Dockerizing the EP application

## AudioAnalysis & SyncTestDetails:

* Migrate the target framework of the AudioAnalysis & SyncTestDetails .Net application to .Net 6.0.
* Identify the breaking changes of the entire application considering that the host machine is Linux.
* Earlier the application was running as a windows task scheduler on the windows server, now since the host machine is LINUX , The application is now deployed as self-contained docker image where it will run as self-configurable service .
* To schedule the frequency of the application need to update the Delay in appsettings.json,   
    
  In the below example we have set the delay to 60 seconds means for every 60 seconds the application will be triggered.

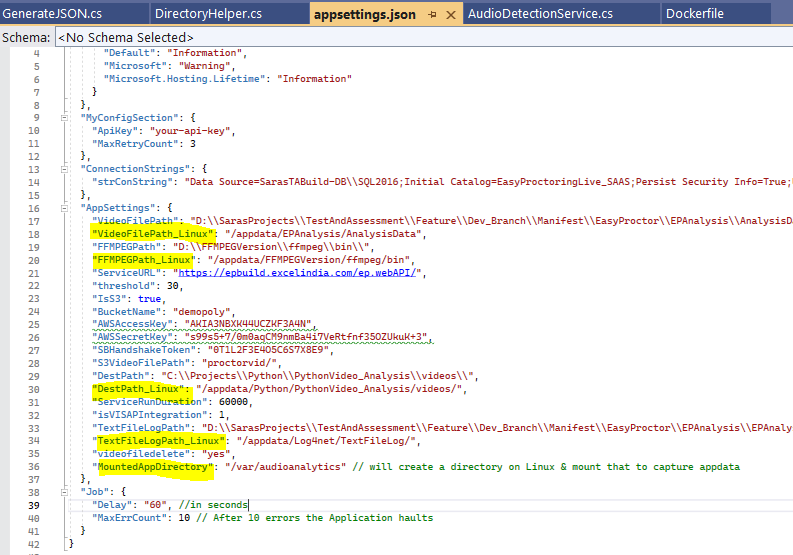
"Job": {

"Delay": "60", //in seconds

"MaxErrCount": 10 // After 10 errors the Application haults

}

* Specify the Linux specify path in the appsettings.json



* MountedAppDirectory represents the directory will be created to capture the appdata.
* Nuget packages of the application is updated to make it compatible with .Net 6.0

**Commands to Dockerize the AudioAnalysis & SyncTestDetails application.**

**Pre-requisites in windows :**

* Install latest version of Docker Desktop

**In Windows:**

**AudioAnalysis**

* Open the Audio Analysis (.Net 6.0) solution in Visual Studio Right click on the project & Add -> Docker Support & select Linux as Target OS & click OK.
* It will generate a DockerFile where the solution is located.
* Navigate to the directory where the Dockerfile is located & from CMD execute

docker build -t audioanalysis .

**Tagging & pushing to Docker hub :**

docker tag audioanalysis esdockerhub.excelindia.com/ audioanalysis:latest

docker push esdockerhub.excelindia.com/ audioanalysis:latest

**To Export the Docker Image as a tar file on Windows :**

docker save -o AudioAnalysis audioanalysis

* The above command will export the Docker Image as Tar file to the current directory where the command prompt is opened.

**SyncTestDetails:**

* Open the SyncTestDetails (.Net 6.0) solution in Visual Studio Right click on the project & Add -> Docker Support & select Linux as Target OS & click OK.
* It will generate a DockerFile where the solution is located.
* Navigate to the directory where the Dockerfile is located & from CMD execute

docker build -t synctestdetails .

**Tagging & pushing to Docker hub :**

docker tag synctestdetails esdockerhub.excelindia.com/ synctestdetails:latest

docker push esdockerhub.excelindia.com/synctestdetails:latest

**To Export the Docker Image as a tar file on Windows :**

docker save -o SyncTestDetails synctestdetails

* The above command will export the Docker Image as Tar file to the current directory where the command prompt is opened.

**To run the application on the host machine Linux(Ubuntu) :**

**Pre-requisites in windows :**

* Install latest version of Docker on Ubuntu (Host machine)

Below are the commands to be executed on the terminal :

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

Add the Docker repository:

For x86\_64/amd64 systems:

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Replace $(lsb\_release -cs) with your Ubuntu version, like "bionic" for 18.04 or "focal" for 20.04.

Update the package database again to include the Docker packages:

sudo apt update

Install Docker:

sudo apt install docker-ce

Start and enable the Docker service:

sudo systemctl start docker

sudo systemctl enable docker

Verify that Docker is installed and running by running the following command:

sudo docker –version

**To run the docker image :**

It involves 2 ways one is to copy paste the tar file extracted in the earlier step or pull the docker image from the docker hub.

**AudioAnalysis(Exported as Tar)**

* copy paste the tar file created on windows to Linux machine
* Import the Docker Image on Linux:

Open the terminal where the tar file is pasted & execute

sudo docker load -i AudioAnalysis

* Verify the Imported Image:

sudo docker images

* To run an image

sudo docker run -v /var/audioanalysis:/appdata -d --name AudioAnalysis audioanalysis

**SyncTestDetails (Exported as Tar)**

* copy paste the tar file created on windows to Linux machine
* Import the Docker Image on Linux:

Open the terminal where the tar file is pasted & execute

sudo docker load -i SyncTestDetails

* Verify the Imported Image:

sudo docker images

* To run an image

sudo docker run -v /var/synctestdetails:/appdata -d --name SyncTestDetails synctestdetails

## CQI\_PythonVideoAnalyis:

**Code changes in the script :**

* CQI\_PythonVideoAnalyis.py

import platform

# Get the operating system name

os\_name = platform.system()

print(f"Detected operating system: {os\_name}")

# Conditionally import Config\_Linux as cfg if OS is Linux, otherwise import Config as cfg

if os\_name == 'Linux':

import Config\_Linux as cfg

else:

import Config as cfg

* clone config.py & Create Config\_Linux.py & set the Linux Specific Path

Folder= {

"Path": "/appdata/"

},

Headshot= {

"Path": "/appdata/"

},

* created a starter.py to call main script recursively

import subprocess

import time

import os

import platform

# Get the operating system name

os\_name = platform.system()

# Conditionally import Config\_Linux as cfg if OS is Linux, otherwise import Config as cfg

if os\_name == 'Linux':

import Config\_Linux as cfg

else:

import Config as cfg

lock\_file = "CQI\_PythonVideoAnalyis\_Latest\_Zoom\_Tokbox\_lock.lock"

delay = cfg.Job["Frequency"]

print(f"Delay between each iteration is : {delay} seconds")

while True:

# Check if the lock file exists, indicating that a process is running

if os.path.exists(lock\_file):

print("Another process is already running. Waiting...")

time.sleep(60) # Wait for 1 minute before checking again

continue

try:

# Create the lock file to indicate that the process is running

with open(lock\_file, "w") as lock:

pass

# Run script.py using subprocess

subprocess.run(["python", "CQI\_PythonVideoAnalyis\_Latest\_Zoom\_Tokbox.py"])

except Exception as e:

# Handle exceptions and log them

print(f"An exception occurred: {e}")

finally:

# Remove the lock file to release the lock

os.remove(lock\_file)

# Sleep for 5 minutes (300 seconds) before running again

time.sleep(delay)

* Create a Dockerfile in the root folder of python application.

# Use the official python image as the base image

FROM python:3.11

# Set the working directory to /app

WORKDIR /app

# Copy the current directory contents into the container

COPY . /app

# Install the required packages

RUN pip install -r requirements.txt

# Install libGL Libraries

RUN apt-get update && apt-get install ffmpeg libsm6 libxext6 -y

# Run the script when the container starts

CMD ["python", "starter.py"]

**In Windows :**

* Open the cmd in the directory where Dockerfile is located & execute

docker build -t cqi\_pythonvideoanalyis .

* Save it as Tar

Open the cmd in the directory where you want to save the image as tar file

docker save -o CQI\_pythonvideoanalyis cqi\_pythonvideoanalyis

The above command will export the Docker Image as Tar file to the current directory where the command prompt is opened.

**To run the application on the host machine Linux(Ubuntu) :**

**Pre-requisites in windows :**

* Install latest version of Docker on Ubuntu (Host machine) [ steps mentioned earlier]

**To run the docker image :**

It involves 2 ways one is to copy paste the tar file extracted in the earlier step or pull the docker image from the docker hub.

**cqi\_pythonvideoanalyis (Exported as Tar)**

* copy paste the tar file created on windows to Linux machine
* Import the Docker Image on Linux:

Open the terminal where the tar file is pasted & execute

sudo docker load -i CQI\_pythonvideoanalyis

* Verify the Imported Image:

sudo docker images

* To run an image

sudo docker run -v /var/pythonvideoanalyis:/app/appdata -d --name CQI\_pythonvideoanalyis cqi\_pythonvideoanalyis

## NGINX & DNS configuration on the host machine (LINUX) (To be setup before deploying the EP Web application) :

* **Pre-requisites in LINUX(Ubuntu) :**

NGINX

To install open the terminal & execute the below commands :

sudo apt update

sudo apt install nginx

To Start NGINX

sudo systemctl start nginx

Enable Nginx to Start on Boot:

To ensure that Nginx starts automatically when your system reboots, run:

sudo systemctl enable nginx

Check Nginx Status:

sudo systemctl status nginx

* **Configure Nginx:**

You need to create a configuration file for your reverse proxy. You can create a new configuration file in the /etc/nginx/sites-available/

**Execute in terminal**

sudo rm /etc/nginx/sites-available/default

sudo gedit /etc/nginx/sites-available/ep-config

\*\* make sure to install gedit on LINUX to execute the above command.

Add the following configuration related to EP web applications to the ep-config file

server {

listen 443 ssl;

server\_name epdocker.excelindia.com;

ssl\_certificate /etc/ssl/excelindia.crt;

ssl\_certificate\_key /etc/ssl/excelindia.key;

location /EP.Web/ {

proxy\_pass https://localhost:3050/EP.Web/;

}

location /ep.web/ {

proxy\_pass https://localhost:3050/EP.Web/;

}

location /EP.WebAPI/ {

proxy\_pass https://localhost:8031/;

proxy\_connect\_timeout 120s;

proxy\_read\_timeout 120s;

proxy\_send\_timeout 120s;

}

location /ep.webapi/ {

proxy\_pass https://localhost:8031/;

proxy\_connect\_timeout 120s;

proxy\_read\_timeout 120s;

proxy\_send\_timeout 120s;

}

location /EPLiveProctoring/ {

proxy\_pass https://localhost:3060/EPLiveProctoring/;

}

location /epliveproctoring/ {

proxy\_pass https://localhost:3060/EPLiveProctoring/;

}

location /SystemRedinessCheck/{

proxy\_pass https://localhost:3060/SystemRedinessCheck/;

}

location /OrganizationLogo/{

proxy\_pass https://localhost:3060/OrganizationLogo/;

}

# Additional SSL and security settings go here if needed

}

**\*\* Make sure that we have the ssl certificate files stored in the respective paths on the host machine.**

/etc/ssl/excelindia.crt

/etc/ssl/excelindia.key

* Create a symbolic link to enable the configuration:

sudo ln -s /etc/nginx/sites-available/ep-config /etc/nginx/sites-enabled/

* Test the Nginx configuration:

sudo nginx –t

* Restart Nginx:

After testing, restart Nginx to apply the changes:

sudo systemctl restart nginx

## EP.Web:

* Create nginx.conf at root level of EpWeb(UI)

server {

listen 443 ssl;

server\_name epdocker.excelindia.com;

ssl\_certificate /etc/nginx/ssl/excelindia.crt;

ssl\_certificate\_key /etc/nginx/ssl/excelindia.key;

root /var/www/web/EPWeb;

location / {

try\_files $uri $uri/ /index.html;

}

}

* create a Dockerfile at root level of EpWeb.App

FROM nginx

COPY excelindia.crt /etc/nginx/ssl/

COPY excelindia.key /etc/nginx/ssl/

COPY nginx.conf /etc/nginx/conf.d/

COPY . /var/www/web/EPWeb

EXPOSE 443

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

\*\* Make sure to place the excelindia.crt & excelindia.key at the current directory

**In Windows**

* Open cmd where the Dockerfile is located & execute .

docker build -t epwebui .

* To export the docker image as tar

docker save -o EPWebUI epwebui

This will save the exported docker image as tar file to the directory where the cmd is opened.

* Command to upload the docker image to Dockerhub account( only if we are using Dockerhub account)

docker tag epwebui esdockerhub.excelindia.com/epwebui:latest

docker push esdockerhub.excelindia.com/epwebui:latest

**In LINUX(Ubuntu)**

Copy & paste the tar file from windows to Linux(Host system) & open a terminal from that directory where the tar file is pasted & execute .

sudo docker load -i EPWebUI

sudo docker run -d -p 3050:443 --name EPWebUI epwebui

The above command will host the Ep.web application on the poet 3050. We can browse the application now.

[**https://localhost:3050/EP.Web/app/index.html#/**](https://localhost:3050/EP.Web/app/index.html#/)

## EPLiveProctoring :

**Dockerizing UI(EPLive) assuming we have dist folder**

* create a Dockerfile at same level of EPLiveProctoring

# Use official nginx image as the base image

FROM nginx

# Copy the SSL certificate and key into the container

COPY excelindia.crt /etc/nginx/ssl/

COPY excelindia.key /etc/nginx/ssl/

# Copy the Nginx configuration file with SSL settings

COPY nginx.conf /etc/nginx/conf.d/default.conf

# Copy the dist folder to the nginx html folder

COPY ./ /usr/share/nginx/html

# Expose port

EXPOSE 443

CMD ["nginx", "-g", "daemon off;"].

* Create nginx.conf at the same level

server {

listen 443 ssl;

server\_name epdocker.excelindia.com;

ssl\_certificate /etc/nginx/ssl/excelindia.crt;

ssl\_certificate\_key /etc/nginx/ssl/excelindia.key;

location / {

root /usr/share/nginx/html;

}

}

\*\* Make sure to place the excelindia.crt & excelindia.key at the current directory

**In Windows**

* Open cmd where the Dockerfile is located & execute .

docker build -t eplive.

* To export the docker image as tar

docker save -o EPLive eplive

This will save the exported docker image as tar file to the directory where the cmd is opened.

* Command to upload the docker image to Dockerhub account( only if we are using Dockerhub account)

docker tag eplive esdockerhub.excelindia.com/ eplive:latest

docker push esdockerhub.excelindia.com/ eplive:latest

**In LINUX(Ubuntu)**

Copy & paste the tar file from windows to Linux(Host system) & open a terminal from that directory where the tar file is pasted & execute .

sudo docker load -i EPLive

sudo docker run -d -p 3060:443 --name EPLive eplive

The above command will host the Ep.web application on the poet 3050. We can browse the application now.

[**https://localhost:3060/EPLiveProctoring/#/login**](https://localhost:3060/EPLiveProctoring/#/login)

## EPWebAPI :

* Code changes done in the application to resolve LINUX specific issues (Like Timezone Converter).

**Pre-requisites in windows :**

* Install latest version of Docker Desktop

**In Windows:**

* SSL Configuration .Net Web API

in startup.cs

using Microsoft.AspNetCore.HttpsPolicy;

app.UseHttpsRedirection();//inside configure method

In the Dockerfile(place the excelindia.crt & excelindia.key in the same directory where dockerfile is located)

EXPOSE 443

# Copy your SSL certificate and configure HTTPS

COPY excelindia.crt /etc/ssl/certs/

COPY excelindia.key /etc/ssl/private/

ENV ASPNETCORE\_URLS=https://+:443

ENV ASPNETCORE\_HTTPS\_PORT=443

ENV ASPNETCORE\_Kestrel\_\_Certificates\_\_Default\_\_Path=/etc/ssl/certs/excelindia.crt

ENV ASPNETCORE\_Kestrel\_\_Certificates\_\_Default\_\_KeyPath=/etc/ssl/private/excelindia.key

* Open the solution in Visual Studio Right click on the project EP.WebAPI & Add -> Docker Support & select Linux as Target OS & click OK.
* It will generate a DockerFile where the solution is located.
* Navigate to the directory where the Dockerfile is located & from CMD execute

docker build -t epcoreapi.

**Tagging & pushing to Docker hub :**

docker tag epcoreapi esdockerhub.excelindia.com/ epcoreapi:latest

docker push esdockerhub.excelindia.com/ epcoreapi:latest

**To Export the Docker Image as a tar file on Windows :**

docker save -o EPCoreAPI epcoreapi

* The above command will export the Docker Image as Tar file to the current directory where the command prompt is opened.

**In LINUX(Ubuntu)**

Copy & paste the tar file from windows to Linux(Host system) & open a terminal from that directory where the tar file is pasted & execute .

sudo docker load -i EPCoreAPI

sudo docker run --name EPCoreAPI -e ASPNETCORE\_ENVIRONMENT=Development -p 8031:443 -d epcoreapi

The above command will host the Ep.WebAPI application on the poet 8031. We can browse the application now.

**https://localhost:8031/Ep.WebAPI**