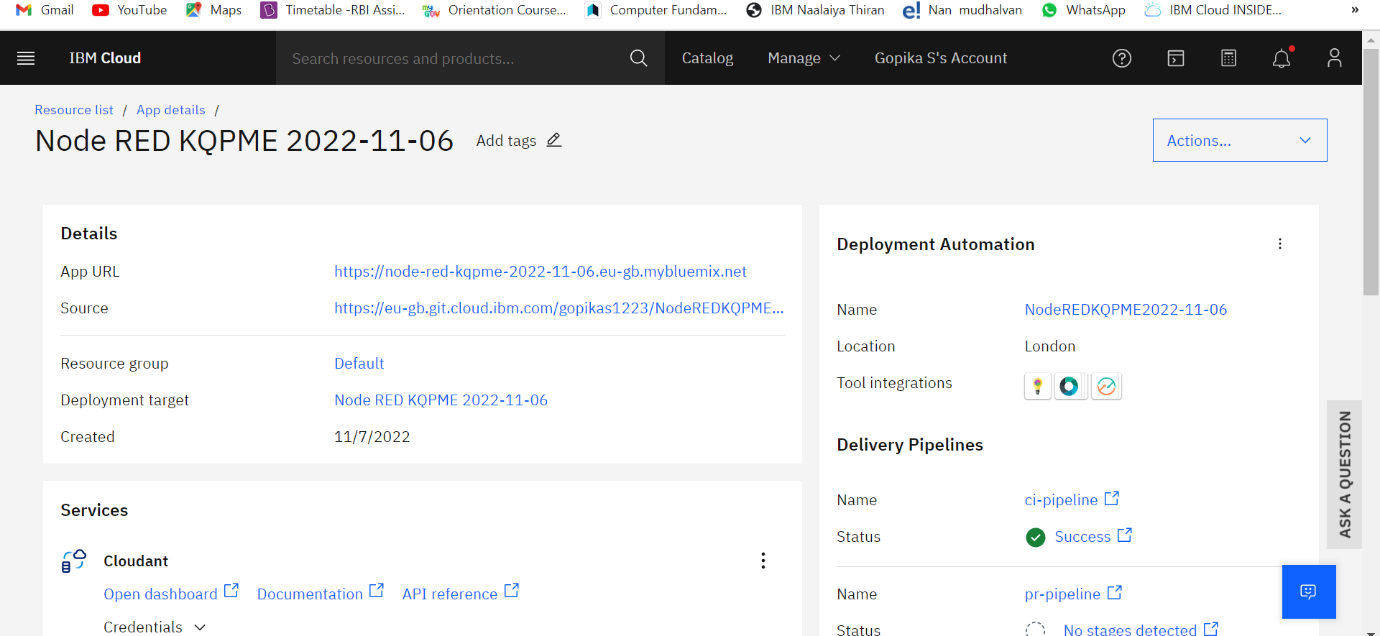
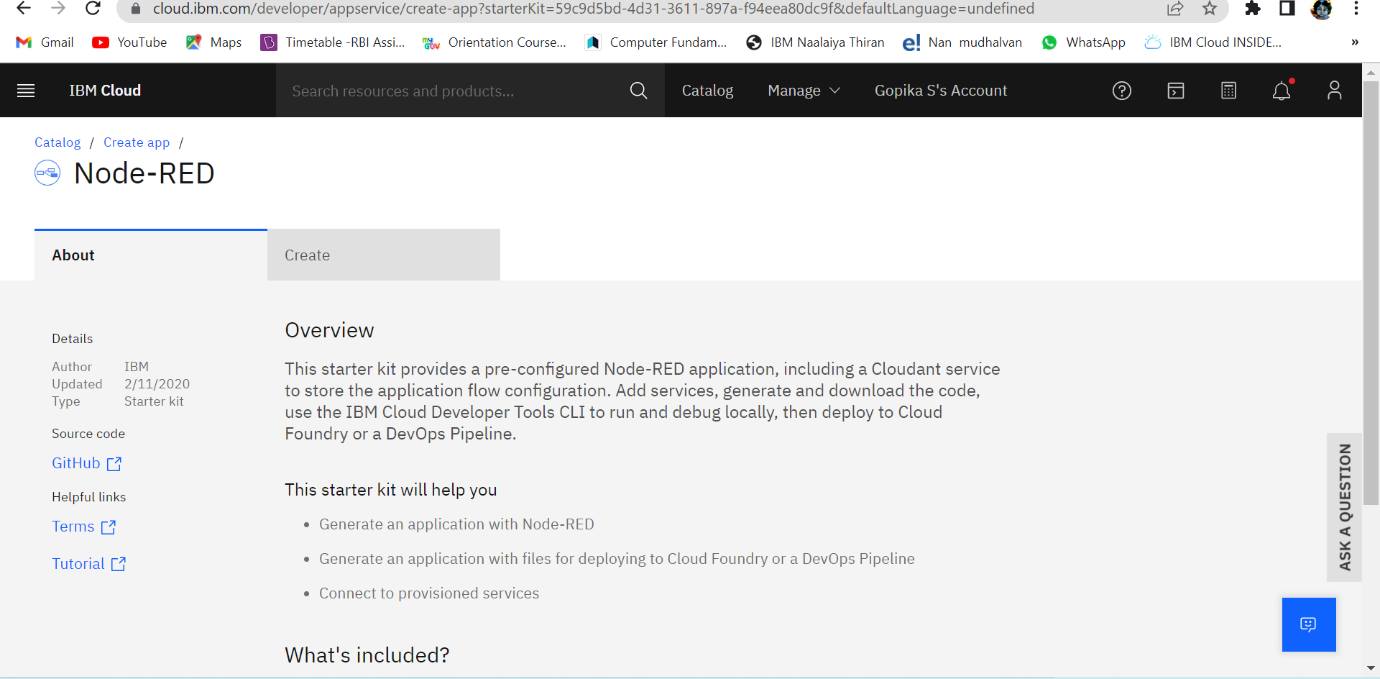
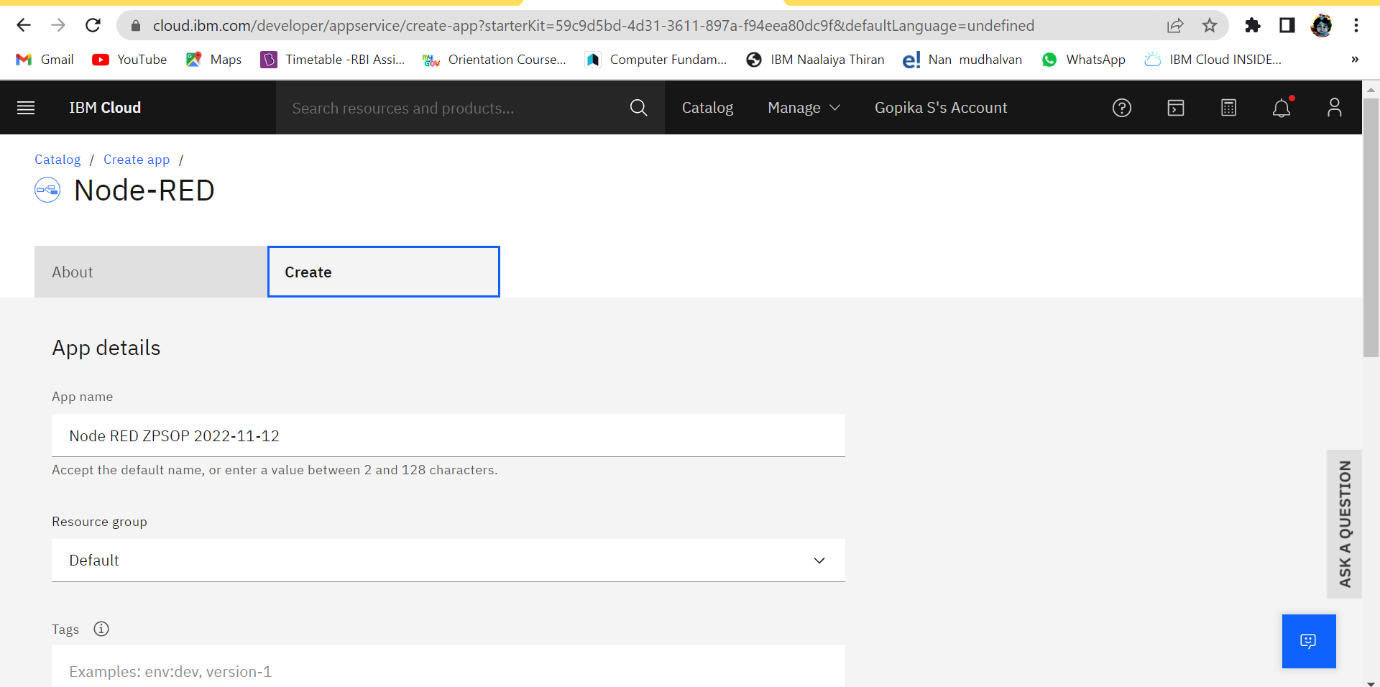
**SPRINT :3**

|  |  |
| --- | --- |
| DATE | 13 November 2022 |
| TEAM ID | PNT2022TMID34135 |
| PROJECT NAME | IoT based smart crop protection system for agriculture |

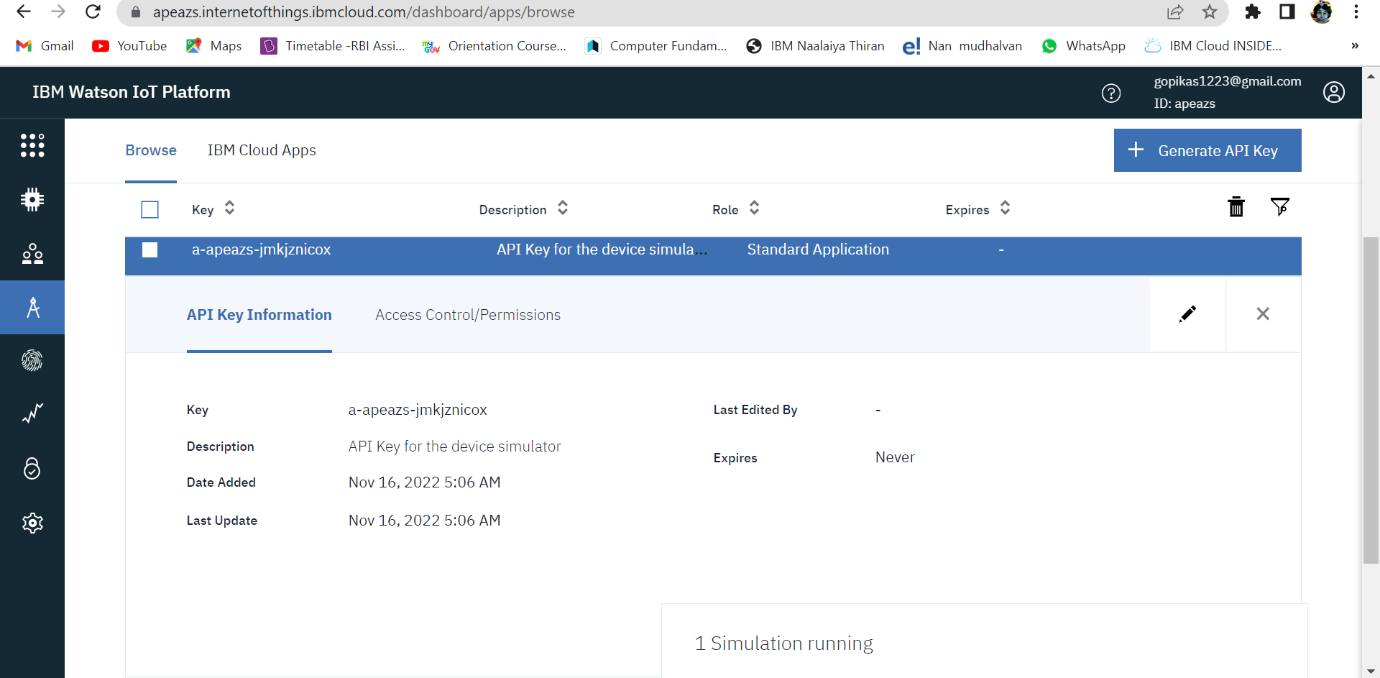
**1.Create a NODE RED service:**

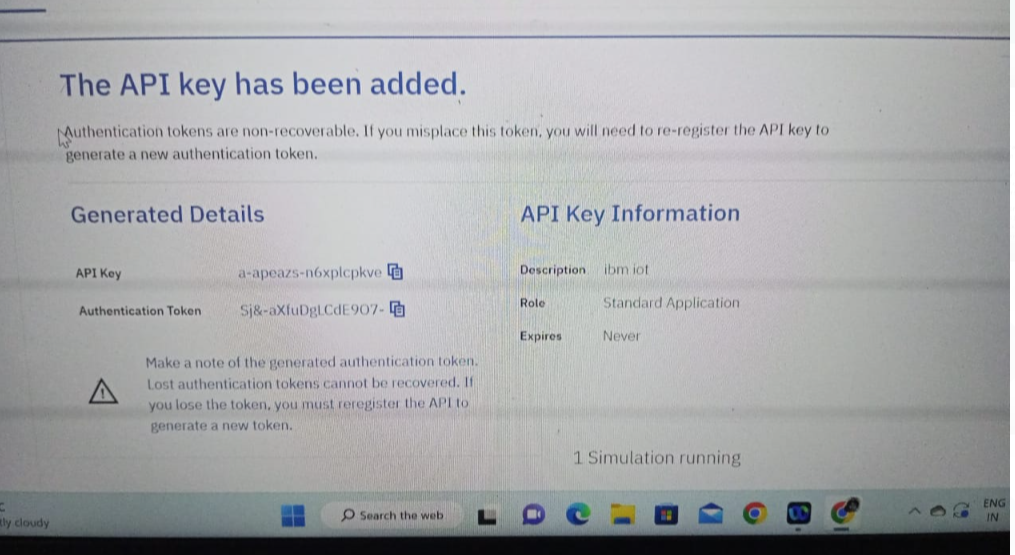






**2. Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform**:





# 3. Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform:

**import cv2**

**import numpy as np**

**import wiotp.sdk.device**

**import playsound**

**import random**

**import time**

**import datetime**

**import ibm\_boto3**

**from ibm\_botocore.client import Config, ClientError**

**#CloudantDB**

**from cloudant.client import Cloudant**

**from cloudant.error import CloudantException**

**from cloudant.result import Result, ResultByKey**

**from clarifai\_grpc.channel.clarifai\_channel import Clarifaichannel from clarifai\_grpc.grpc.api import service\_pb2\_grpc**

**stub service\_pb21\_grpc.V2Stub (ClarifaiChannel.get\_grpc\_channel()) from clarifai\_grpc.grpc.api import service\_pb2, resources\_pb2**

**from clarifai\_grpc.grpc.api.status import status\_code\_pb2**

**This is how you authenticate.**

**metadata = (('authorization', 'Key bc885e5165d74ef48f42f6f6a2c9eb87'),)**

**COS\_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints COS\_API\_KEY\_ID = "f6Ap-ct18m0759UzL7XPbAF7170ome PLLUQOzqmnAzb5" eg "WO0YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk" COS AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"**

**COS\_RESOURCE\_CRN = "crn: v1:bluemix: public: cloud-object-storage: global:a/6b644a3fda97448b888c23eeef263ed6:199able5-0d9d-4201-8e4a-98d868c04368::" eg "crn:vl:bluemix: public: cloud-object-stc**

**clientdb= Cloudant ("apikey-v2-16u3crmdpkghhxe fdikvpssoh5fwezrmuup5fv5g3ubz", "b0ab119f45d3e6255eabb978e7e2f0e1", url="https://apikey-v2-16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119 clientdb.connect()**

**#Create resource**

**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:**

**print ("Starting file transfer for (0) to bucket: {1}\n". format (item\_name, bucket\_name))**

**#set 5 MB chunks part\_size = 1024\* 1024 \* 5**

**set threadhold to 15 MB**

**file\_threshold = 1024\* 1024 \* 15**

**# set the transfer threshold and chunk size**

**transfer\_config= ibm\_boto3.s3.transfer.TransferConfig(**

**multipart\_threshold-file\_threshold,**

**multipart chunksize-part size**

**the upload\_fileobj method will automatically execute a multi-part upload**

**# in 5 MB chunks for all files over 15 MB**

**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))**

**def myCommandCallback (cmd):**

**print ("Command received: %s" % cmd.data)**

**command=cmd.data['command']**

**print (command)**

**if (command=='lighton'):**

**print (lighton')**

**elif (command=='lightoff'):**

**print('lightoff') elif (command== 'motoron'):**

**print ('motoron') elif (command== 'motoroff'):**

**print ('motoroff')**

**myConfig = {**

**"identity":{**

**"orgId": "hj5fmy",**

**"typeId": "NodeMCU", "deviceId": "12345"**

**},**

**"auth": {**

**"token": "12345678"**

**client wiotp.sdk.device. DeviceClient (config=myConfig, logHandlers=None)**

**client.connect()**

**database\_name = "sample"**

**my\_database = clientdb.create\_database (database\_name) if my database.exists():**

**print (f" (database\_name}' successfully created.")**

**cap=cv2.VideoCapture ('garden.mp4')**

**if (cap.isopened () ==True):**

**print('File opened')**

**else:**

**print('File not found")**

**while (cap.isopened()):**

**ret, frame cap.read()**

**gray cv2.cvtColor (frame, cv2.COLOR\_BGR2GRAY) ims cv2.resize (frame, (960, 540))**

**cv2.imwrite ('ex.jpg', ims) with open ("ex.jpg", "rb")**

**as f:**

**file\_bytes = f.read()**

**This is the model ID of a publicly available General model. You may use any other public or custom model ID.**

**request service\_pb2. PostModelOutputsRequest(**

**model\_id='aaa03c23b3724a16a56b629203edc62c',**

**inputs=[resources\_pb2. Input (data=resources\_pb2.Data (image=resources\_pb2. Image (base64=file\_bytes))**

**response stub. PostModelOutputs (request, metadata=metadata)**

**if response.status.code != status\_code\_pb2.SUCCESS:**

**raise Exception ("Request failed, status code: " + str(response.status.code))**

**detect=False**

**for concept in response.outputs [0].data.concepts:**

**#print('12s: %.2f' (concept.name, concept.value)) if (concept.value>0.98):**

**#print (concept.name)**

**if (concept.name=="animal"):**

**print ("Alert! Alert! animal detected")**

**playsound.playsound ('alert.mp3")**

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

**cv2.imwrite (picname+'.jpg', frame)**

**multi\_part\_upload ('gnaneshwar', picname+'.jpg', picname+'.jpg')**

**json\_document={"link":COS\_ENDPOINT+'/'+'gnaneshwar'+'/'+picname+'.jpg"}**

**new\_document = my\_database.create\_document (json\_document)**

**if new\_document.exists(): print (f"Document successfully created.")**

**time.sleep(5)**

**detect=True**

**moist=random.randint (0, 100)**

**humidity=random.randint (0,100)**

**myData={'Animal': detect, 'moisture' :moist, 'humidity': humidity}**

**print (myData)**

**if (humidity!=None):**

**client.publishEvent (eventId="status", msgFormat="json", data myData, qos=0, onPublish=None)**

**print("Publish Ok..")**

**client.commandCallback = myCommandCallback**

**cv2.imshow('frame', ims)**

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**

**client.disconnect()**

**cap.release()**

**cv2.destroyAllWindows()**

# 4. After developing python code, commands are received just print the statements which represent the control of the devices:

# OUTPUT:

# 

**COMPLETION OF SPRINT :3**

