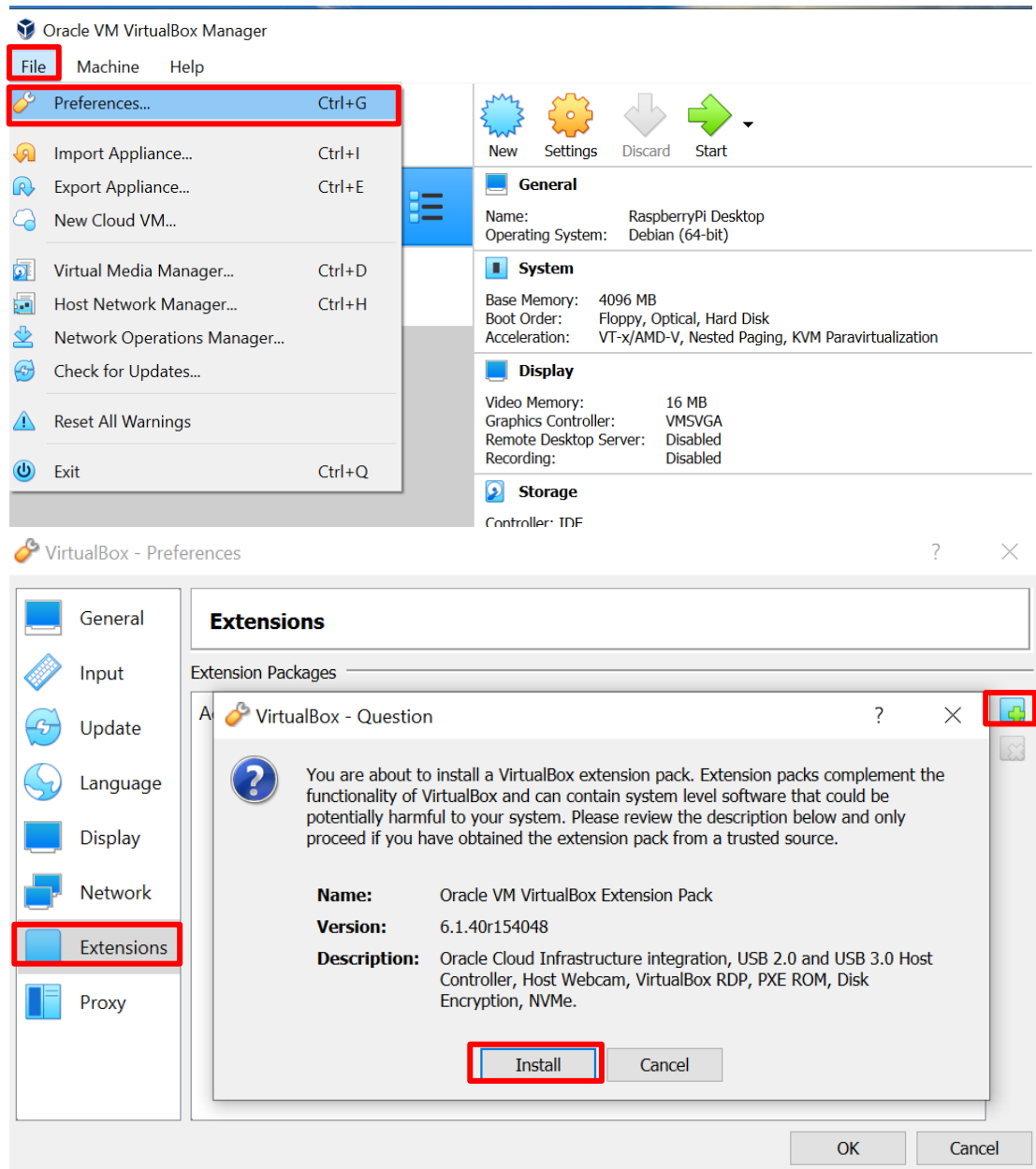
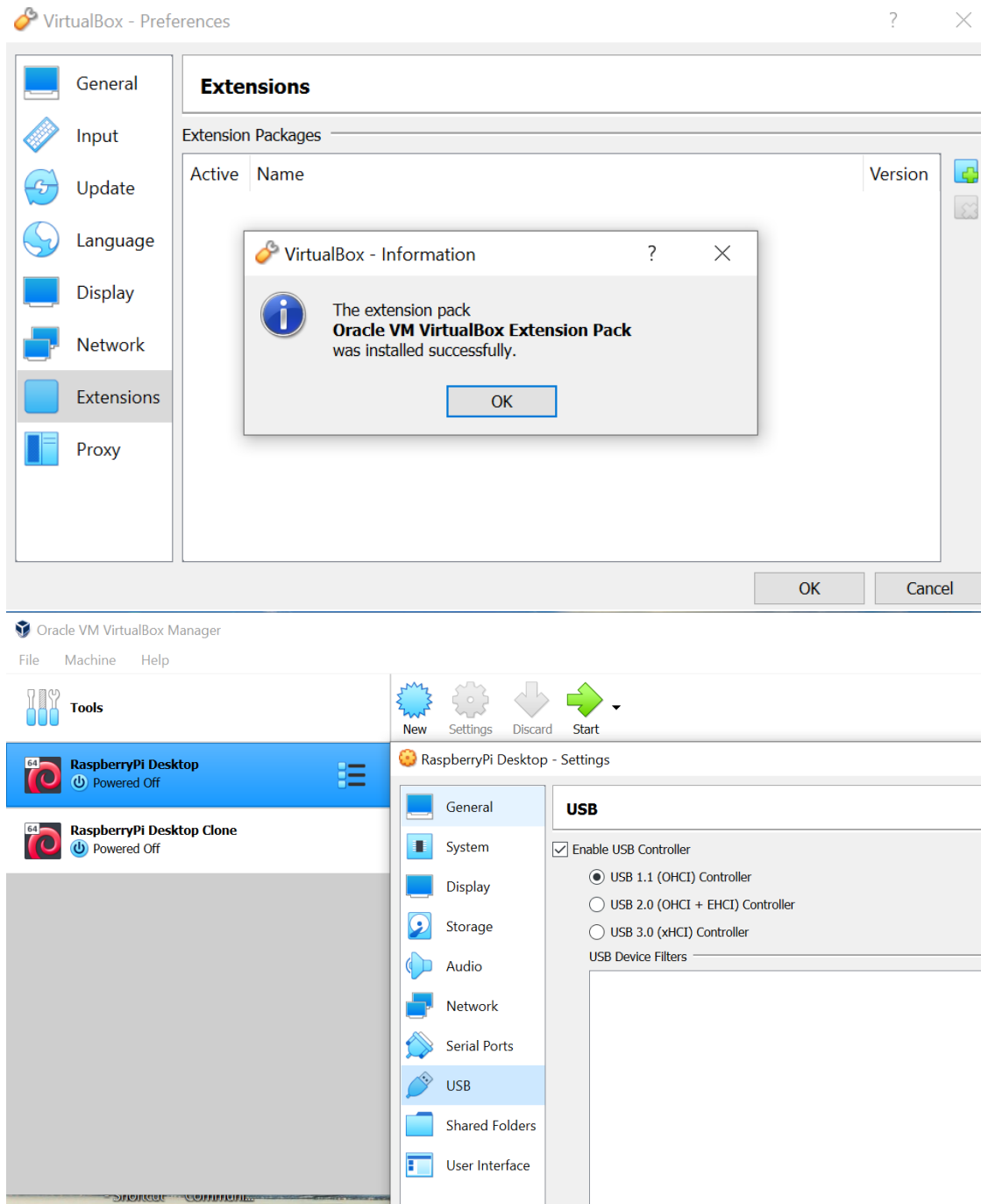


## Week12: Homework1: Facial Recognition on Raspberry Pi with AWS Rekognition

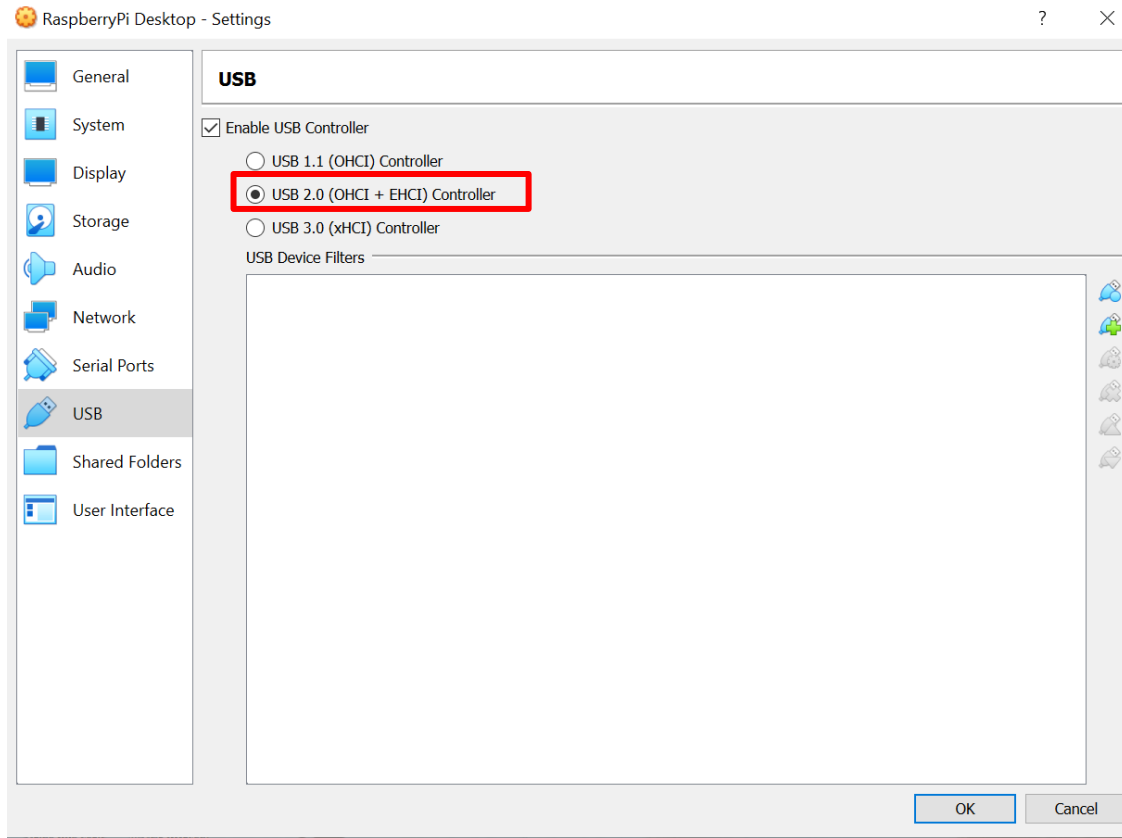
### 1. Install extension pack for VirtualBox

- Download extension pack for VirtualBox from this website [Downloads – Oracle VM VirtualBox](#)
- Open VirtualBox, follow the following instructions to install extension pack as below:





**Note: The virtual machine I am using is “RaspberryPi Desktop”. It is a Linux Debian(64bit) VM, it works fine.**



Reference link: [How to Install VirtualBox Extension Pack \(lifewire.com\)](https://lifecycle.com/How-to-Install-VirtualBox-Extension-Pack/)

## 2. Attach webcam to your virtual machine

In this project, we will use the integrated camera on my laptop.

### a. Start virtual machine "RaspberryPi Desktop"

Keep virtual machine "RaspberryPi Desktop" running

### b. Launch "Command Prompt" on your guest OS (means my laptop, windows 10)

```
C:\Users\root>cd "c:\Program Files\Oracle\VirtualBox"
```

```
c:\Program Files\Oracle\VirtualBox>VBoxManage list webcams
```

```
c:\Program Files\Oracle\VirtualBox>VBoxManage controlvm "RaspberryPi Desktop"
webcam attach .1
```

```

Command Prompt
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\root>cd "c:\Program Files\Oracle\VirtualBox"

c:\Program Files\Oracle\VirtualBox>VBoxManage list webcams
Video Input Devices: 1
.1 "Integrated Camera"
\\?\usb#vid_04f2&pid_b449&mi_00#6&293a28f6&0&0000#{65e8773d-8f56-11d0-a3b9-00a0c9223196}\global

c:\Program Files\Oracle\VirtualBox>VBoxManage controlvm "RaspberryPi Desktop" webcam attach .1
c:\Program Files\Oracle\VirtualBox>

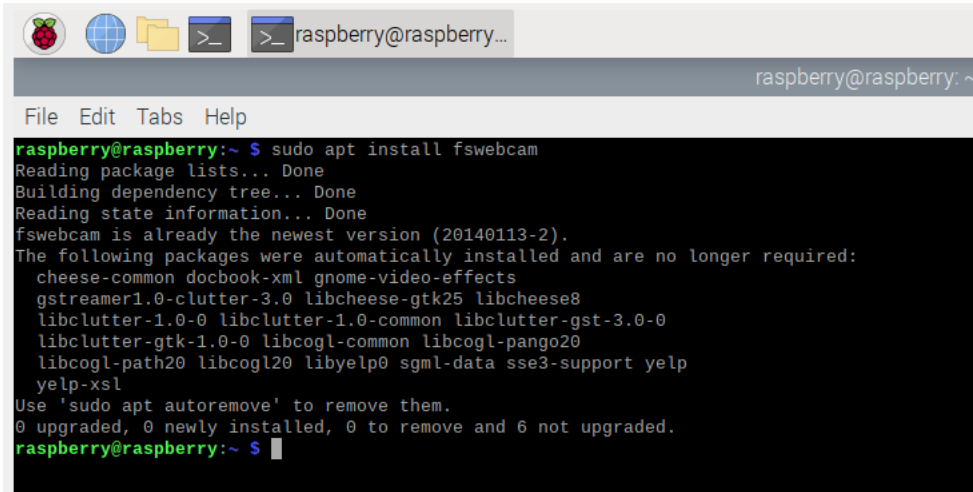
```

\* Replace "RaspberryPi DeskTop" with your virtual machine name.

- c. Back to the virtual machine “RaspberryPi Desktop” terminal, verify whether webcam works

Install fswebcam

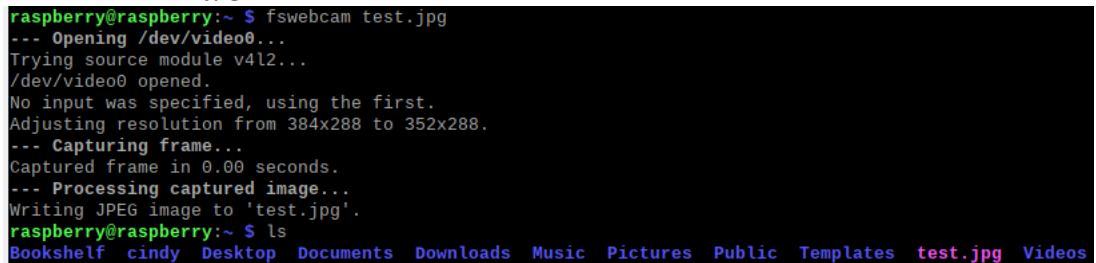
\$ sudo apt install fswebcam



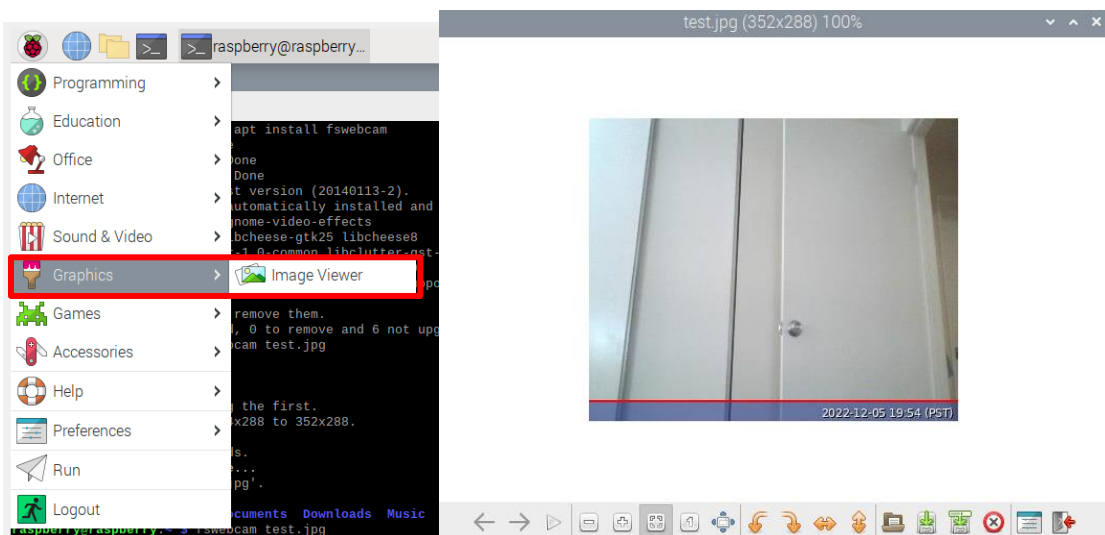
```
raspberrypi@raspberrypi: ~  
File Edit Tabs Help  
raspberrypi@raspberrypi:~ $ sudo apt install fswebcam  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
fswebcam is already the newest version (20140113-2).  
The following packages were automatically installed and are no longer required:  
  cheese-common docbook-xml gnome-video-effects  
  gstreamer1.0-clutter-3.0 libcheese-gtk25 libcheese8  
  libclutter-1.0-0 libclutter-1.0-common libclutter-gst-3.0-0  
  libclutter-gtk-1.0-0 libcogl-common libcogl-pango20  
  libcogl-path20 libcogl20 libyelp0 sgml-data sse3-support yelp  
  yelp-xsl  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.  
raspberrypi@raspberrypi:~ $
```

Test fswebcam

\$ fswebcam test.jpg



```
raspberrypi@raspberrypi:~ $ fswebcam test.jpg  
--- Opening /dev/video0...  
Trying source module v4l2...  
/dev/video0 opened.  
No input was specified, using the first.  
Adjusting resolution from 384x288 to 352x288.  
--- Capturing frame...  
Captured frame in 0.00 seconds.  
--- Processing captured image...  
Writing JPEG image to 'test.jpg'.  
raspberrypi@raspberrypi:~ $ ls  
Bookshelf cindy Desktop Documents Downloads Music Pictures Public Templates test.jpg Videos
```



### 3. Implementation

#### a. Install Boto3 and OpenCV

\$ pip install boto3

```
raspberrypi@raspberrypi:~$ pip install boto3
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting boto3
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProtocolError('Connection aborted.', RemoteDisconnected('Remote end closed connection without response'))': /simple/boto3/boto3-1.26.24-py3-none-any.whl
  Downloading https://www.piwheels.org/simple/boto3/boto3-1.26.24-py3-none-any.whl (129 kB)
    | 129 kB 168 kB/s
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading https://www.piwheels.org/simple/s3transfer/s3transfer-0.6.0-py3-none-any.whl (79 kB)
    | 79 kB 312 kB/s
Collecting jmespath<2.0.0,>=0.7.1
  Downloading https://www.piwheels.org/simple/jmespath/jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting botocore<1.30.0,>=1.29.24
  Downloading https://www.piwheels.org/simple/botocore/botocore-1.29.24-py3-none-any.whl (10.2 MB)
    | 10.2 MB 673 kB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore<1.30.0,>=1.29.24->boto3) (1.26.5)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore<1.30.0,>=1.29.24->boto3) (2.8.1)
Installing collected packages: jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.26.24 botocore-1.29.24 jmespath-1.0.1 s3transfer-0.6.0
raspberrypi@raspberrypi:~$
```

\$ pip install opencv-python

```
raspberrypi@raspberrypi:~$ pip install opencv-python
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting opencv-python
  Downloading opencv-python-4.6.0.66.tar.gz (90.3 MB)
    | 90.3 MB 304 bytes/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing wheel metadata ... done
Requirement already satisfied: numpy>=1.17.3 in /usr/lib/python3/dist-packages (from opencv-python) (1.19.5)
Building wheels for collected packages: opencv-python
  Building wheel for opencv-python (PEP 517) ... done
  Created wheel for opencv-python: filename=opencv_python-4.6.0.66-cp39-cp39-linux_i686.whl size=22028738 sha256=fa6c18e910b130c0174641b16acb9a5706199292db9545d29f80a12fa27a77bf
  Stored in directory: /home/raspberrypi/.cache/pip/wheels/6c/3a/b0/162197b99d01e5d1a44096c7392a6bf8ae182a4ee9a85ef9af
Successfully built opencv-python
Installing collected packages: opencv-python
Successfully installed opencv-python-4.6.0.66
raspberrypi@raspberrypi:~$
```

Verify that opencv-python is installed successfully

\$ python3

>>> import cv2

```
raspberrypi@raspberrypi:~$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import cv2
>>> quit
Use quit() or Ctrl-D (i.e. EOF) to exit
>>> quit()
raspberrypi@raspberrypi:~$
```

b. Create an AWS IAM user with all RekognitionFullAccess permission

## Add user



### Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

[+ Add another user](#)

### Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type\* ☒ **Access key - Programmatic access**  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **Password - AWS Management Console access**  
Enables a **password** that allows users to sign-in to the AWS Management Console.

[Users](#) > rekognition-tester

### Summary

[Delete user](#) [?](#)

User ARN `arn:aws:iam::032108903651:user/rekognition-tester` [🔗](#)

Path `/`



Creation time 2022-12-05 20:36 PST

[Permissions](#) [Groups](#) [Tags](#) [Security credentials](#) [Access Advisor](#)

▼ Permissions policies (2 policies applied)

[Add permissions](#)

[+ Add inline policy](#)

Policy name ▼	Policy type ▼	
Attached directly		
 AmazonRekognitionFullAccess	AWS managed policy	<a href="#">✕</a>
 AmazonS3FullAccess	AWS managed policy	<a href="#">✕</a>

**Remember that download .csv file which contains access Key ID and secret access key. We need to use this info in our program.**

- c. Create an AWS S3 bucket by clicking this link [S3 Management Console \(amazon.com\)](https://s3.amazonaws.com/)  
I uploaded 2 folders of 2 people's images. These images are used as reference images.

The screenshot shows the Amazon S3 Buckets console. On the left is a navigation menu with options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, Access analyzer for S3, Block Public Access settings, Storage Lens, Dashboards, and AWS Organizations settings. The main content area shows an 'Account snapshot' and a list of buckets. The 'Create bucket' button is highlighted with a red rectangle.

Name	AWS Region	Access	Creation date
mybucket-ml-sensor-data	US West (Oregon) us-west-2	Objects can be public	December 5, 2022, 14:20:21 (UTC-08:00)
mydata02	US East (N. Virginia) us-east-1	Objects can be public	December 5, 2022, 14:50:51 (UTC-08:00)
sagemaker-studio-032108903651-3rv49utut2e	US East (N. Virginia) us-east-1	Objects can be public	December 5, 2022, 14:54:31 (UTC-08:00)
sagemaker-studio-032108903651-az580kpnaur	US West (Oregon) us-west-2	Objects can be public	December 5, 2022, 13:41:56 (UTC-08:00)

The screenshot shows the Amazon S3 console for the 'bruceleeimages/' folder. It displays a list of three objects: brucelee-1.jpg, brucelee-2.jpg, and brucelee-3.jpg. Each object has a checkbox, a name, a type (jpg), a last modified date, a size, and a storage class (Standard).

Name	Type	Last modified	Size	Storage class
brucelee-1.jpg	jpg	December 7, 2022, 15:27:31 (UTC-08:00)	358.7 KB	Standard
brucelee-2.jpg	jpg	December 7, 2022, 15:27:34 (UTC-08:00)	763.4 KB	Standard
brucelee-3.jpg	jpg	December 7, 2022, 15:27:34 (UTC-08:00)	318.9 KB	Standard

The screenshot shows the Amazon S3 console for the 'jackiechanimages/' folder. It displays a list of two objects: jackiechan-1.jpg and jackiechan-2.png. Each object has a checkbox, a name, a type (jpg/png), a last modified date, a size, and a storage class (Standard).

Name	Type	Last modified	Size	Storage class
jackiechan-1.jpg	jpg	December 7, 2022, 15:28:37 (UTC-08:00)	48.6 KB	Standard
jackiechan-2.png	png	December 7, 2022, 15:28:37 (UTC-08:00)	189.5 KB	Standard

#### d. Implement Codes

For Indexing, create facial\_indexing.py:

```
import boto3

s3_client = boto3.client(
    's3',
    aws_access_key_id='AKIAQ060W7TR74PGM60U', # add the aws access key
    aws_secret_access_key='rIipUjLHWVT+a3+q4cGFYmWFkkaDr4VGyc5YXaVe', # add the aws secret access key
)

collectionId='facialindex' #collection name
rek_client=boto3.client('rekognition',
    aws_access_key_id='AKIAQ060W7TR74PGM60U', # add the aws access key
    aws_secret_access_key='rIipUjLHWVT+a3+q4cGFYmWFkkaDr4VGyc5YXaVe', # add the aws secret access key
    region_name='us-west-1',) # add the region here

Bucket = 'facialtestbucket' #S3 bucket name
all_objects = s3_client.list_objects_v2(Bucket =bucket )

#delete existing collection if it exists
list_response=rek_client.list_collections(MaxResults=2)
if collectionId in list_response['CollectionIds']:
    rek_client.delete_collection(CollectionId=collectionId)

#create a new collection
rek_client.create_collection(CollectionId=collectionId)

#add all images in current bucket to the collections #use folder names as the labels
for content in all_objects['Contents']:
    collection_name,collection_image =content['Key'].split('/')
    if collection_image:
        label = collection_name
        print('indexing: ',label)
        image = content['Key']
        index_response=rek_client.index_faces(CollectionId=collectionId,
            Image={'S3Object':{'Bucket':bucket,'Name':image}},
            ExternalImageId=label,
            MaxFaces=1,
            QualityFilter="AUTO",
            DetectionAttributes=['ALL'])
        print('FaceId: ',index_response['FaceRecords'][0]['Face']['FaceId'])
~
"facial_indexing.py" [dos] 39L, 1668B
```

For Matching, create a facial\_matching.py:

```
import time
import boto3
import cv2

# open camera
cap = cv2.VideoCapture(0)

collectionId='facialindex' #collection name
rek_client=boto3.client('rekognition',
    aws_access_key_id='AKIAQ060W7TR74PGM60U', # add the aws access key
    aws_secret_access_key='rIipUjLHWVT+a3+q4cGFYmWFkkaDr4VGyc5YXaVe', # add the aws secret access key
    region_name='us-west-1',)
~
n=1
while True:
    time.sleep(2)
    #milli = int(round(time.time() * 1000))
    # set dimensions
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 2560)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1440)
    image = 'image'+str(n)+'.jpg'
    n=n+1
    # take frame
    ret, frame = cap.read()
    # write frame to file
    cv2.imwrite(image, frame)

    print('captured '+image)
    with open(image, 'rb') as image:
        try: #match the captured images against the indexed faces
            match_response =rek_client.search_faces_by_image(CollectionId=collectionId, Image={'Bytes':image.read()}, MaxFaces=1, FaceMatchThreshold=85)
            if match_response['FaceMatches']:
                print('Hello, ',match_response['FaceMatches'][0]['Face']['ExternalImageId'])
                print('Similarity: ',match_response['FaceMatches'][0]['Similarity'])
                print('Confidence:',match_response['FaceMatches'][0]['Face']['Confidence'])
            else:
                print('No faces matched')
        except:
            print('No face detected')
            time.sleep(10)
~
"facial_matching.py" [dos] 40L, 1495B written
```

13,0-1 All



#### 4. Test and Run

First, run `facial_indexing.py` first

```
$ python3 facial_indexing.py
```

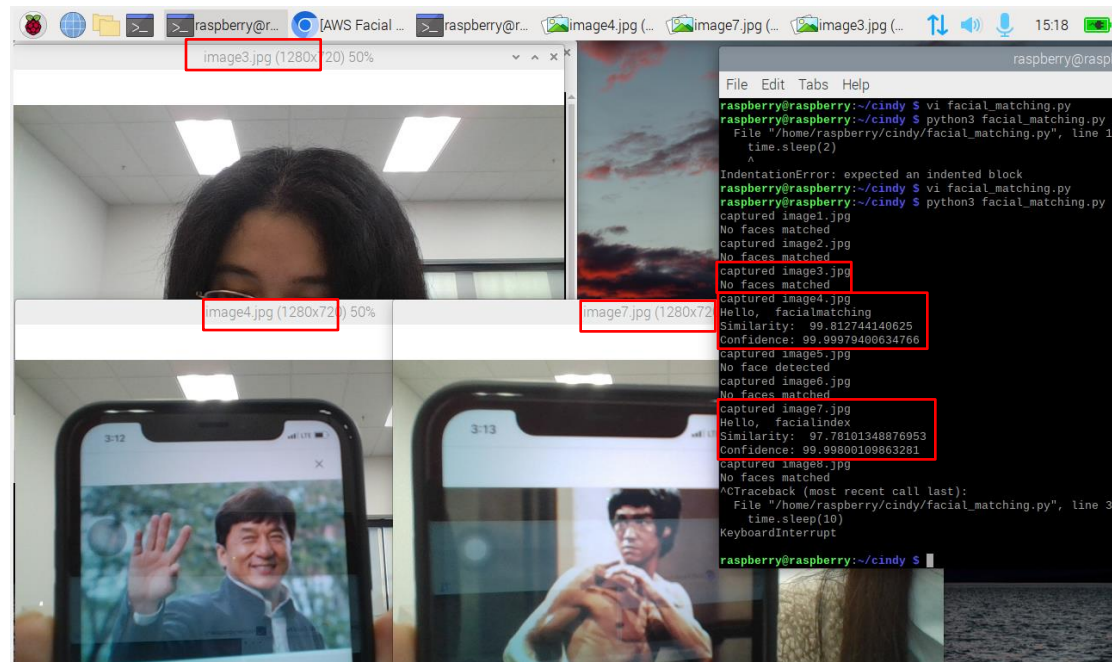
```

raspberrypi@raspberrypi:~/cindy $ vi facial_indexing.py
raspberrypi@raspberrypi:~/cindy $ python3 facial_indexing.py
indexing: bruceleeimages
FaceId: 613b82c2-7c66-4a3d-9a62-070dac27893f
indexing: bruceleeimages
FaceId: 5318211e-9ca4-4e38-97cd-09bb50fbd16f
indexing: bruceleeimages
FaceId: 96a8a99e-ea38-4876-b365-33ff98df0dbe
indexing: jackiechanimages
FaceId: 3fc80101-4f20-4703-a449-3bfd5bcf6c59
indexing: jackiechanimages
FaceId: 2989990d-b8fb-4f8a-945c-13dbe2c06ada
raspberrypi@raspberrypi:~/cindy $

```

Second, run `facial_matching.py`

```
$ python3 facial_matching.py
```



Done!!!

Reference Link:

#### 4.6. Webcam Passthrough (oracle.com)

## [Enabling Webcam in VirtualBox Guest OS on Windows Host \(scribles.net\)](#)