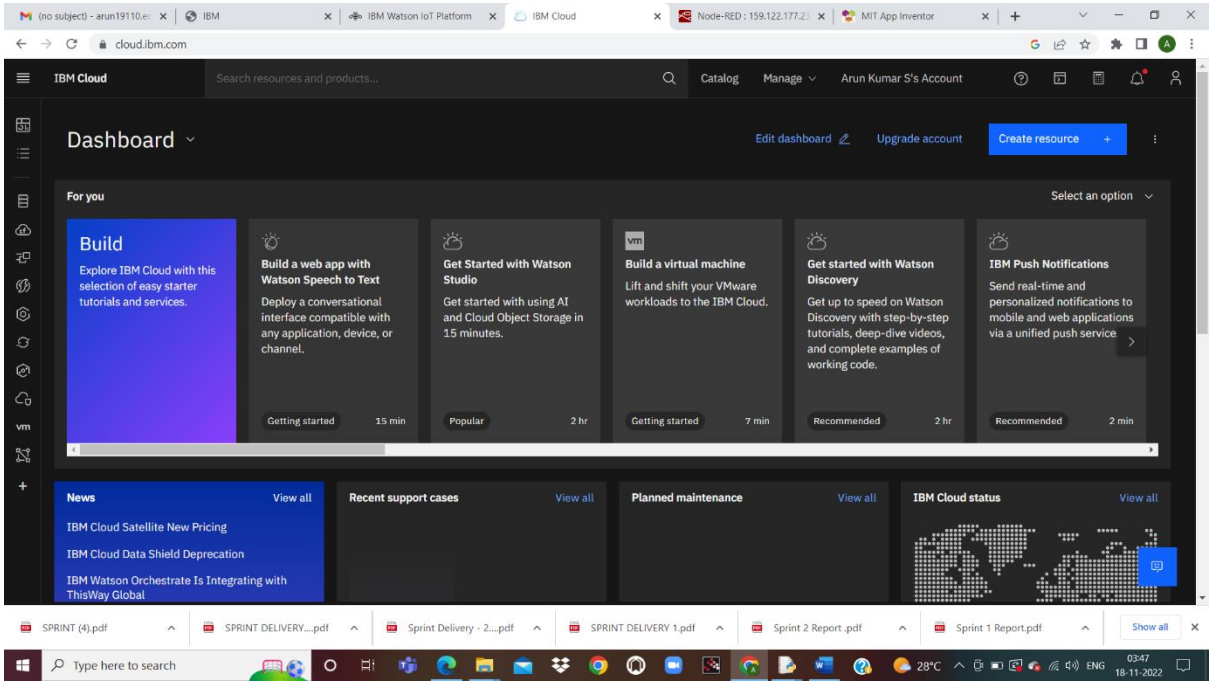
SmartFarmer - IoT Enabled Smart Farming Application

Project Development Phase – Sprint 1

IBM Cloud

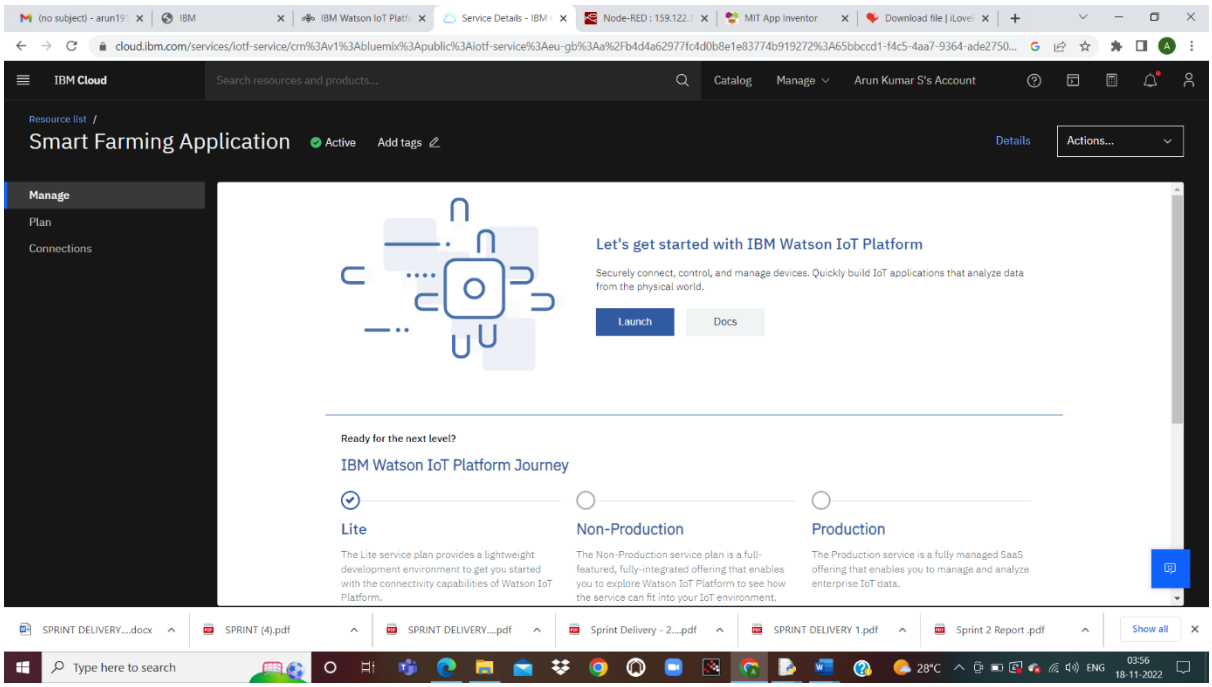


- **Search URL cloud.ibm.com**
- **Create an account**
- **Sign up with login credentials**

Create and Configure IBM Cloud services

- **Create the IBM Watson IOT Platform**
- **Create Node red service**

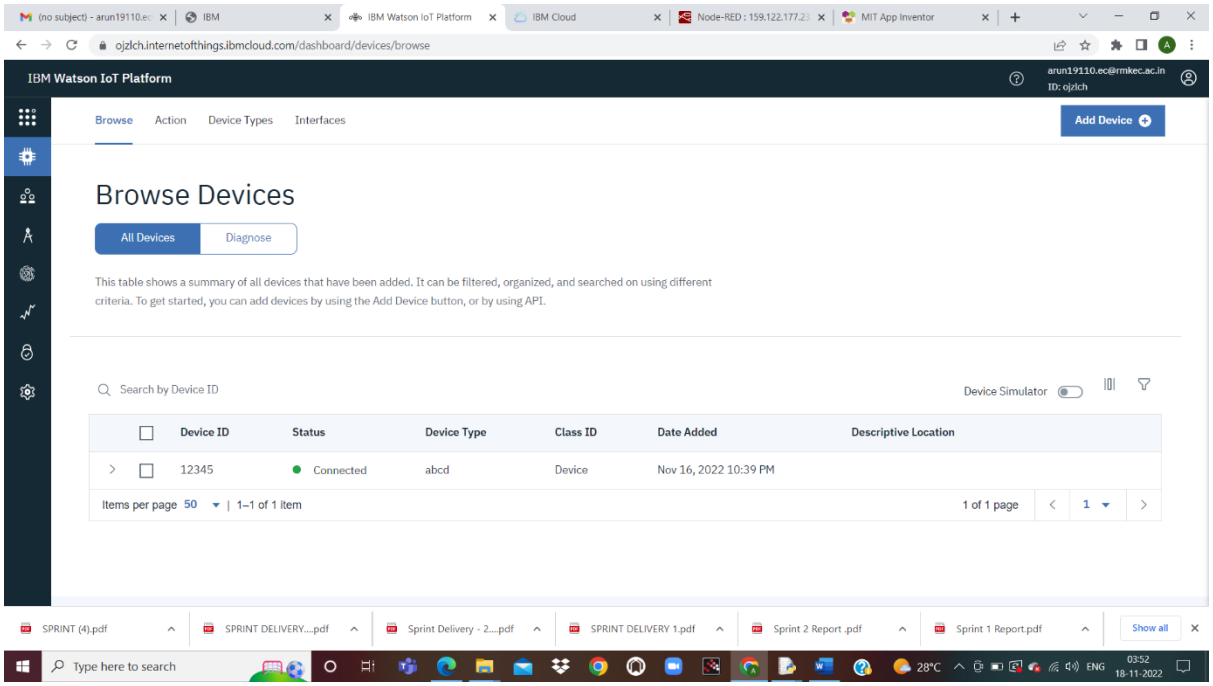
IBM Watson IOT Platform



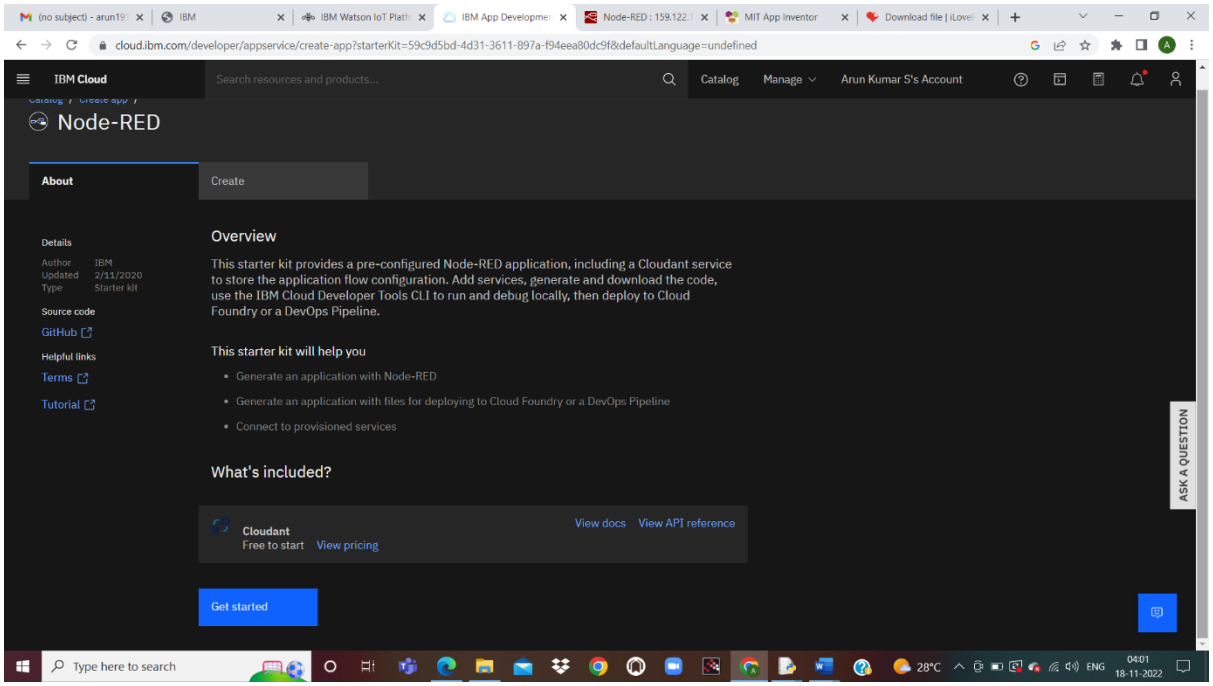
A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.

Procedure

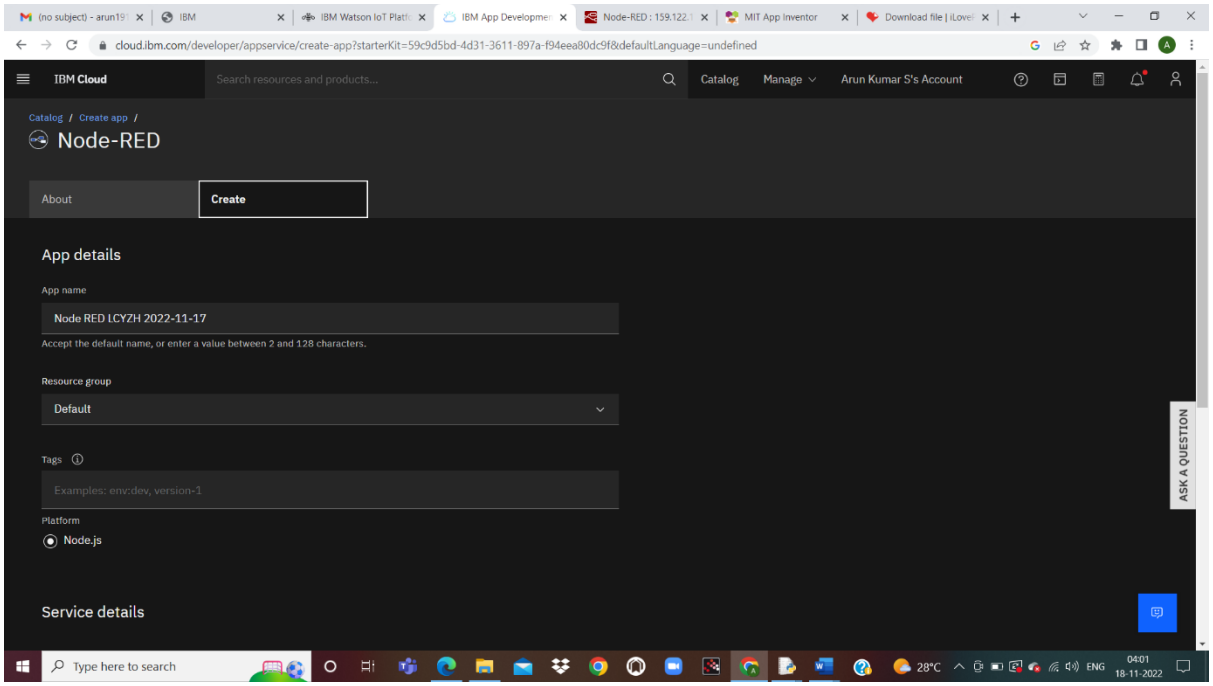
- **Create an account in IBM cloud using your email ID**
- **Create IBM Watson Platform in services in your IBM cloud account**
- **Launch the IBM Watson IoT Platform**
- **Create a new device**
- **Give credentials like device type, device ID, Auth. Token**
- **Create API key and store API key and token elsewhere.**

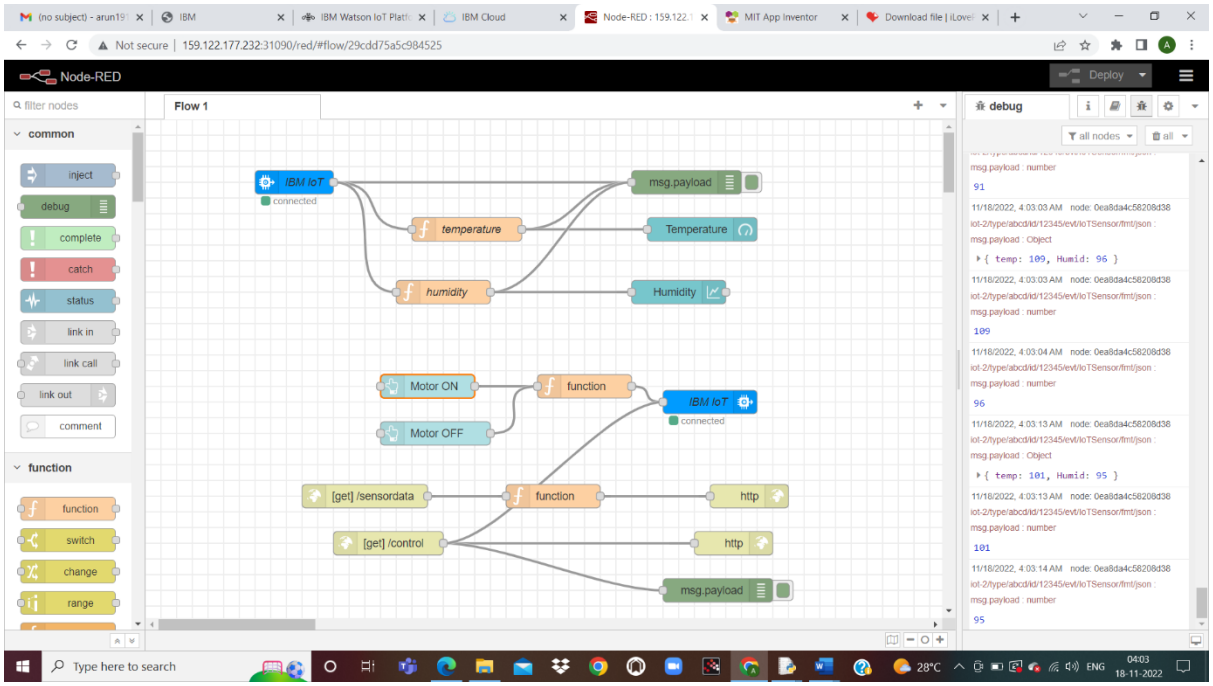


Create Node red Service



Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring togetherhardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser- based flow editor, which can be used to create JavaScript functions.





Python IDLE

- **Install python 3.7.0 version**
- **Import necessary libraries**

Code

```
organization = "ojzlch"
deviceType = "abcd"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
   print("Command received: %s" % cmd.data['command'])
   status=cmd.data['command']
   if status=="motoron":
      print ("motor is on")
   elif status == "motoroff":
      print ("motor is off")
   else :
      print ("please send proper command")

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()

# Connect and send a datapoint "hello" with value "world"
into the cloud as an event of type "greeting" 10 times
deviceCli.connect()
```

```
while True:
        #Get Sensor Data from DHT11

        temp=random.randint(90,110)
        Humid=random.randint(60,100)

        data = { 'temp' : temp, 'Humid': Humid }
        #print data
        def myOnPublishCallback():
           print ("Published Temperature = %s C" % temp,
"Humidity = %s %%" % Humid, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json",
data, qos=0, on_publish=myOnPublishCallback)
        if not success:
           print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```