```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "fb054efb-c141-45dd-9274-17d34933da90",
   "metadata": {},
   "source": [
    "# Assignment1 : Containerization Vs. Serverless\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "ea1a76b3-60bc-42ec-84a6-c44abc041d4d",
   "metadata": {},
   "source": [
    " ## Description\n",
    " \n",
    " Assignment1 aims to provide a practical experience of the differences and trade-offs between serverless and container-based architectures. By working with both architectures, you will gain a deeper understanding of their strengths and limitations and be able to make an informed decision on which architecture best suits your application requirements. Furthermore, building and deploying a container for a data science application will give you an opportunity to practice with Docker and familiarize yourself with containerization concepts.\n",
    " \n",
    " First step is running a given program on a local host. By running the program locally, you can get familiar with how it works and identify any dependencies it has. Second, you will attempt to create a serverless function from the program, which will give you an idea of the limitations of serverless computing for this program. Finally, you will containerize the application to address the serverless limitations. Containerizing the application will provide more flexibility and control over the infrastructure, allowing you to customize the environment to suit the needs of the application."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "17513629",
   "metadata": {},
   "source": [
    "## Prerequisite:\n",
    "\n",
    "Note: Please make sure you have Docker installed on your local machine before continuing. You can download Docker Desktop using this link
```

```
[here](https://www.docker.com/products/docker-desktop/)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "07f9ae49",
   "metadata": {},
   "source": [
    "## Run and test a Sentiment Analysis program on your local machine\n",
    "For simplicity, a python script has been provided for you in this
assignment. The Python script is a sentiment analysis program that uses
NLTK library to calculate the sentiment score of an input text."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "4f406f13-4fbd-4705-b493-8c807959c4e4",
   "metadata": {},
   "outputs": [],
   "source": [
    "# sentiment_analysis.py\n",
    "\n",
    "import json\n",
    "import nltk\n",
    "from nltk.sentiment import SentimentIntensityAnalyzer\n",
    "\n",
    "def sentiment_analysis (text):\n",
    "\n",
    "    nltk.download('vader_lexicon')\n",
    "    sia = SentimentIntensityAnalyzer()\n",
    "    sentiment_scores = sia.polarity_scores(text)\n",
    "\n",
    "    if sentiment_scores['compound'] > 0:\n",
    "        response = \"Positive sentiment\"\n",
    "    elif sentiment_scores['compound'] < 0:\n",
    "        response = \"Negative sentiment\"\n",
    "    else:\n",
    "        response = \"Neutral sentiment\"\n",
    "    return response"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "35e99626-7f06-4ad6-ba74-21ad1506924b",
   "metadata": {},
```

```
   "source": [
    "You can call this function from your local machine with a text input
as follows:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "5c179a0f-5602-407b-bdda-9640389b7981",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Define the input text\n",
    "text = \"I really enjoyed the movie. The acting was great and the plot
was interesting.\"\n",
    "result= sentiment_analysis(text)\n",
    "result"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "09993990-0d0a-4cff-8631-33032169cd62",
   "metadata": {},
   "source": [
    " ## Create, deploy and test a serverless function\n",
    " \n",
    "Apply the required justifications on your code and deploy your program
as a serverless function using AWS Lambda. You can follow the guidelines
from assignment0 to create a new Lambda function and upload your code.
Remember to first click on 'Deploy', then 'Test' to configure your test
event."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "b56b15e2",
   "metadata": {},
   "outputs": [],
   "source": [
    "import json\n",
    "import nltk\n",
    "from nltk.sentiment import SentimentIntensityAnalyzer\n",
    "\n",
    "def lambda_handler(event, context):\n",
    "    def sentiment_analysis (text):\n",
```

```
    "\n",
    "        nltk.download('vader_lexicon')\n",
    "        sia = SentimentIntensityAnalyzer()\n",
    "        sentiment_scores = sia.polarity_scores(text)\n",
    "\n",
    "        if sentiment_scores['compound'] > 0:\n",
    "            response = \"Positive sentiment\"\n",
    "        elif sentiment_scores['compound'] < 0:\n",
    "            response = \"Negative sentiment\"\n",
    "        else:\n",
    "            response = \"Neutral sentiment\"\n",
    "        return response\n",
    "    result = sentiment_analysis(event['text'])\n",
    "    return {\n",
    "        'statusCode': 200,\n",
    "        'body': json.dumps(result)\n",
    "    }"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "2e41d2c1-f18d-4aa9-bffc-aca592b195d2",
   "metadata": {},
   "source": [
    "You can invoke your Lambda function directly by creating a JSON event
object and passing it as an argument to the function."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "cf1c3fc7",
   "metadata": {},
   "outputs": [],
   "source": [
    "{\n",
    "  \"text\": \"I really enjoyed the movie. The acting was great and the
plot was interesting.\"\n",
    "}"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "da86eb7c",
   "metadata": {},
   "source": [
```

```
    "1. Are you able to test this program on cloud? \n",
    "\n",
    "2. What error message do you get?\n",
    "\n",
    "When you test your function, you may encounter an error message indicating that the NLTK library is missing. The Sentiment analysis program has a dependency on the NLTK library that needs to be installed before running your program. To resolve the dependency issue, one way is to containerize the application using a docker container. "
   ]
  },
  {
   "attachments": {},
   "cell_type": "markdown",
   "id": "74bcfa0b",
   "metadata": {},
   "source": [
    "## How to containerize the Sentiment Analysis program?\n",
    "Containerization encapsulates all of the dependencies and configuration required to run your program, allowing you to easily deploy it to different environments without worrying about compatibility issues. To do so, you need to know how to build your own Docker image and push it to Docker Hub. Why? Because you might want to make that image available to your teammates or even the rest of the world. To containerize the Sentiment Analysis program you need to walk through the following steps:\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9500ce10-ab6b-467a-981f-fd3cf16184bc",
   "metadata": {},
   "source": [
    "### 1-Create a Dockerfile: \n",
    "To containerize an application with dependencies, you will need to create a Dockerfile that specifies the image to use, any dependencies required, and the commands to run the application.\n",
    "First, let’s create a directory to work in with the command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "c1618c7a",
   "metadata": {},
   "outputs": [],
   "source": [
```

```
    "mkdir ~/DOCKER"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "8dc0e96a",
   "metadata": {},
   "source": [
    "Change into that directory with:\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "8c993442",
   "metadata": {},
   "outputs": [],
   "source": [
    "cd ~/DOCKER"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "11c45b7e",
   "metadata": {},
   "source": [
    "Now, we'll create our Dockerfile with the command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "787aefa4",
   "metadata": {},
   "outputs": [],
   "source": [
    "nano Dockerfile"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "b4098df2",
   "metadata": {},
   "source": [
    "Here is an example Dockerfile for a sentiment analysis application
that pulls down the python:3.7-slim-buster image, read the contents of a
```

```
new file (requirements.txt) and install everything listed in that file. "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "83dbbe7a",
   "metadata": {},
   "outputs": [],
   "source": [
    "FROM python:3.7-slim-buster\n",
    "\n",
    "WORKDIR /app\n",
    "\n",
    "COPY requirements.txt .\n",
    "\n",
    "RUN pip install --no-cache-dir -r requirements.txt\n",
    "\n",
    "COPY . .\n",
    "\n",
    "CMD [ \"python\", \"sentiment_analysis.py\" ]"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "62bf7523",
   "metadata": {},
   "source": [
    "This Dockerfile starts with the official Python 3.7-slim-buster image,
sets the working directory to /app, copies the requirements.txt file to the
container, installs the requirements, and then copies the application code
to the container. Finally, it specifies the command to run the
application.\n",
    "\n",
    "To create the requirements.txt file, issue the command:\n",
    "\n",
    "\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "18e34cf4",
   "metadata": {},
   "outputs": [],
```

```
   "source": [
    "nano requirements.txt"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0369a060",
   "metadata": {},
   "source": [
    "Make sure to include all of the necessary dependencies, such as the
NLTK library and any other packages required by the program. You can create
a requirements.txt file with the following contents:\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a7fd48bf",
   "metadata": {},
   "outputs": [],
   "source": [
    "nltk==3.6.2"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "a8f2b4f1-b928-4737-b703-3e38f44893b2",
   "metadata": {},
   "source": [
    "## 2- Build and run the Docker image\n",
    "\n",
    "To build the Docker image, navigate to the directory containing the
Dockerfile and run the following command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "d2cc3c33",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker build -t sentiment-analysis ."
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "bf74ca6d-871b-4308-9ce2-6082897db076",
   "metadata": {},
   "source": [
    "\n",
    "This will build the Docker image with the tag sentiment-analysis.\n",
    "\n",
    "Once the image is built, you can run it using the following command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "d0ec8ad2",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker run sentiment-analysis"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "3952994d",
   "metadata": {},
   "source": [
    "This will run the sentiment analysis application in the container."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c7b8731f-4994-4ac0-9dcc-b041d1a24ce8",
   "metadata": {},
   "source": [
    "(For Mac users): For users running Apple M1, M2 or M3 chip, you may
encounter this docker warning: WARNING: The requested image's platform
(linux/arm64/v8) does not match the detected host platform (linux/amd64/v3)
and no specific platform was requested. As a solution, you may need to
specify your build command by adding \"--platform linux/amd64\"\n",
    "\n",
    "Example:\n",
    "- docker build  --platform linux/amd64 -t
image-classification:latest\n",
    "- docker run --platform linux/amd64 image-classification:latest\n",
    "\n",
    "Additionally, the setting \"Use Rosetta for x86/amd64 emulation on
Apple Silicon\" on Docker Desktop should be turned on."
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "90a96e69-8d2c-4370-b995-bf98f873f2fb",
   "metadata": {},
   "source": [
    "## 3- Push the Docker image to Docker hub \n",
    "\n",
    "Push the Docker image to a container registry, such as Docker Hub (or Amazon ECR). Make sure you have a Docker Hub account. If you don't have one, you can create one for free at https://hub.docker.com/.\n",
    "\n",
    "Log in to Docker Hub using the Docker command line interface (CLI) by running the following command and entering your Docker Hub username and password when prompted:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "05d73d41",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker login"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "6a837be4",
   "metadata": {},
   "source": [
    "You can find the image ID of a Docker image by running the following command in the terminal:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "b36e84be",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker images"
   ]
```

```
  },
  {
   "cell_type": "markdown",
   "id": "20c953c7",
   "metadata": {},
   "source": [
    "This command lists all the Docker images on your system along with their repository, tag, and image ID. Look for the image you just created and note down its ID, which should be listed in the second column under the IMAGE ID heading. The ID is a long string of characters and numbers that uniquely identifies the image."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "8250b6d3",
   "metadata": {},
   "source": [
    " Tag your Docker image with your Docker Hub username and repository name using the following command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "6957445f",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker tag <image-id> <docker-hub-username>/<repository-name>:<tag>"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e439186f",
   "metadata": {},
   "source": [
    "Replace image-id with the ID of the Docker image you want to push, docker-hub-username with your Docker Hub username, repository-name with the name of the repository you want to push the image to (e.g., sentiment-analysis), and tag with a tag for the image (e.g., latest).\n",
    "\n",
    "For example, if your Docker Hub username is maryam, your repository name is cloudcomputing, and your image ID is b8e96b32bd12, the command would be:"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "b488b8d9",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker tag b8e96b32bd12 maryam/sentimentanalysis:latest"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e0a1bc48-0af5-42e2-a07c-c6a5a2e851fc",
   "metadata": {},
   "source": [
    "Push the Docker image to Docker Hub using the following command:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "f789f4ce",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker push <docker-hub-username>/<repository-name>:<tag>"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "78bac33b",
   "metadata": {},
   "source": [
    "Replace docker-hub-username, repository-name, and tag with the same
values you used in the previous step.\n",
    "\n",
    "For example:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "33e81b84",
   "metadata": {},
   "outputs": [],
```

```
   "source": [
    "docker push maryam/sentimentanalysis:latest"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "511599d6",
   "metadata": {},
   "source": [
    "After the push is complete, you can verify that the Docker image is
available on Docker Hub by visiting https://hub.docker.com/ and logging in
with your Docker Hub account. You should see the image listed under the
repository you pushed it to. \n",
    "\n",
    "That's it! Your Docker image for the sentiment analysis program
container is now available on Docker Hub and can be pulled and easily
deployed to a new machine/environment."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "dcf0b17f",
   "metadata": {},
   "source": [
    "To do so, you only need to pull the docker image from Docker Hub and
run it using the two following commands: "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "5c8fdf40",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker pull <docker-hub-username>/<repository-name>:<tag>"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "84668e72",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker run <docker-hub-username>/<repository-name>:<tag>"
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "1225d11b",
   "metadata": {},
   "source": [
    "##  4- Testing the portability of the container \n",
    "\n",
    "Your Docker image for the sentiment analysis program container is available on Docker Hub and can be pulled and deployed to a cloud platform like AWS, Azure or Google Cloud platform or any other environment. For simplicity, you can run your container on a virtual machine in Play_With_Docker to test the portability property of the container.\n",
    "\n",
    "Login into Play_With_Docker (https://labs.play-with-docker.com/). Play_With_Docker is a simple and interactive playground to learn Docker. It automatically creates a virtual machine that lets you run and test your docker images.\n",
    "\n",
    "Once you successfully started the virtual machine, you have to run the following commands:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "efdecd9e",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker login"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "1e4ec854",
   "metadata": {},
   "source": [
    "You will need the same credentials as docker hub to login."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "e0b4d184",
```

```
   "metadata": {},
   "outputs": [],
   "source": [
    "docker pull <docker-hub-username>/<repository-name>:<tag>"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e00e4f77",
   "metadata": {},
   "source": [
    "You may want to use your docker-hub-username, the repository-name and
the tag that you used for creating the docker image of sentiment analysis
program."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a658fcd7",
   "metadata": {},
   "outputs": [],
   "source": [
    "docker run <docker-hub-username>/<repository-name>:<tag>"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "e3777163",
   "metadata": {},
   "source": [
    "After running this command, your sentiment analysis program will be
executed successfully. If not, you have to revisit the previous steps to
make sure that you have created the docker image correctly. "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "6ba32137",
   "metadata": {},
   "source": [
    "Congratulations! You have now successfully deployed the container of a
sentiment analysis application. "
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "298cf56a",
   "metadata": {},
   "source": [
    "## 5- Create Lambda function with ECR image\n",
    "\n",
    "Once you created and tested your container successfully, you are able to deploy your container in serverless architecture according to the following steps:\n",
    "\n",
    "### Step 1. Create local docker image"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "d6d37ad7",
   "metadata": {},
   "source": [
    "`Dockerfile`"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "0046e7e7",
   "metadata": {},
   "outputs": [],
   "source": [
    "FROM public.ecr.aws/lambda/python:3.8\n",
    "\n",
    "COPY sentiment_analysis_lambda.py ${LAMBDA_TASK_ROOT}\n",
    "COPY requirements.txt .\n",
    "\n",
    "RUN pip install -r requirements.txt\n",
    "\n",
    "COPY nltk_data /nltk_data\n",
    "\n",
    "CMD [ \"sentiment_analysis_lambda.lambda_handler\" ]"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "2f71ed01",
   "metadata": {},
   "source": [
    "`requirements.txt`"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "05cd6a52",
 "metadata": {},
 "outputs": [],
 "source": [
  "boto3==1.20.0\n",
  "nltk==3.6.3"
 ]
},
{
 "cell_type": "markdown",
 "id": "59127fad",
 "metadata": {},
 "source": [
  "`sentiment_analysis_lambda.py`"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "28dd5b95",
 "metadata": {},
 "outputs": [],
 "source": [
  "import os\n",
  "import json\n",
  "import nltk\n",
  "from nltk.sentiment import SentimentIntensityAnalyzer\n",
  "\n",
  "nltk.data.path.append(\"/nltk_data\")\n",
  "\n",
  "def lambda_handler(event, context):\n",
  "    def sentiment_analysis (text):\n",
  "        sia = SentimentIntensityAnalyzer()\n",
  "        sentiment_scores = sia.polarity_scores(text)\n",
  "\n",
  "        if sentiment_scores['compound'] > 0:\n",
  "            response = \"Positive sentiment\"\n",
  "        elif sentiment_scores['compound'] < 0:\n",
  "            response = \"Negative sentiment\"\n",
  "        else:\n",
  "            response = \"Neutral sentiment\"\n",
```

```
    "        return response\n",
    "    result = sentiment_analysis(event['text'])\n",
    "    return {\n",
    "        'statusCode': 200,\n",
    "        'body': json.dumps(result)\n",
    "    }"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "2d036332",
   "metadata": {},
   "source": [
    "Download nltk_data to the same directory by running this python script:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "cabe0a7e",
   "metadata": {},
   "outputs": [],
   "source": [
    "import nltk\n",
    "nltk.download('vader_lexicon')"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "795c393e",
   "metadata": {},
   "source": [
    "Now, your directory should look like this:\n",
    "```\n",
    ".\n",
    "├── Dockerfile\n",
    "├── nltk_data\n",
    "│   └── sentiment\n",
    "│       └── vader_lexicon.zip\n",
    "├── requirements.txt\n",
    "└── sentiment_analysis_lambda.py\n",
    "```"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "1db58a32",
   "metadata": {},
   "source": [
    "Build docker image with command: `docker build -t
sentiment-analysis-lambda-image .`"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "2a40021b",
   "metadata": {},
   "source": [
    "### Step 2. Push docker image to ECR\n",
    "To push our Docker image to Amazon Elastic Container Registry
(ECR):\n",
    "\n",
    "1. Login to the AWS Management Console.\n",
    "1. Navigate to Elastic Container Registry and click on Create
repository.\n",
    "2. Name it `sentiment-analysis-lambda-image` and enable `Scan on push`
to create the repository.\n",
    "3. Follow instructions from `View push commands` shown on the ECR
repository page which will be similar to:\n",
    "- Authenticate Docker to the ECR registry\n",
    "- Tag your image\n",
    "- Push your image to the repository\n",
    "\n",
    "Note: Make sure you have properly configured your IAM Policy. You need
to ensure you have the “AmazonEC2ContainerRegistryFullAccess” policy which
provides full access to ECR"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "5e1cff55",
   "metadata": {},
   "source": [
    "### Step 3. Create Lambda function from ECR image\n",
    "1. Navigate to the Lambda service in the AWS Management Console.\n",
    "1. Click on Create function.\n",
    "1. Choose Container image as the source.\n",
    "1. Input `sentiment-analysis-lambda` as the function name.\n",
    "1. Under Container image URI, select the
`sentiment-analysis-lambda-image` from the ECR by clicking `Browse
images`.\n"
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "3d17fda8",
   "metadata": {},
   "source": [
    "### Step 4. Test Lambda function\n",
    "\n",
    "with input \n",
    "```\n",
    "{\n",
    "  \"text\": \"I really enjoyed the movie. The acting was great and the plot was interesting.\"\n",
    "}\n",
    "```\n",
    "output should be \n",
    "```\n",
    "{\n",
    "  \"statusCode\": 200,\n",
    "  \"body\": \"\\\"Positive sentiment\\\"\"\n",
    "}\n",
    "```"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9fd269c3",
   "metadata": {},
   "source": [
    "### Step 5. (Optional) Update Lambda function with new image\n",
    "1. Build new docker image with updated code.\n",
    "2. Push new docker image to ECR.\n",
    "3. Navigate to the Lambda service in the AWS Management Console.\n",
    "4. Click on `sentiment-analysis-lambda` function.\n",
    "5. Under `Image`, click on `Deploy new image`.\n",
    "6. Select the new image from the ECR by clicking `Browse images`.\n",
    "7. Click `Save`.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "31fd05e2",
   "metadata": {},
   "source": [
    "## 6- Create your own docker container and ECR image"
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "216133e6",
   "metadata": {},
   "source": [
    "Now, it is the time to create your own docker container and ECR image. You can choose one of the data science application that you have developed previously. Alternatively, you may want to use the following image classification application to complete your assignment. "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c8dbfc55",
   "metadata": {},
   "source": [
    "Here's a sample code for image classification using Convolutional Neural Networks (CNN). This code trains a CNN on the CIFAR-10 dataset, which consists of 50,000 32x32 color training images and 10,000 test images, each belonging to one of 10 classes."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "60b59392",
   "metadata": {},
   "outputs": [],
   "source": [
    "# image_classification.py\n",
    "\n",
    "import tensorflow as tf\n",
    "from tensorflow.keras import datasets, layers, models\n",
    "import matplotlib.pyplot as plt\n",
    "\n",
    "# Load the dataset\n",
    "(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()\n",
    "\n",
    "# Normalize pixel values to be between 0 and 1\n",
    "train_images, test_images = train_images / 255.0, test_images / 255.0\n",
    "\n",
    "# Define the CNN architecture\n",
```

```
    "model = models.Sequential()\n",
    "model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))\n",
    "model.add(layers.MaxPooling2D((2, 2)))\n",
    "model.add(layers.Conv2D(64, (3, 3), activation='relu'))\n",
    "model.add(layers.MaxPooling2D((2, 2)))\n",
    "model.add(layers.Conv2D(64, (3, 3), activation='relu'))\n",
    "model.add(layers.Flatten())\n",
    "model.add(layers.Dense(64, activation='relu'))\n",
    "model.add(layers.Dense(10))\n",
    "\n",
    "# Compile the model\n",
    "model.compile(optimizer='adam',\n",
    "
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
    "              metrics=['accuracy'])\n",
    "\n",
    "# Train the model\n",
    "history = model.fit(train_images, train_labels, epochs=10, \n",
    "                    validation_data=(test_images, test_labels))\n",
    "\n",
    "# Evaluate the model\n",
    "test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)\n",
    "print('Test accuracy:', test_acc)\n",
    "\n",
    "\n",
    "# Plot the training and validation accuracy over time\n",
    "plt.plot(history.history['accuracy'], label='accuracy')\n",
    "plt.plot(history.history['val_accuracy'], label = 'val_accuracy')\n",
    "plt.xlabel('Epoch')\n",
    "plt.ylabel('Accuracy')\n",
    "plt.ylim([0.5, 1])\n",
    "plt.legend(loc='lower right')\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0deb8edd",
   "metadata": {},
   "source": [
    "Once you choose your application, you will need to do the followings:
\n",
    "\n",
    "- Create a serverless function for the application. \n",
```

```
    "- Create a docker container for the application. \n",
    "- Compare the serverless and container-based architectures,
considering factors such as ease of deployment, cost, and complexity.\n",
    "\n",
    "\n",
    "\n",
    "## Deliverables:\n",
    "\n",
    "- A Dockerfile and a docker image of the image classification
application.\n",
    "- An output file containing the execution result of running the docker
container on local host.\n",
    "- An output file containing the execution result of running the docker
container on Play_With_Docker.\n",
    "- A report detailing the advantages and disadvantages of
containerization vs serverless, and a conclusion on which method to use for
the given application.\n",
    "\n",
    "\n",
    " "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "8722b349",
   "metadata": {},
   "source": [
    "# Good luck!"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0675f662",
   "metadata": {},
   "source": [
    "| Criteria             | SQL Database | Document DB  | Key-Value DB |
Column-Family DB | Graph DB     |\n",

"|----------------------|--------------|--------------|--------------|-----
--------------|-------------|\n",
    "| Data Volume          | Typically     | Scalable     | Scalable     |
Scalable          | Scalable    |\n",
    "|                      | Structured   | Structured   | Semi-        |
Semi-Structured   | Semi-       |\n",
    "|                      | and Relational| and          | structured   |
(Tabular)         | Structured  |\n",
```

```
    "|                      | (Tabular)    | Semi-        | (e.g., JSON) |
(e.g., JSON,     | (Graph)     |\n",
    "|                      |              | Structured   |              |
XML)             |             |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Access Patterns      | SQL Queries  | Query        | Simple       |
Simple Key/Value  | Graph Traversals |\n",
    "|                      | (Structured  | (Flexible)   | Get/Put      |
Get/Put           | Complex Query  |\n",
    "|                      | and Complex) |              | Operations   |
Operations        | Language      |\n",
    "|                      |              |              | Range        |
Range Queries     | (e.g., Gremlin) |\n",
    "|                      |              |              | Queries      |
                  |              |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Performance Needs    | High         | High         | High         |
High              | High        |\n",
    "|                      | (Transactional| (Flexible)   | (Simple)     |
(Simple)          | (Complex)   |\n",
    "|                      | and Complex) |              |              |
                  | (e.g., Real- |\n",
    "|                      |              |              |              |
                  | time Data)  |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Cost Considerations  | Licensing and| Licensing    | Licensing    |
Licensing and     | Licensing   |\n",
    "|                      | Hardware     | and Cloud    | and Cloud    |
Cloud (Variable)  | and Cloud   |\n",
    "|                      | Costs        | Costs        | Costs        |
Costs             | Costs       |\n",
    "|                      | Complexity   | Scalability  | Simplicity   |
Scalability       | Scalability |\n",
    "|                      | (Management  | and          | and          |
and               | and         |\n",
    "|                      | and          | Complexity   | Simplicity    |
Efficiency        | Efficiency  |\n",
    "|                      | Optimization)|              |              |
                |              |\n",
```

```
"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "3ff82223",
   "metadata": {},
   "source": [
    "| Criteria             | SQL Database | Document DB  | Key-Value DB |
Column-Family DB | Graph DB     |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Data Volume          | Typically     | Scalable     | Scalable     |
Scalable          | Scalable    |\n",
    "|                      | Structured   | Structured   | Semi-        |
Semi-Structured   | Semi-       |\n",
    "|                      | and Relational| and          | structured   |
(Tabular)         | Structured  |\n",
    "|                      | (Tabular)    | Semi-        | (e.g., JSON) |
(e.g., JSON,     | (Graph)     |\n",
    "|                      |              | Structured   |              |
XML)              |             |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Access Patterns      | SQL Queries  | Query        | Simple       |
Simple Key/Value  | Graph Traversals |\n",
    "|                      | (Structured  | (Flexible)   | Get/Put      |
Get/Put           | Complex Query  |\n",
    "|                      | and Complex) |              | Operations   |
Operations        | Language      |\n",
    "|                      |              |              | Range        |
Range Queries     | (e.g., Gremlin) |\n",
    "|                      |              |              | Queries      |
                  |              |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Performance Needs    | High         | High         | High         |
High              | High        |\n",
    "|                      | (Transactional| (Flexible)   | (Simple)     |
(Simple)          | (Complex)   |\n",
    "|                      | and Complex) |              |              |
                  | (e.g., Real- |\n",
```

```
    "|                      |             |              |             |
                | time Data)  |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n",
    "| Cost Considerations  | Licensing and| Licensing    | Licensing    |
Licensing and     | Licensing   |\n",
    "|                      | Hardware     | and Cloud    | and Cloud    |
Cloud (Variable)  | and Cloud   |\n",
    "|                      | Costs        | Costs        | Costs        |
Costs             | Costs       |\n",
    "|                      | Complexity   | Scalability  | Simplicity   |
Scalability       | Scalability |\n",
    "|                      | (Management  | and          | and          |
and               | and         |\n",
    "|                      | and          | Complexity   | Simplicity    |
Efficiency        | Efficiency  |\n",
    "|                      | Optimization)|              |              |
                |              |\n",

"|----------------------|-------------|-------------|-------------|-----
-------------|------------|\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "bbb13f9f",
   "metadata": {},
   "source": [
    "| Criteria             | Block Storage  | Object Storage | File
Storage   |\n",

"|----------------------|----------------|----------------|----------------
|\n",
    "| Data Volume          | Scalable       | Highly Scalable| Scalable
   |\n",
    "|                      | (Fixed Blocks) | (Unstructured)| (Structured)
   |\n",

"|----------------------|----------------|----------------|----------------
|\n",
    "| Access Patterns      | Direct Block   | HTTP REST API  | File System
   |\n",
    "|                      | Level Access   | (S3, Azure    | Protocol
(NFS,  |\n",
    "|                      | (Low-Level)    | Blob Storage,  | SMB, FTP)
```

```
   |\n",
    "|                      |                | Google Cloud   |
   |\n",
    "|                      |                | Storage)       |
   |\n",

"|----------------------|---------------|---------------|---------------
|\n",
    "| Performance Needs    | High I/O       | High           | Balanced
I/O   |\n",
    "|                      | Operations,   | Read-Intensive,| (Read and
   |\n",
    "|                      | Low Latency    | Write-Intensive| Write
Access)  |\n",

"|----------------------|---------------|---------------|---------------
|\n",
    "| Cost Considerations  | Licensing and  | Pay-as-You-Go  | Licensing,
   |\n",
    "|                      | Hardware Costs | Model with     | Hardware,
and  |\n",
    "|                      | (Variable)     | Data Storage  | Operating
   |\n",
    "|                      |                | Usage-based    | Costs
   |\n",
    "|                      |                | Pricing        |
   |\n",

"|----------------------|---------------|---------------|---------------
|\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "bce84f57",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
```

```
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```