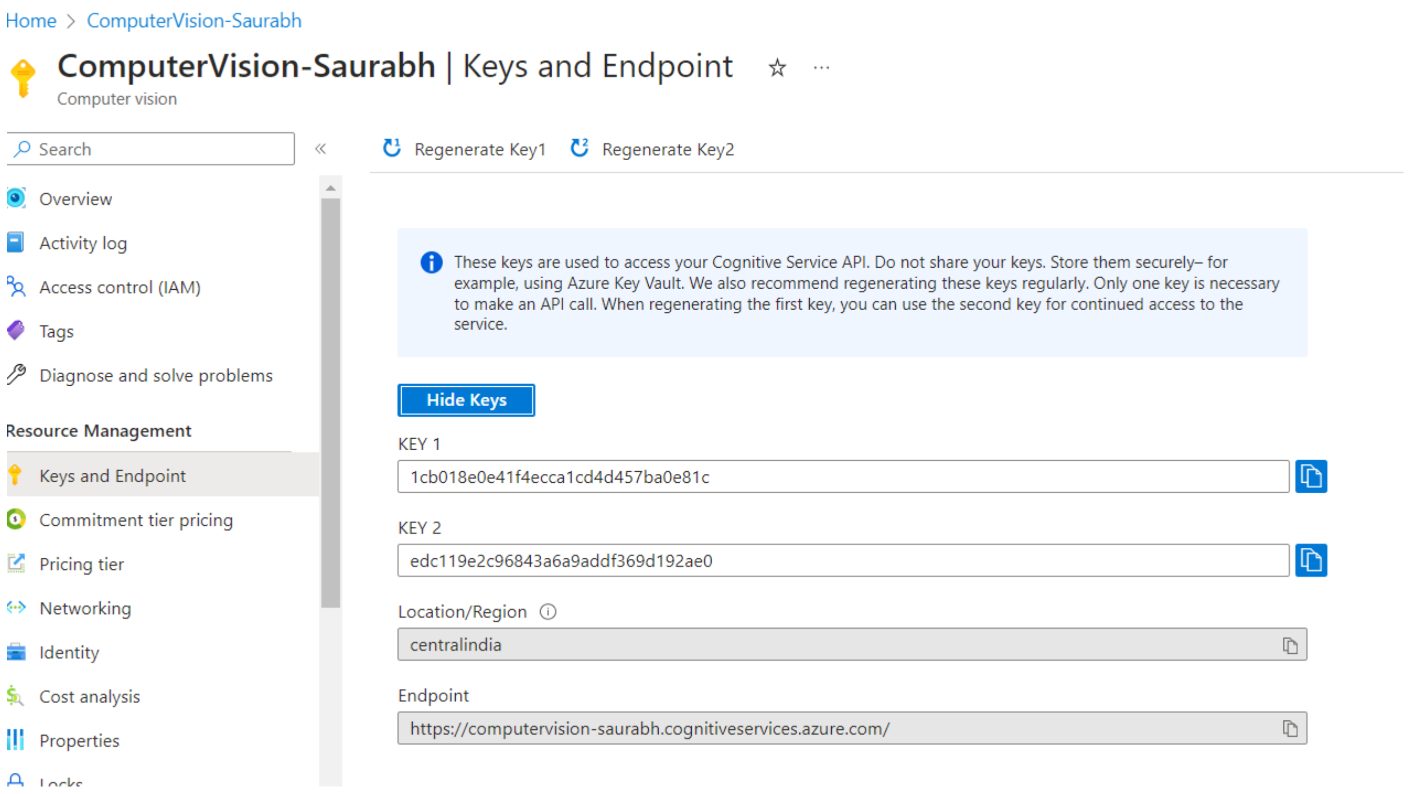
Below is the overview of our resource. Location is central India and other details.



Resource group (move) : computer vision
Status : Active
Location : Central India
Subscription (move) : Azure for Students
Subscription ID : 351fd358-bda7-4287-bda4-efb3375f0ab6
Tags (edit) : Click here to add tags

API type : Computer Vision
Pricing tier : Standard
Endpoint : https://computervision-saurabh.cognitiveservices.azure.com/
Manage keys : Click here to manage keys
Autoscale : Disabled

We can find keys from the left side of the screen in Keys and Endpoint.



These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

KEY 1
1cb018e0e41f4ecca1cd4d457ba0e81c

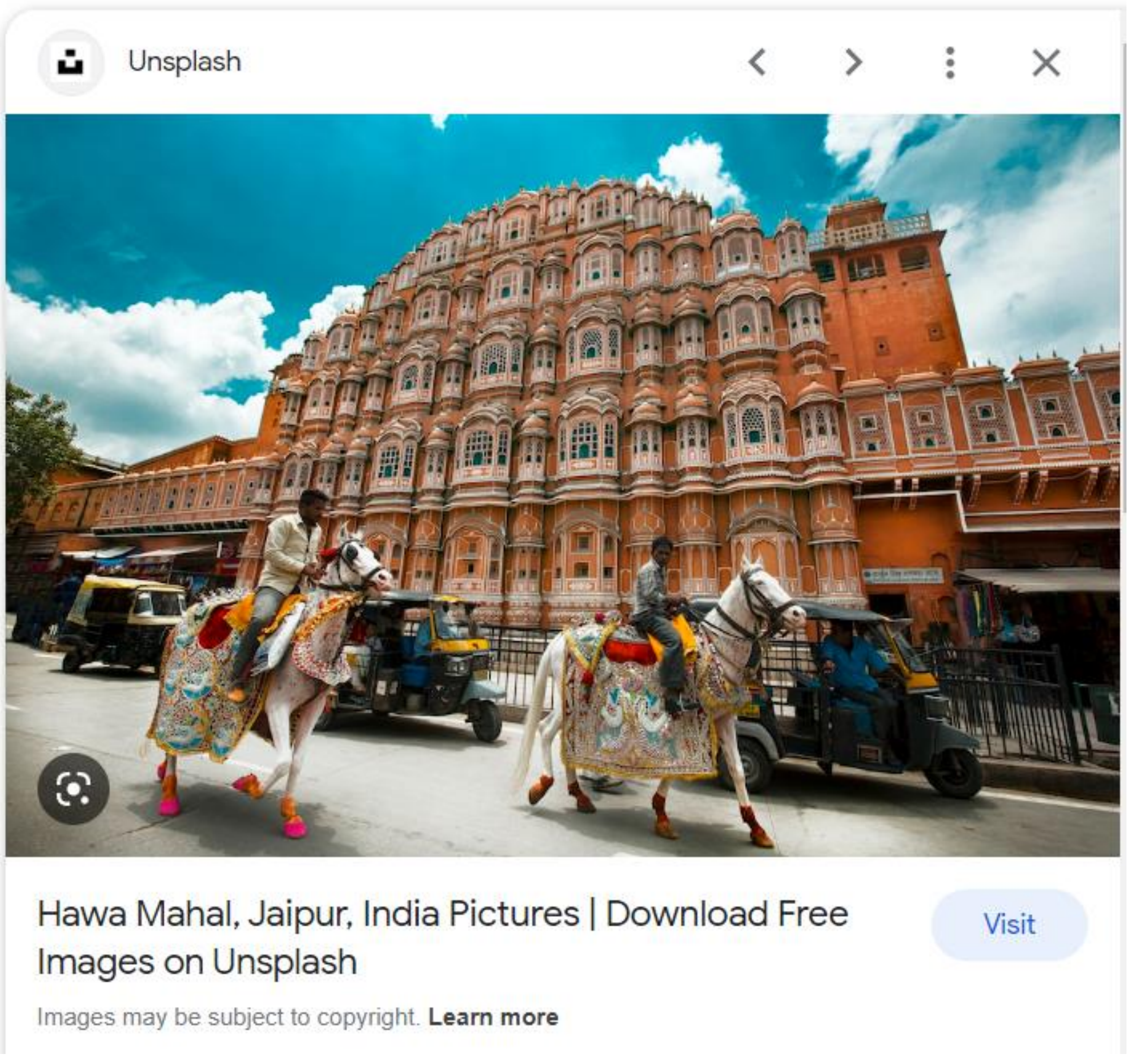
KEY 2
edc119e2c96843a6a9addf369d192ae0

Location/Region
centralindia

Endpoint
https://computervision-saurabh.cognitiveservices.azure.com/

We have created computer vision on azure to use these cognitive services using key and endpoint to later use it in python.

Below is the image that we are going to use in computer vision to detect, describe and find tags.



Hawa Mahal, Jaipur, India Pictures | Download Free Images on Unsplash

Images may be subject to copyright. **Learn more**

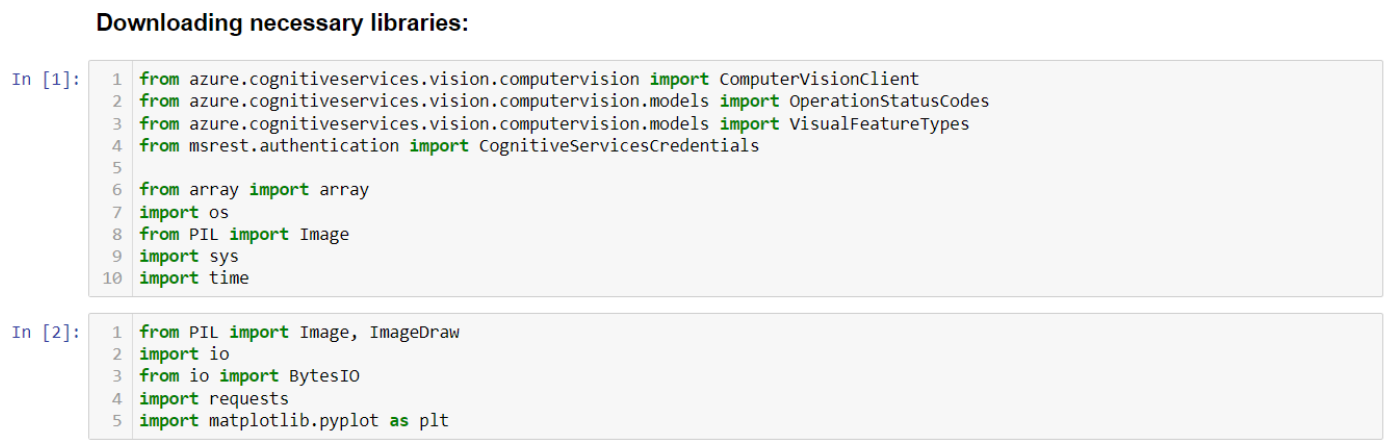
We are using a photograph by Siva (2016).

Web address from where the image address was taken¶

https://www.google.com/search?q=unsplash+hawa+mahal&tbm=isch&ved=2ahUKEwiXxL_I3bH-AhXt-3MBHQoWBT0Q2-cCegQIABAA&oq=unsplash+hawa+mahal&gs_lcp=CgNpbWcQAzoECCMQJzoICAAQgAQQsQM6BQgAEIAEOggIABCxAxCDAToECAAQAzoHCAAQigUQQzoKCAAQigUQsQMQQzoLCAAQgAQQsQMQgwE6BwgAEBgQgARQjGRYpbUBYIC4AWgAcAB4AIAB7QWIAfQgkgEOMC4xNy4wLjEuMi4wLjGYAQCgAQGgAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=zqg9ZJelOu33z7sPiqyU6AM&bih=746&biw=1536&rlz=1C1UEAD_enIN996IN996#imgrc=6mhniEGQhG9JHM

Courtesy Unsplash images.

Importing libraries:



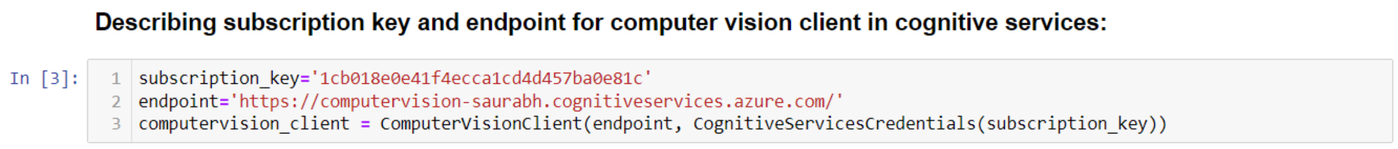
Downloading necessary libraries:

```
In [1]:
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from azure.cognitiveservices.vision.computervision.models import OperationStatusCodes
from azure.cognitiveservices.vision.computervision.models import VisualFeatureTypes
from msrest.authentication import CognitiveServicesCredentials

from array import array
import os
from PIL import Image
import sys
import time
```

```
In [2]:
from PIL import Image, ImageDraw
import io
from io import BytesIO
import requests
import matplotlib.pyplot as plt
```

We first import all the necessary libraries on jupyter notebook needed for computer vision. This includes CommputerVisionClient, OperatioStatusCodes, CognitiveServiceCredentials from azure.

Also, Image from PIL.



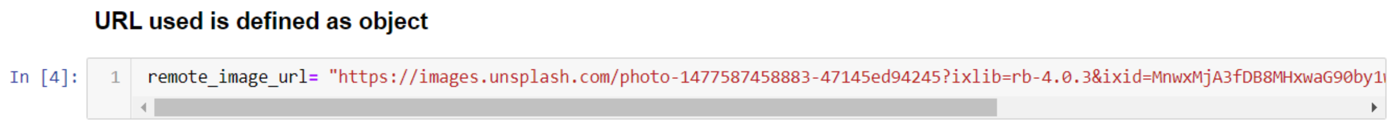
Describing subscription key and endpoint for computer vision client in cognitive services:

```
In [3]:
subscription_key='1cb018e0e41f4ecca1cd4d457ba0e81c'
endpoint='https://computervision-saurabh.cognitiveservices.azure.com/'
computervision_client = ComputerVisionClient(endpoint, CognitiveServicesCredentials(subscription_key))
```

We create an object subscription_key to pass KEY1 that we created in Azure for computer vision resource.

Similarly, we pass endpoint details created in azure computer vision under object endpoint as shown above.

We connect to azure using endpoint and KEY.



URL used is defined as object

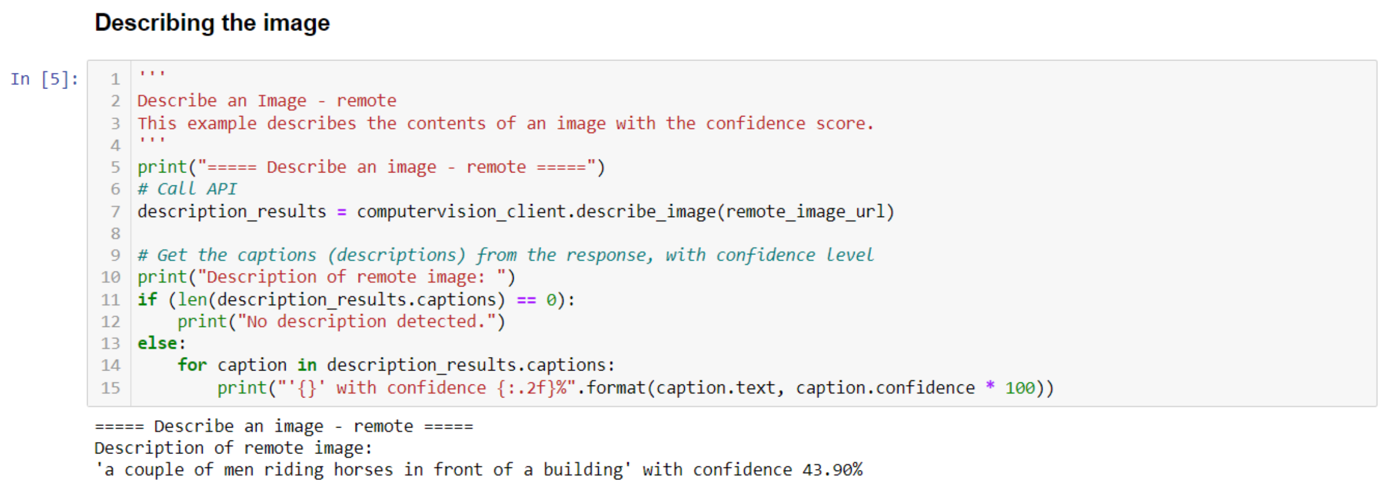
```
In [4]:
remote_image_url= "https://images.unsplash.com/photo-1477587458883-47145ed94245?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1
```

Image address

https://images.unsplash.com/photo-1477587458883-47145ed94245?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8M3x8aGF3YSUyMG1haGFsJTJDJTIwamFpcHVyJTJDJTIwaW5kaWF8ZW58MHx8MHx8&w=1000&q=80

We create an object which has a web address of an image.

Describing the image:



Describing the image

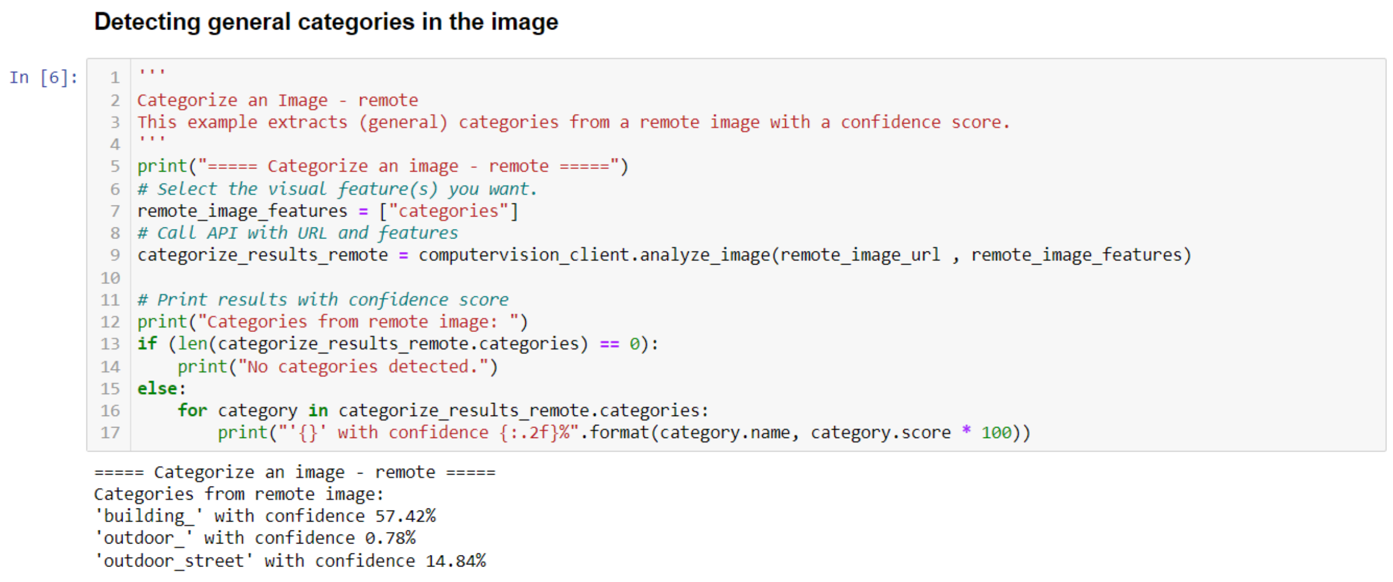
```
In [5]:
'''
Describe an Image - remote
This example describes the contents of an image with the confidence score.
'''
print("===== Describe an image - remote =====")
# Call API
description_results = computervision_client.describe_image(remote_image_url)

# Get the captions (descriptions) from the response, with confidence level
print("Description of remote image: ")
if (len(description_results.captions) == 0):
    print("No description detected.")
else:
    for caption in description_results.captions:
        print("'{}' with confidence {:.2f}%".format(caption.text, caption.confidence * 100))
```

```
===== Describe an image - remote =====
Description of remote image:
'a couple of men riding horses in front of a building' with confidence 43.90%
```

The above code is for describing the image. Thus, azure describes the image as a couple of men riding horses in front of a building' with confidence 43.90%’. Which means it rightly describes that men are riding horses in front of a building with a confidence of 43.90% but there is a lot more happening in the picture like tuk tuk rikshaws are also on the street. Azure cognitive services chose to describe the most relevant event.

Confidence is not very high for image description.

Categories in the image:



Detecting general categories in the image

```
In [6]:
'''
Categorize an Image - remote
This example extracts (general) categories from a remote image with a confidence score.
'''
print("===== Categorize an image - remote =====")
# Select the visual feature(s) you want.
remote_image_features = ["categories"]
# Call API with URL and features
categorize_results_remote = computervision_client.analyze_image(remote_image_url , remote_image_features)

# Print results with confidence score
print("Categories from remote image: ")
if (len(categorize_results_remote.categories) == 0):
    print("No categories detected.")
else:
    for category in categorize_results_remote.categories:
        print("'{}' with confidence {:.2f}%".format(category.name, category.score * 100))
```

```
===== Categorize an image - remote =====
Categories from remote image:
'building_' with confidence 57.42%
'outdoor_' with confidence 0.78%
'outdoor_street' with confidence 14.84%
```

In the above code we are categorizing this image from the web. So we use analyse this image.

Categories and confidence:

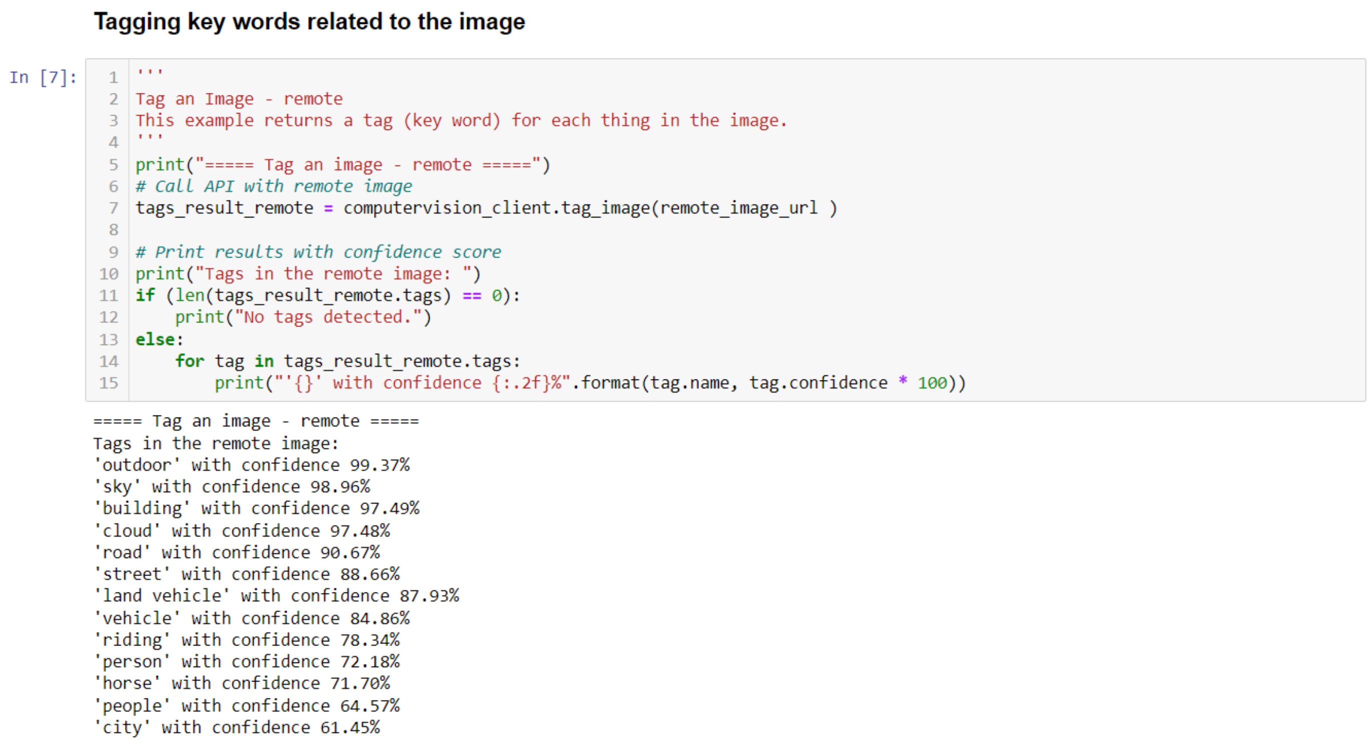
Building: 57.42%

Outdoor: 0.78%

Outdoor street: 14.84%

We observe that building has a good confidence level but outdoor has a very little confidence level.

Let us see how many tags can be associated with this image:



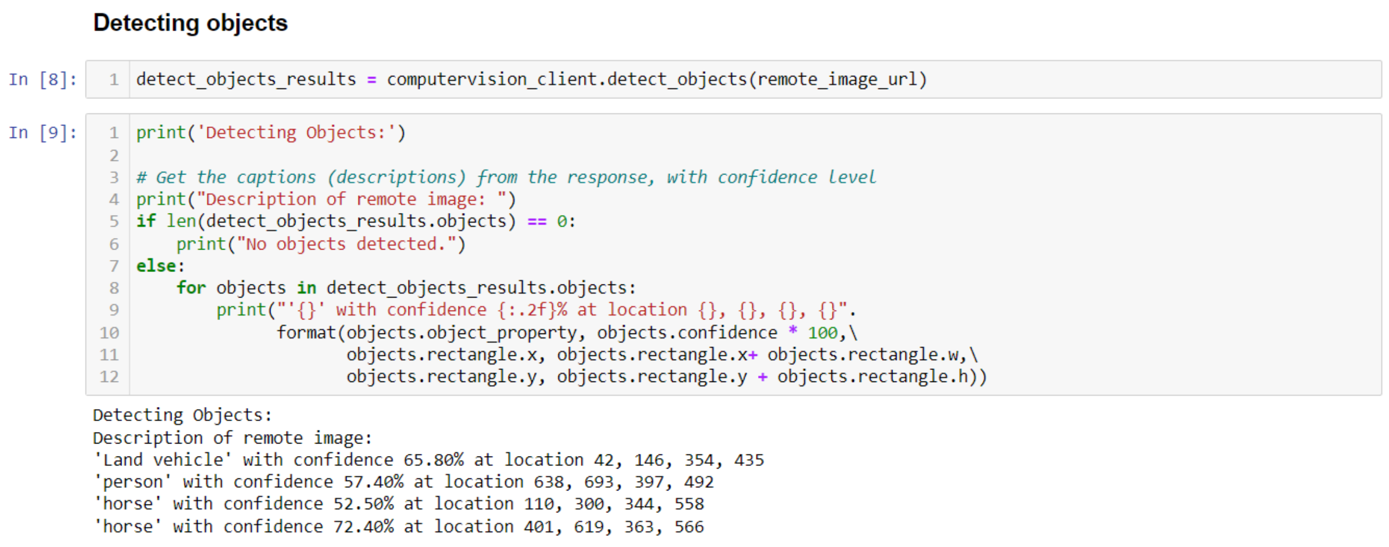
Tagging key words related to the image

In [7]:
```python
'''
Tag an Image - remote
This example returns a tag (key word) for each thing in the image.
'''
print("===== Tag an image - remote =====")
# Call API with remote image
tags_result_remote = computervision_client.tag_image(remote_image_url )

# Print results with confidence score
print("Tags in the remote image: ")
if (len(tags_result_remote.tags) == 0):
    print("No tags detected.")
else:
    for tag in tags_result_remote.tags:
        print("'{}' with confidence {:.2f}%".format(tag.name, tag.confidence * 100))
```

```
===== Tag an image - remote =====
Tags in the remote image:
'outdoor' with confidence 99.37%
'sky' with confidence 98.96%
'building' with confidence 97.49%
'cloud' with confidence 97.48%
'road' with confidence 90.67%
'street' with confidence 88.66%
'land vehicle' with confidence 87.93%
'vehicle' with confidence 84.86%
'riding' with confidence 78.34%
'person' with confidence 72.18%
'horse' with confidence 71.70%
'people' with confidence 64.57%
'city' with confidence 61.45%
```

We see that outdoor has been associated by azure with the highest confidence of 99.56% followed by sky and building. Lowest confidence for tag given by azure are city and people with confidence of 61.45% and 64.57% respectively.

Detecting Objects:



Detecting objects

In [8]:
```python
detect_objects_results = computervision_client.detect_objects(remote_image_url)
```

In [9]:
```python
print('Detecting Objects:')

# Get the captions (descriptions) from the response, with confidence level
print("Description of remote image: ")
if len(detect_objects_results.objects) == 0:
    print("No objects detected.")
else:
    for objects in detect_objects_results.objects:
        print("'{}' with confidence {:.2f}% at location {}, {}, {}, {}".
              format(objects.object_property, objects.confidence * 100,\
                     objects.rectangle.x, objects.rectangle.x+ objects.rectangle.w,\
                     objects.rectangle.y, objects.rectangle.y + objects.rectangle.h))
```

```
Detecting Objects:
Description of remote image:
'Land vehicle' with confidence 65.80% at location 42, 146, 354, 435
'person' with confidence 57.40% at location 638, 693, 397, 492
'horse' with confidence 52.50% at location 110, 300, 344, 558
'horse' with confidence 72.40% at location 401, 619, 363, 566
```

We can see above objects detected with confidence and their coordinates.

Describing URL to get image_data from

In [12]:
```
response = requests.get("https://images.unsplash.com/photo-1477587458883-47145ed94245?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG
image_data = response.content
```

Image address

https://images.unsplash.com/photo-1477587458883-47145ed94245?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8M3x8aGF3YSUyMG1haGFsJTJDJTIwamFpcHVyJTJDJTIwaW5kaWF8ZW58MHx8MHx8&w=1000&q=80

Creating binary stream and storing image data using PIL lib.

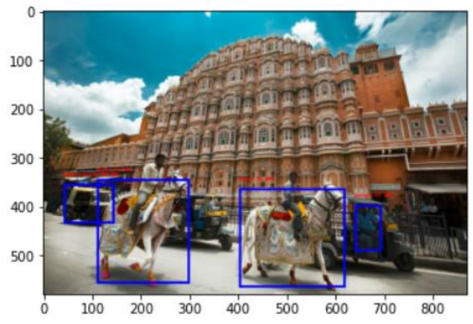
In [13]:
```
image = Image.open(io.BytesIO(image_data))
```

We created binary stream using PIL library and stored data as in image.

Detected Objects:

Drawing rectangles and text over objects detected

In [14]:
```
draw = ImageDraw.Draw(image)
for obj in detected_object:
    x,y = obj.rectangle.x, obj.rectangle.y
    w,h = obj.rectangle.w, obj.rectangle.h
    draw.rectangle([x, y, x+w, y+h], outline = 'blue', width = 5)
    text = '{} {:.2f}%'.format(obj.object_property, obj.confidence * 100)
    draw.text((x, y-25), text, fill='red')

# Show image
image.show()
plt.imshow(image);
```



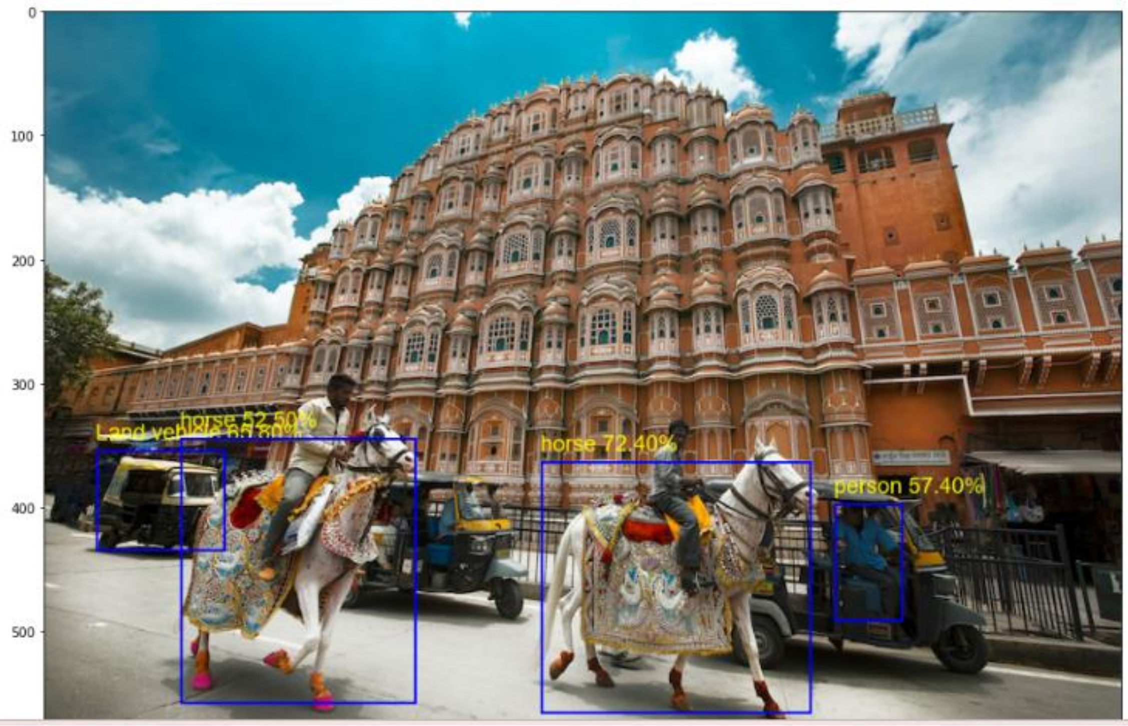
We can see that objects have been detected but they are not very clear.

Let us enlarge the image and adjust font size to have a clear view.

Using ImageFont from PIL lib to adjust the size of font

```
In [16]:  from PIL import ImageFont
```

```
In [17]:  image = Image.open(io.BytesIO(image_data))
          draw = ImageDraw.Draw(image)
          font_size = 18  # set the font size to 24
          for obj in detected_object:
              x, y = obj.rectangle.x, obj.rectangle.y
              w, h = obj.rectangle.w, obj.rectangle.h
              draw.rectangle([x, y, x+w, y+h], outline='blue', width=2)
              text = '{} {:.2f}%'.format(obj.object_property, obj.confidence * 100)
              draw.text((x, y-25), text, fill='yellow', font=ImageFont.truetype("arial.ttf", font_size))

          # Show image with enlarged size
          fig, ax = plt.subplots(figsize=(15, 15))
          ax.imshow(image)
          plt.show()
```



Thus, we can see clearly now that a total of four objects are detected by our machine. Above we can see rectangles drawn on each of the object deteected and their names as detected with the confidence level of each one of them.

Object	Confidence	Coordinates
Land vehicle	65.80%	42, 146, 354, 435
Person	57.40%	638, 693, 397, 492
Horse	52.50%	110, 300, 344, 558
Horse	72.40%	401, 619, 363, 566

References:

Siva, Aditya. (2016). Hawa Mahal Road, Jaipur, India. [Photograph]. Retrieved from https://images.unsplash.com/photo-1477587458883-47145ed94245?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8M3x8aGF3YSUyMG1haGFsJTJDJTIwamFpcHVyJTJDJTIwaW5kaWF8ZW58MHx8MHx8&w=1000&q=80 (Accessed April 19, 2023).