

```
import datetime
import ibm_boto3
from ibm_botocore.client import Config, ClientError
import cv2
import numpy as np
import sys
import ibmiotf.application
import ibmiotf.device
import random
import time
```

```
from cloudfant.client import Cloudfant
from cloudfant.error import CloudfantException
from cloudfant.result import Result, ResultByKey
```

```
organization = "bb2bpw"
deviceType = "RaspberryPi"
deviceId = "24102001"
authMethod = "token"
authToken = "raspberry"
```

```
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    print(cmd.data['command'])

    if cmd.data['command']=="sirenon":
        print("SIREN ON")

    if cmd.data['command']=="sirenoff":
        print("SIREN OFF")
```

```
if cmd.data['command']=="ledon":  
    print("BLINKING LED ON")
```

```
if cmd.data['command']=="ledoff":  
    print("BLINKING LED OFF")
```

```
if cmd.data['command']=="motoron":  
    print("MOTOR ON")
```

```
if cmd.data['command']=="motoroff":  
    print("MOTOR OFF")
```

```
try:
```

```
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":  
authMethod, "auth-token": authToken}
```

```
    deviceCli = ibmiotf.device.Client(deviceOptions)
```

```
except Exception as e:
```

```
    print("Caught exception connecting device: %s" % str(e))
```

```
    sys.exit()
```

```
deviceCli.connect()
```

```
animal_classifier=cv2.CascadeClassifier("haar-animal.xml")
```

```
video=cv2.VideoCapture(0)
```

```
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
```

```
COS_API_KEY_ID = "ffU9G4WuxXvsAV0muEVv-iAi2x3oS_dcS5Q8qceZ2ZXA"
```

```
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
```

```
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-  
storage:global:a/8899ffc5103f4b6c824747890ea97e9f:a7ddbd11-4325-464e-82b5-d07c670c2642::"
```

```
client = Cloudant("apikey-v2-  
3r2lgzf2d6tor5oq125zsnodnm119qtsqnorjqjaff4", "2481c47aba0e16cdeca03d4a11c6deca",  
url="https://apikey-v2-  
3r2lgzf2d6tor5oq125zsnodnm119qtsqnorjqjaff4:2481c47aba0e16cdeca03d4a11c6deca@ebc43d84-  
877b-439c-be4f-3f20160e4b30-bluemix.cloudantnosqldb.appdomain.cloud")
```

```
client.connect()
```

```
database_name = "securitycamera"
```

```
cos = ibm_boto3.resource("s3",  
    ibm_api_key_id=COS_API_KEY_ID,  
    ibm_service_instance_id=COS_RESOURCE_CRN,  
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,  
    config=Config(signature_version="oauth"),  
    endpoint_url=COS_ENDPOINT  
)
```

```
def multi_part_upload(bucket_name, item_name, file_path):
```

```
    try:
```

```
        part_size = 1024 * 1024 * 5
```

```
        file_threshold = 1024 * 1024 * 15
```

```
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(  
            multipart_threshold=file_threshold,  
            multipart_chunksize=part_size  
)
```

```
with open(file_path, "rb") as file_data:
    cos.Object(bucket_name, item_name).upload_fileobj(
        Fileobj=file_data,
        Config=transfer_config
    )
```

```
print("Transfer for {0} Complete!\n".format(item_name))
```

```
except ClientError as be:
```

```
    print("CLIENT ERROR: {0}\n".format(be))
```

```
except Exception as e:
```

```
    print("Unable to complete multi-part upload: {0}".format(e))
```

```
while True:
```

```
    check,frame=video.read()
```

```
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    animal=animal_classifier.detectMultiScale(gray,1.3,5)
```

```
    for(x,y,w,h) in animal:
```

```
        cv2.rectangle(frame, (x,y), (x+y,y+h), (124,255,0), 2)
```

```
        cv2.imshow('Animal Detection', frame)
```

```
        picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
```

```
        picname=picname+".jpg"
```

```
        pic=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
```

```
        cv2.imwrite(picname,frame)
```

```
        mammal=1
```

```
        my_database = client.create_database(database_name)
```

```

multi_part_upload("cloud-object-storage-wb-cos-standard-kcg",picname,pic+".jpg")

if my_database.exists():
    print("{}{database_name}' successfully created.")
    json_document = {
        "_id": pic,
        "link":COS_ENDPOINT+"/cloud-object-storage-wb-cos-standard-kcg/"+picname
    }
    new_document = my_database.create_document(json_document)
    if new_document.exists():
        print("Document '(new_document)' successfully created.")
    time.sleep(1)
    t=26
    h=63
    m=38
    data = {"d":{'temperature': t, 'humidity': h, 'soilmoisture': m, 'mammal': mammal}}

    def myOnPublishCallback():
        print ("Published data to IBM Watson")

    success = deviceCli.publishEvent("data", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoT")
        time.sleep(1)
        mammal=0

deviceCli.commandCallback = myCommandCallback

Key=cv2.waitKey(1)
if Key==ord('q'):

```

```
video.release()
```

```
deviceCli.disconnect()
```

```
cv2.destroyAllWindows()
```

```
break
```