Department of Computer Science, Electrical and Space Engineering Luleå University of Technology

Cloud Services

Lab 3 Report: Programming Cloud Services - Compute Services

Author:
Hanna Ogbazghi
Elyas Khorasani
Yacine Rabehi

Supervisors:
Karan Mitra

30 November 2021

Contents

1 Objective

The objective of this lab is to:
• Setup a programming environment for building (using programming tools and languages) Cloud services for a major Cloud provider, for example, Amazon Web services (AWS).
• Develop Cloud services for managing the EC2 compute services using the APIs provided by the AWS.
• Develop Cloud services for monitoring the EC2 compute services using the APIs provided by the AWS.

2 Exercise A

2.1 Question 1: Identify ways of creating the Amazon EC2 service clients

The diagram shown in figure 1 illustrates the dependencies for the employed functions (methods). We note that the access to the dependencies is limited for boto3 (we cannot see the different calls made in order to execute the methods). There might be more access to the details when employing Java instead[1].

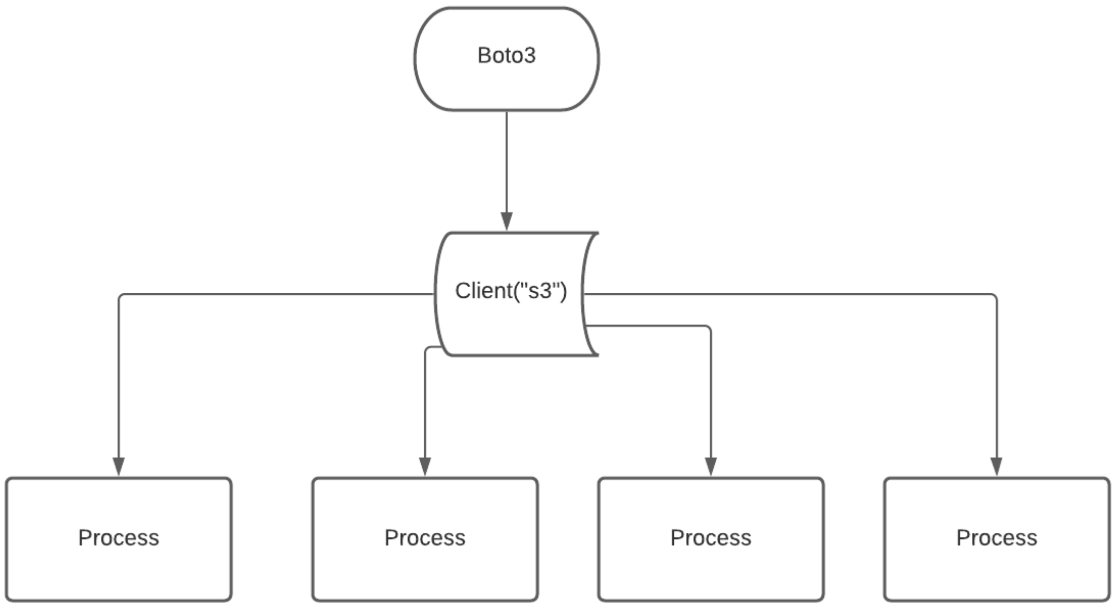
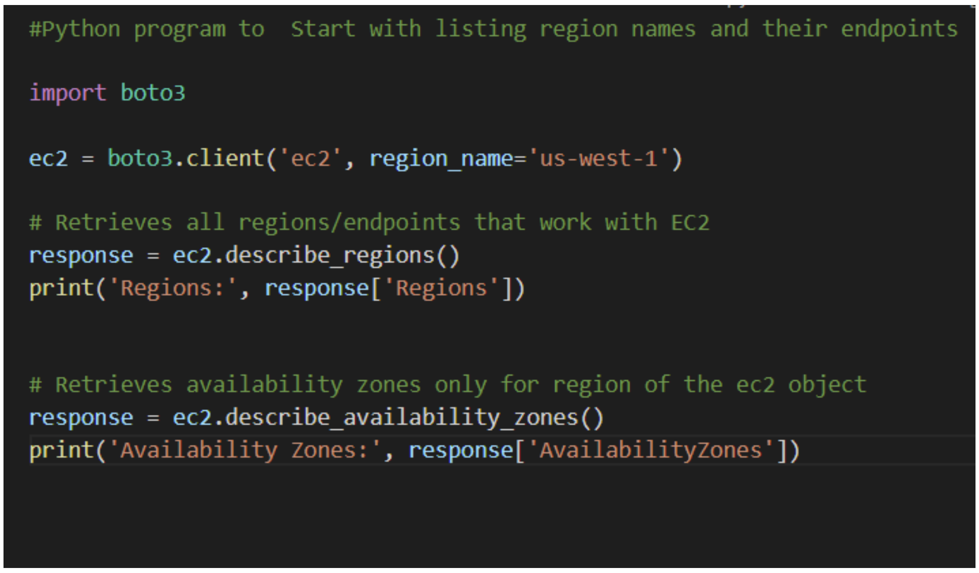


Figure 1: Dependency Diagram for the methods employed

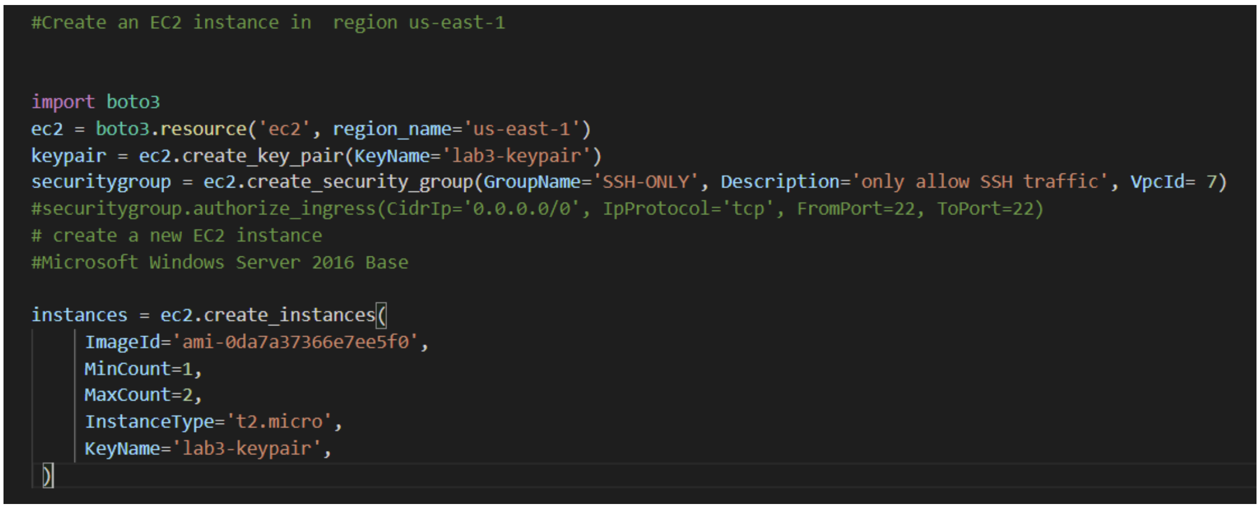
2.2 Question 2: Python program to Start with listing region names and their endpoints.



```python
#Python program to  Start with listing region names and their endpoints

import boto3

ec2 = boto3.client('ec2', region_name='us-west-1')

# Retrieves all regions/endpoints that work with EC2
response = ec2.describe_regions()
print('Regions:', response['Regions'])


# Retrieves availability zones only for region of the ec2 object
response = ec2.describe_availability_zones()
print('Availability Zones:', response['AvailabilityZones'])
```

Figure 2: List EC2 instances

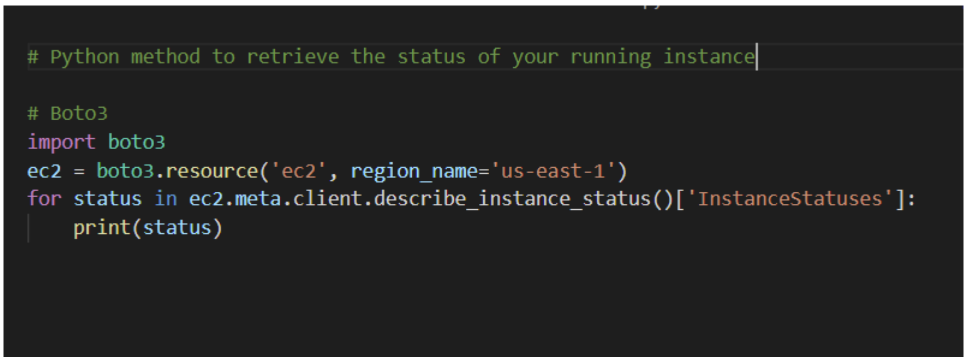
2.3 Question 3: Python program to run an instance from the list of regions.



```python
#Create an EC2 instance in  region us-east-1


import boto3
ec2 = boto3.resource('ec2', region_name='us-east-1')
keypair = ec2.create_key_pair(KeyName='lab3-keypair')
securitygroup = ec2.create_security_group(GroupName='SSH-ONLY', Description='only allow SSH traffic', VpcId= 7)
#securitygroup.authorize_ingress(CidrIp='0.0.0.0/0', IpProtocol='tcp', FromPort=22, ToPort=22)
# create a new EC2 instance
#Microsoft Windows Server 2016 Base

instances = ec2.create_instances(
        ImageId='ami-0da7a37366e7ee5f0',
        MinCount=1,
        MaxCount=2,
        InstanceType='t2.micro',
        KeyName='lab3-keypair',
 )
```

Figure 3: Create An EC2 Instance

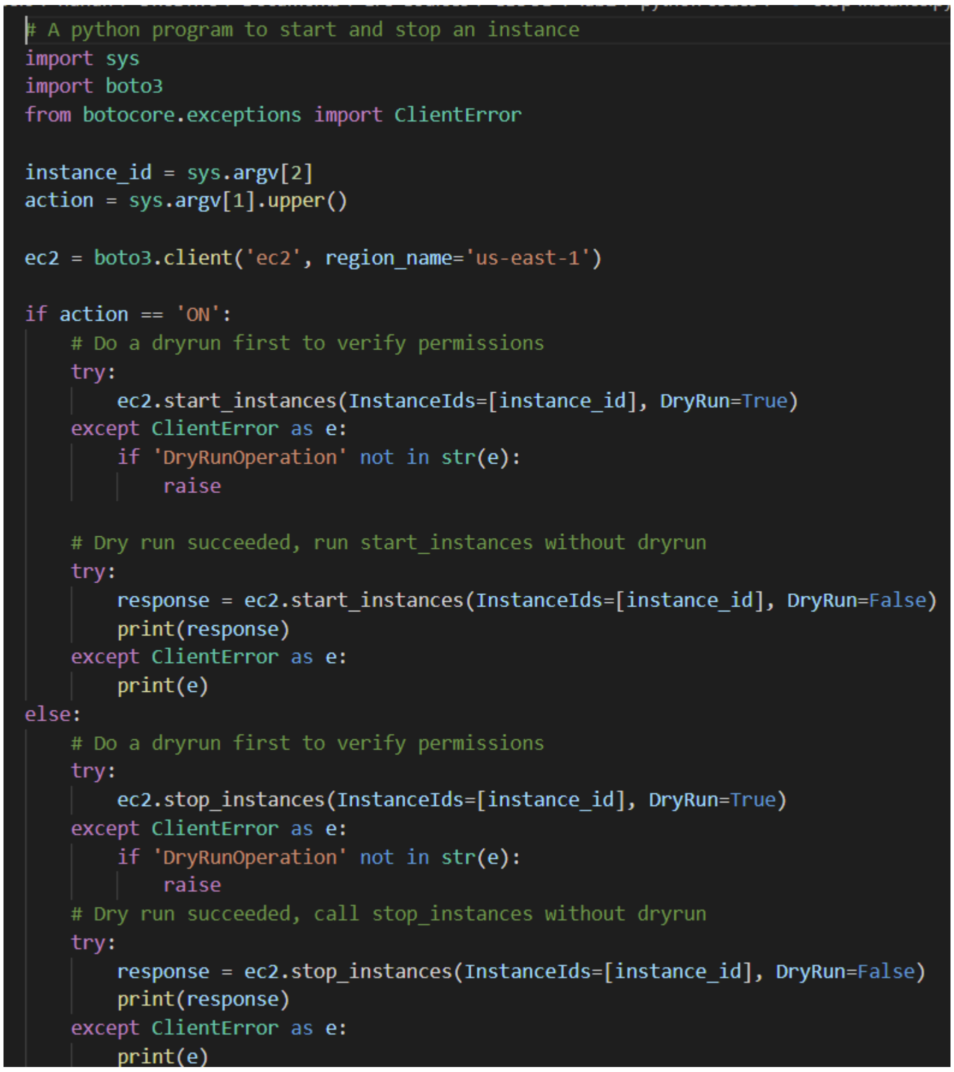
2.4 Question 4: Python program to retrieve the status of your running instance(s).



```
# Python method to retrieve the status of your running instance

# Boto3
import boto3
ec2 = boto3.resource('ec2', region_name='us-east-1')
for status in ec2.meta.client.describe_instance_status()['InstanceStatuses']:
    print(status)
```

Figure 4: Instance Status

2.5 Question 5: Python program to start and stop an instance(s) that you started.



```python
# A python program to start and stop an instance
import sys
import boto3
from botocore.exceptions import ClientError

instance_id = sys.argv[2]
action = sys.argv[1].upper()

ec2 = boto3.client('ec2', region_name='us-east-1')

if action == 'ON':
    # Do a dryrun first to verify permissions
    try:
        ec2.start_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
        if 'DryRunOperation' not in str(e):
            raise

    # Dry run succeeded, run start_instances without dryrun
    try:
        response = ec2.start_instances(InstanceIds=[instance_id], DryRun=False)
        print(response)
    except ClientError as e:
        print(e)
else:
    # Do a dryrun first to verify permissions
    try:
        ec2.stop_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
        if 'DryRunOperation' not in str(e):
            raise
    # Dry run succeeded, call stop_instances without dryrun
    try:
        response = ec2.stop_instances(InstanceIds=[instance_id], DryRun=False)
        print(response)
    except ClientError as e:
        print(e)
```

Figure 5: Start and Stop An Instance

3 Exercise B

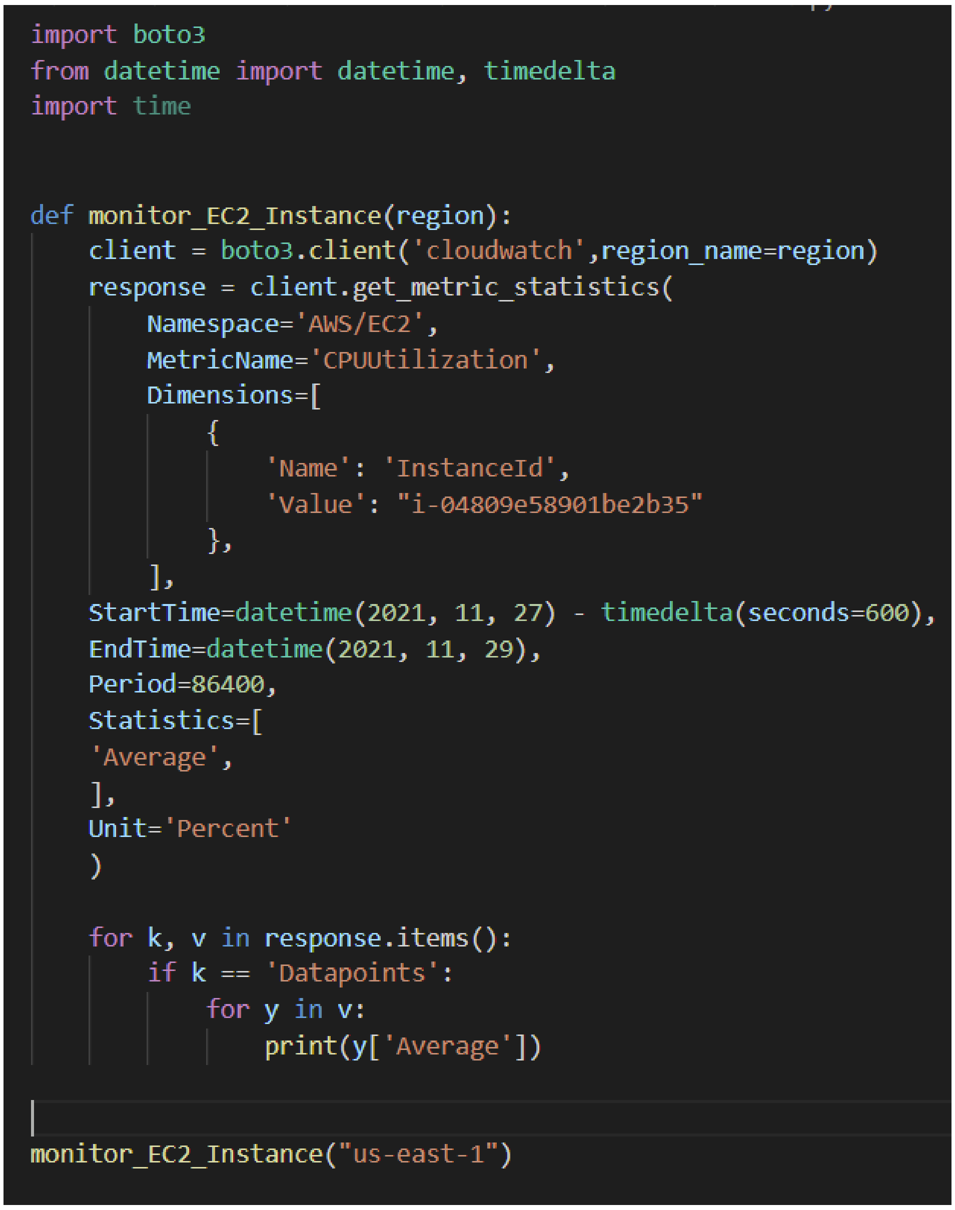
3.1 Question 1: Identify ways of creating the CloudWatch clients.

To gather and track metrics, collect and monitor log files, and create alarms one can use Amazon CloudWatch. Amazon CloudWatch can monitor AWS resources like Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics and log files generated by your applications and services[1]. The creation of the CloudWatch clients was implemented to monitor the status of EC2 instances (see section 3.2).

3.2 Question 2: Python program to monitor the status of your EC2 instances.

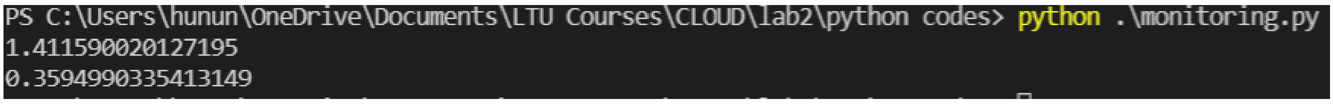
There are several metrics that we can use in-order to monitor the instances.The following link provides the available CloudWatch metrics for instances.

`https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/viewing_metrics_with_cloudwatch`



```python
import boto3
from datetime import datetime, timedelta
import time


def monitor_EC2_Instance(region):
    client = boto3.client('cloudwatch',region_name=region)
    response = client.get_metric_statistics(
        Namespace='AWS/EC2',
        MetricName='CPUUtilization',
        Dimensions=[
            {
                'Name': 'InstanceId',
                'Value': "i-04809e58901be2b35"
            },
        ],
    StartTime=datetime(2021, 11, 27) - timedelta(seconds=600),
    EndTime=datetime(2021, 11, 29),
    Period=86400,
    Statistics=[
    'Average',
    ],
    Unit='Percent'
    )

    for k, v in response.items():
        if k == 'Datapoints':
            for y in v:
                print(y['Average'])


monitor_EC2_Instance("us-east-1")
```

Figure 6: EC2 CPU Utilization



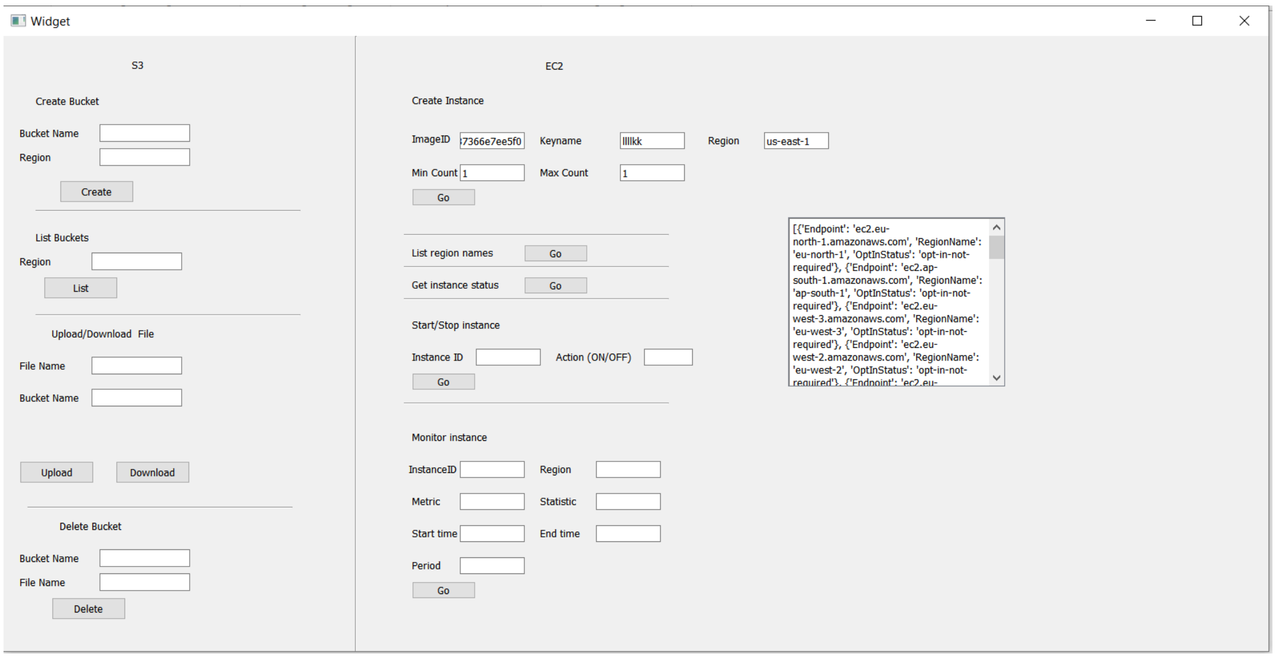
```
PS C:\Users\hunun\OneDrive\Documents\LTU Courses\CLOUD\lab2\python codes> python .\monitoring.py
1.411590020127195
0.3594990335413149
```

Figure 7: CPU Utilization Result

4 Exercise C

4.1 Question: build a CLI or a UI facilitating the calling of created functions.

We developed a user interface using QT creator. We note that we added the functions for both EC2 and S3.



ImageID: 7366e7ee5f0
Keyname: lllkk
Region: us-east-1
Min Count: 1
Max Count: 1

```
[{'Endpoint': 'ec2.eu-
north-1.amazonaws.com', 'RegionName':
'eu-north-1', 'OptInStatus': 'opt-in-not-
required'}, {'Endpoint': 'ec2.ap-
south-1.amazonaws.com', 'RegionName':
'ap-south-1', 'OptInStatus': 'opt-in-not-
required'}, {'Endpoint': 'ec2.eu-
west-3.amazonaws.com', 'RegionName':
'eu-west-3', 'OptInStatus': 'opt-in-not-
required'}, {'Endpoint': 'ec2.eu-
west-2.amazonaws.com', 'RegionName':
'eu-west-2', 'OptInStatus': 'opt-in-not-
required'}, {'Endpoint': 'ec2.eu-
```

Figure 8: User Interface For S3 and EC2

5 Appendix

```
    # -*- coding: utf-8 -*-
# Form implementation generated from reading ui file 'form.ui'
# Created by: PyQt5 UI code generator 5.15.4

#User interface for EC2 and S3(from lab2)

import boto3
import sys
import os
import logging
from  botocore.exceptions import ClientError
from datetime import datetime, timedelta
import time
from PyQt5 import QtCore, QtGui, QtWidgets
import string


class Ui_Widget(object):
    def setupUi(self, Widget):
        Widget.setObjectName("Widget")
        Widget.resize(1580, 767)
        self.label = QtWidgets.QLabel(Widget)
        self.label.setGeometry(QtCore.QRect(160, 20, 131, 31))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(Widget)
        self.label_2.setGeometry(QtCore.QRect(40, 70, 91, 21))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(Widget)
        self.label_3.setGeometry(QtCore.QRect(20, 110, 81, 21))
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(Widget)
        self.label_4.setGeometry(QtCore.QRect(20, 140, 81, 21))
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(Widget)
        self.label_5.setGeometry(QtCore.QRect(40, 240, 91, 21))
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(Widget)
        self.label_6.setGeometry(QtCore.QRect(20, 270, 81, 21))
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(Widget)
        self.label_7.setGeometry(QtCore.QRect(60, 360, 141, 21))
        self.label_7.setObjectName("label_7")
        self.label_8 = QtWidgets.QLabel(Widget)
        self.label_8.setGeometry(QtCore.QRect(20, 400, 81, 21))
        self.label_8.setObjectName("label_8")
        self.label_9 = QtWidgets.QLabel(Widget)
        self.label_9.setGeometry(QtCore.QRect(20, 440, 81, 21))
        self.label_9.setObjectName("label_9")
        self.pushButton = QtWidgets.QPushButton(Widget)
        self.pushButton.setGeometry(QtCore.QRect(20, 530, 93, 28))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(Widget)
        self.pushButton_2.setGeometry(QtCore.QRect(140, 530, 93, 28))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 = QtWidgets.QPushButton(Widget)
        self.pushButton_3.setGeometry(QtCore.QRect(50, 300, 93, 28))
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_4 = QtWidgets.QPushButton(Widget)
        self.pushButton_4.setGeometry(QtCore.QRect(70, 180, 93, 28))
        self.pushButton_4.setObjectName("pushButton_4")
        self.lineEdit = QtWidgets.QLineEdit(Widget)
        self.lineEdit.setGeometry(QtCore.QRect(120, 110, 113, 22))
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_2.setGeometry(QtCore.QRect(120, 140, 113, 22))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.lineEdit_3 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_3.setGeometry(QtCore.QRect(110, 270, 113, 22))
        self.lineEdit_3.setObjectName("lineEdit_3")
        self.label_11 = QtWidgets.QLabel(Widget)
        self.label_11.setGeometry(QtCore.QRect(70, 600, 91, 21))
        self.label_11.setObjectName("label_11")
        self.label_12 = QtWidgets.QLabel(Widget)
        self.label_12.setGeometry(QtCore.QRect(20, 640, 81, 21))
        self.label_12.setObjectName("label_12")
        self.lineEdit_4 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_4.setGeometry(QtCore.QRect(120, 640, 113, 22))
        self.lineEdit_4.setObjectName("lineEdit_4")
        self.pushButton_5 = QtWidgets.QPushButton(Widget)
        self.pushButton_5.setGeometry(QtCore.QRect(60, 700, 93, 28))
        self.pushButton_5.setObjectName("pushButton_5")
        self.lineEdit_999 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_999.setGeometry(QtCore.QRect(740, 570, 81, 21))
        self.lineEdit_999.setObjectName("lineEdit_999")
        self.lineEdit_997 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_997.setGeometry(QtCore.QRect(590, 390, 81, 21))
        self.lineEdit_997.setObjectName("lineEdit_997")
        self.lineEdit_998 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_998.setGeometry(QtCore.QRect(800, 390, 61, 21))
        self.lineEdit_998.setObjectName("lineEdit_998")
        self.label_13 = QtWidgets.QLabel(Widget)
        self.label_13.setGeometry(QtCore.QRect(510, 350, 131, 20))
        self.label_13.setObjectName("label_13")
        self.pushButton_6 = QtWidgets.QPushButton(Widget)
        self.pushButton_6.setGeometry(QtCore.QRect(510, 680, 80, 22))
        self.pushButton_6.setObjectName("pushButton_6")
        self.label_14 = QtWidgets.QLabel(Widget)
        self.label_14.setGeometry(QtCore.QRect(510, 260, 111, 20))
        self.label_14.setObjectName("label_14")
        self.label_15 = QtWidgets.QLabel(Widget)
        self.label_15.setGeometry(QtCore.QRect(510, 300, 121, 20))
        self.label_15.setObjectName("label_15")
        self.label_16 = QtWidgets.QLabel(Widget)
        self.label_16.setGeometry(QtCore.QRect(510, 120, 61, 16))
        self.label_16.setObjectName("label_16")
        self.pushButton_7 = QtWidgets.QPushButton(Widget)
        self.pushButton_7.setGeometry(QtCore.QRect(510, 190, 80, 22))
        self.pushButton_7.setObjectName("pushButton_7")
        self.label_19 = QtWidgets.QLabel(Widget)
        self.label_19.setGeometry(QtCore.QRect(677, 30, 47, 14))
        self.label_19.setObjectName("label_19")
        self.label_17 = QtWidgets.QLabel(Widget)
        self.label_17.setGeometry(QtCore.QRect(670, 120, 71, 20))
        self.label_17.setObjectName("label_17")
        self.pushButton_8 = QtWidgets.QPushButton(Widget)
        self.pushButton_8.setGeometry(QtCore.QRect(650, 300, 80, 22))
        self.pushButton_8.setObjectName("pushButton_8")
        self.lineEdit_9910 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9910.setGeometry(QtCore.QRect(740, 530, 81, 21))
        self.lineEdit_9910.setObjectName("lineEdit_9910")
        self.label_18 = QtWidgets.QLabel(Widget)
        self.label_18.setGeometry(QtCore.QRect(670, 570, 61, 20))
        self.label_18.setObjectName("label_18")
        self.label_20 = QtWidgets.QLabel(Widget)
        self.label_20.setGeometry(QtCore.QRect(670, 160, 61, 20))
        self.label_20.setObjectName("label_20")
        self.line_3 = QtWidgets.QFrame(Widget)
        self.line_3.setGeometry(QtCore.QRect(500, 320, 331, 16))
        self.line_3.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_3.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_3.setObjectName("line_3")
        self.lineEdit_9911 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9911.setGeometry(QtCore.QRect(770, 160, 81, 21))
        self.lineEdit_9911.setObjectName("lineEdit_9911")
        self.lineEdit_9912 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9912.setGeometry(QtCore.QRect(740, 610, 81, 21))
        self.lineEdit_9912.setObjectName("lineEdit_9912")
        self.pushButton_9 = QtWidgets.QPushButton(Widget)
        self.pushButton_9.setGeometry(QtCore.QRect(650, 260, 80, 22))
        self.pushButton_9.setObjectName("pushButton_9")
        self.line = QtWidgets.QFrame(Widget)
        self.line.setGeometry(QtCore.QRect(500, 240, 331, 16))
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.label_21 = QtWidgets.QLabel(Widget)
        self.label_21.setGeometry(QtCore.QRect(690, 390, 101, 20))
        self.label_21.setObjectName("label_21")
        self.label_22 = QtWidgets.QLabel(Widget)
        self.label_22.setGeometry(QtCore.QRect(670, 530, 51, 20))
        self.label_22.setObjectName("label_22")
        self.label_23 = QtWidgets.QLabel(Widget)
        self.label_23.setGeometry(QtCore.QRect(510, 490, 101, 20))
        self.label_23.setObjectName("label_23")
        self.lineEdit_9913 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9913.setGeometry(QtCore.QRect(570, 650, 81, 21))
        self.lineEdit_9913.setObjectName("lineEdit_9913")
        self.label_24 = QtWidgets.QLabel(Widget)
        self.label_24.setGeometry(QtCore.QRect(670, 610, 51, 20))
        self.label_24.setObjectName("label_24")
        self.lineEdit_993 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_993.setGeometry(QtCore.QRect(570, 160, 81, 21))
        self.lineEdit_993.setObjectName("lineEdit_993")
        self.label_25 = QtWidgets.QLabel(Widget)
        self.label_25.setGeometry(QtCore.QRect(510, 390, 71, 20))
        self.label_25.setObjectName("label_25")
        self.label_26 = QtWidgets.QLabel(Widget)
        self.label_26.setGeometry(QtCore.QRect(506, 530, 61, 20))
        self.label_26.setObjectName("label_26")
        self.line_2 = QtWidgets.QFrame(Widget)
        self.line_2.setGeometry(QtCore.QRect(500, 280, 331, 16))
        self.line_2.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_2.setObjectName("line_2")
        self.lineEdit_99288 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_99288.setGeometry(QtCore.QRect(570, 120, 81, 21))
        self.lineEdit_99288.setObjectName("lineEdit_99288")
        self.label_27 = QtWidgets.QLabel(Widget)
        self.label_27.setGeometry(QtCore.QRect(510, 70, 91, 20))
        self.label_27.setObjectName("label_27")
        self.lineEdit_992 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_992.setGeometry(QtCore.QRect(770, 120, 81, 21))
        self.lineEdit_992.setObjectName("lineEdit_992")
        self.label_28 = QtWidgets.QLabel(Widget)
        self.label_28.setGeometry(QtCore.QRect(510, 610, 61, 20))
        self.label_28.setObjectName("label_28")
        self.label_29 = QtWidgets.QLabel(Widget)
        self.label_29.setGeometry(QtCore.QRect(510, 570, 51, 20))
        self.label_29.setObjectName("label_29")
        self.lineEdit_9914 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9914.setGeometry(QtCore.QRect(570, 530, 81, 21))
        self.lineEdit_9914.setObjectName("lineEdit_9914")
        self.line_5 = QtWidgets.QFrame(Widget)
        self.line_5.setGeometry(QtCore.QRect(430, 0, 20, 771))
        self.line_5.setFrameShape(QtWidgets.QFrame.VLine)
        self.line_5.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_5.setObjectName("line_5")
        self.lineEdit_9915 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9915.setGeometry(QtCore.QRect(570, 570, 81, 21))
        self.lineEdit_9915.setObjectName("lineEdit_9915")
        self.label_30 = QtWidgets.QLabel(Widget)
        self.label_30.setGeometry(QtCore.QRect(510, 160, 61, 20))
        self.label_30.setObjectName("label_30")
        self.label_31 = QtWidgets.QLabel(Widget)
        self.label_31.setGeometry(QtCore.QRect(510, 650, 51, 20))
        self.label_31.setObjectName("label_31")
        self.pushButton_10 = QtWidgets.QPushButton(Widget)
        self.pushButton_10.setGeometry(QtCore.QRect(510, 420, 80, 22))
        self.pushButton_10.setObjectName("pushButton_10")
        self.lineEdit_9916 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9916.setGeometry(QtCore.QRect(570, 610, 81, 21))
        self.lineEdit_9916.setObjectName("lineEdit_9916")
        self.line_4 = QtWidgets.QFrame(Widget)
        self.line_4.setGeometry(QtCore.QRect(500, 450, 331, 16))
        self.line_4.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_4.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_4.setObjectName("line_4")
        self.line_6 = QtWidgets.QFrame(Widget)
        self.line_6.setGeometry(QtCore.QRect(30, 580, 331, 16))
        self.line_6.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_6.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_6.setObjectName("line_6")
        self.line_7 = QtWidgets.QFrame(Widget)
        self.line_7.setGeometry(QtCore.QRect(40, 340, 331, 16))
        self.line_7.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_7.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_7.setObjectName("line_7")
```

```
        self.line_8 = QtWidgets.QFrame(Widget)
        self.line_8.setGeometry(QtCore.QRect(40, 210, 331, 16))
        self.line_8.setFrameShape(QtWidgets.QFrame.HLine)
        self.line_8.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_8.setObjectName("line_8")
        self.scrollArea = QtWidgets.QScrollArea(Widget)
        self.scrollArea.setGeometry(QtCore.QRect(980, 226, 271, 211))
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 269, 209))
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
        self.textEdit = QtWidgets.QTextEdit(self.scrollAreaWidgetContents)
        self.textEdit.setGeometry(QtCore.QRect(0, 0, 271, 211))
        self.textEdit.setObjectName("textEdit")
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.lineEdit_5 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_5.setGeometry(QtCore.QRect(110, 400, 113, 22))
        self.lineEdit_5.setObjectName("lineEdit_5")
        self.lineEdit_6 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_6.setGeometry(QtCore.QRect(110, 440, 113, 22))
        self.lineEdit_6.setObjectName("lineEdit_6")
        self.label_f = QtWidgets.QLabel(Widget)
        self.label_f.setGeometry(QtCore.QRect(20, 670, 81, 21))
        self.label_f.setObjectName("label_f")
        self.lineEdit_f = QtWidgets.QLineEdit(Widget)
        self.lineEdit_f.setGeometry(QtCore.QRect(120, 670, 113, 22))
        self.lineEdit_f.setObjectName("lineEdit_f")
        self.lineEdit_9917 = QtWidgets.QLineEdit(Widget)
        self.lineEdit_9917.setGeometry(QtCore.QRect(950, 120, 81, 21))
        self.lineEdit_9917.setObjectName("lineEdit_9917")
        self.label_32 = QtWidgets.QLabel(Widget)
        self.label_32.setGeometry(QtCore.QRect(880, 120, 51, 20))
        self.label_32.setObjectName("label_32")

        self.retranslateUi(Widget)
        QtCore.QMetaObject.connectSlotsByName(Widget)

    def retranslateUi(self, Widget):
        _translate = QtCore.QCoreApplication.translate
        Widget.setWindowTitle(_translate("Widget", "Widget"))
        self.label.setText(_translate("Widget", "S3"))
        self.label_2.setText(_translate("Widget", "Create Bucket"))
        self.label_3.setText(_translate("Widget", "Bucket Name"))
        self.label_4.setText(_translate("Widget", "Region"))
        self.label_5.setText(_translate("Widget", "List Buckets"))
        self.label_6.setText(_translate("Widget", "Region"))
        self.label_7.setText(_translate("Widget", "Upload/Download  File"))
        self.label_8.setText(_translate("Widget", "File Name"))
        self.label_9.setText(_translate("Widget", "Bucket Name"))
        self.pushButton.setText(_translate("Widget", "Upload"))
        self.pushButton_2.setText(_translate("Widget", "Download"))
        self.pushButton_3.setText(_translate("Widget", "List"))
        self.pushButton_4.setText(_translate("Widget", "Create"))
        self.label_11.setText(_translate("Widget", "Delete Bucket"))
        self.label_12.setText(_translate("Widget", "Bucket Name"))
        self.pushButton_5.setText(_translate("Widget", "Delete"))
        self.label_13.setText(_translate("Widget", "Start/Stop instance"))
        self.pushButton_6.setText(_translate("Widget", "Go"))
        self.label_14.setText(_translate("Widget", "List region names"))
        self.label_15.setText(_translate("Widget", "Get instance status"))
        self.label_16.setText(_translate("Widget", "ImageID"))
        self.pushButton_7.setText(_translate("Widget", "Go"))
        self.label_19.setText(_translate("Widget", "EC2"))
        self.label_17.setText(_translate("Widget", "Keyname"))
        self.pushButton_8.setText(_translate("Widget", "Go"))
        self.label_18.setText(_translate("Widget", "Statistic"))
        self.label_20.setText(_translate("Widget", "Max Count"))
        self.pushButton_9.setText(_translate("Widget", "Go"))
        self.label_21.setText(_translate("Widget", "Action (ON/OFF)"))
        self.label_22.setText(_translate("Widget", "Region"))
        self.label_23.setText(_translate("Widget", "Monitor instance"))
        self.label_24.setText(_translate("Widget", "End time"))
        self.label_25.setText(_translate("Widget", "Instance ID"))
        self.label_26.setText(_translate("Widget", "InstanceID"))
        self.label_27.setText(_translate("Widget", "Create Instance"))
        self.label_28.setText(_translate("Widget", "Start time"))
        self.label_29.setText(_translate("Widget", "Metric"))
        self.label_30.setText(_translate("Widget", "Min Count"))
        self.label_31.setText(_translate("Widget", "Period"))
        self.pushButton_10.setText(_translate("Widget", "Go"))
        self.label_f.setText(_translate("Widget", "File Name"))
        self.label_32.setText(_translate("Widget", "Region"))


#=============S3 ==============

def create_bucket():
    bucket_name = ui.lineEdit.text()
    region = ui.lineEdit_2.text()
    #create bucket
    try:
        if region is None:
            S3_client = boto3.client('s3')
            S3_client.create_bucket(bucket=bucket_name)

        else:
            S3_client= boto3.client('s3', region_name=region)
            location = {'LocationConstraint':region}
            S3_client.create_bucket(Bucket=bucket_name,
                                    CreateBucketConfiguration=location)


    except ClientError as e:
        logging.error(e)
        return False
    return True
print("Bucket created")

def list_bucket():
    region = ui.lineEdit_3.text()
    s3 = boto3.client("s3")
    blist = ""
    for bucket in s3.list_buckets()["Buckets"]:
        if s3.get_bucket_location(Bucket=bucket['Name'])['LocationConstraint'] == r
            blist = blist + "\n" + bucket["Name"]
            print(bucket["Name"])
    ui.textEdit.insertPlainText(blist)



def download_file():
    s3 = boto3.resource('s3')
    bucket_name = ui.lineEdit_6.text()
    file_name = ui.lineEdit_5.text()
    #f= '/Users/hunun/OneDrive/Documents/LTU Courses/CLOUD/lab3/AWS-Lab3/'
    #file_name2 = f + file_name
    s3.Bucket(bucket_name).upload_file(file_name, file_name)


print("file downloaded")
def upload_file():
    object_name=None
    bucket_name = ui.lineEdit_6.text()
    file_name = ui.lineEdit_5.text()
    # If S3 object_name was not specified, use file_name
    if object_name is None:
        object_name = os.path.basename(file_name)
        #file_name = '/Users/hunun/OneDrive/Documents/LTU Courses/CLOUD/lab3/AWS-La
        #basename = os.path.basename(file_name)


    # Upload the file
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(file_name, bucket_name, object_name)
        print("file uploaded")
    except ClientError as e:
        logging.error(e)
        return False
    return True


def delete_file():
    bucket_name = ui.lineEdit_4.text()
    file_name = ui.lineEdit_f.text()
    s3 = boto3.resource("s3")
    s3.Object(bucket_name, file_name).delete()

#print("file deleted")

#======EC2 functions=========================
def create_instance():
    region = ui.lineEdit_9917.text()
    imageID = ui.lineEdit_99288.text()
    keyname = ui.lineEdit_992.text()
    mincount = ui.lineEdit_993.text()
    mincount = int(mincount)
    maxcount = ui.lineEdit_9911.text()
    maxcount = int(maxcount)
    print(type(maxcount))
    ec2 = boto3.resource('ec2', region_name=region)
    keypair = ec2.create_key_pair(KeyName=keyname)
    # create a new EC2 instance
    #Microsoft Windows Server 2016 Base
    instances = ec2.create_instances(
        ImageId=imageID,
        MinCount=mincount,
        MaxCount=maxcount,
        InstanceType='t2.micro',
        KeyName=keyname,
    )
```

```
def Region_list():
    ec2 = boto3.client('ec2', region_name='us-west-1')
    rlist=''

# Retrieves all regions/endpoints that work with EC2
    response = ec2.describe_regions()
    print('Regions:', response['Regions'])
    rlist = rlist + str(response['Regions'])

# Retrieves availability zones only for region of the ec2 object
    response = ec2.describe_availability_zones()
    print('Availability Zones:', response['AvailabilityZones'])
    rlist = rlist + "\n\n" + str(response['AvailabilityZones'])

    ui.textEdit.insertPlainText(rlist)

def Instance_status():

    ec2 = boto3.resource('ec2', region_name='us-west-1')
    blist=''
    for status in ec2.meta.client.describe_instance_status()['InstanceStatuses']:
        print(status)
        blist = blist + "\n" + str(status)
    ui.textEdit.insertPlainText(blist)

def Start_stop():
    instance_id =  ui.lineEdit_997.text()
    action = ui.lineEdit_998.text()

    ec2 = boto3.client('ec2',region_name = "us-west-1")


    if action == 'ON':
    # Do a dryrun first to verify permissions
        try:
            ec2.start_instances(InstanceIds=[instance_id], DryRun=True)
        except ClientError as e:
            if 'DryRunOperation' not in str(e):
                 raise

    # Dry run succeeded, run start_instances without dryrun
    try:
        response = ec2.start_instances(InstanceIds=[instance_id], DryRun=False)
        print(type(response))
        ui.textEdit.insertPlainText(str(response))
    except ClientError as e:
        print(e)
    else:
    # Do a dryrun first to verify permissions
        try:
         ec2.stop_instances(InstanceIds=[instance_id], DryRun=True)
        except ClientError as e:
         if 'DryRunOperation' not in str(e):
            raise

    # Dry run succeeded, call stop_instances without dryrun
    try:
        response = ec2.stop_instances(InstanceIds=[instance_id], DryRun=False)
        print(type(response))
        ui.textEdit.insertPlainText(str(response))
    except ClientError as e:
        print(e)


def monitor_EC2_Instance():
    instanceID = ui.lineEdit_9914.text()
    metric = ui.lineEdit_9915.text()
    startTime = ui.lineEdit_9916.text()
    k = startTime.split(",")
    s1 = int(k[0])
    s2 = int(k[1])
    s3 = int(k[2])

    period = int(ui.lineEdit_9913.text())
    region = ui.lineEdit_9910.text()
    statistic = ui.lineEdit_999.text()
    endTime = ui.lineEdit_9912.text()
    y = endTime.split(",")
    y1 = int(y[0])
    y2 = int(y[1])
    y3 = int(y[2])

    client = boto3.client('cloudwatch',region_name=region)
    response = client.get_metric_statistics(
        Namespace='AWS/EC2',
        MetricName=metric,
        Dimensions=[
            {
                'Name': 'InstanceId',
                'Value': instanceID
            },
        ],
    StartTime=datetime(s1,s2,s3) - timedelta(seconds=600),
    EndTime=datetime(y1,y2,y3),
    Period=period,
    Statistics=[
    statistic,
    ],
    Unit='Percent'
    )

    for k, v in response.items():
        if k == 'Datapoints':
            for y in v:
                print(y['Average'])
                print(type(y['Average']))
    a = str(y['Average'])

    ui.textEdit.insertPlainText(a)


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Widget = QtWidgets.QWidget()
    ui = Ui_Widget()
    ui.setupUi(Widget)

    #====S3=====================================
    ui.pushButton_4.clicked.connect(create_bucket)
    ui.pushButton_3.clicked.connect(list_bucket)
    ui.pushButton.clicked.connect(download_file)
    ui.pushButton.clicked.connect(upload_file)
    ui.pushButton_5.clicked.connect(delete_file)


    #====EC2=====================================
    ui.pushButton_7.clicked.connect(create_instance)
    ui.pushButton_8.clicked.connect(Instance_status)
    ui.pushButton_9.clicked.connect(Region_list)
    ui.pushButton_10.clicked.connect(Start_stop)
    ui.pushButton_6.clicked.connect(monitor_EC2_Instance)
    Widget.show()
    sys.exit(app.exec_())
```

6 Bibliography

1.Amazon Web Services, Inc. (n.d.). Amazon CloudWatch FAQs - Amazon Web Services (AWS). [online] Available at: https://aws.amazon.com/cloudwatch/faqs/.