```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "d2ab2199",
   "metadata": {},
   "source": [
    "# Lab3: Cloud Storage Services and Setup\n",
    "\n",
    "Maryam R.Aliabadi, August 7th, 2023"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "16c7018f",
   "metadata": {},
   "source": [
    "\n",
    "## Learning objectives\n",
    "\n",
    "The goal of this lab is to launch and configure an S3 bucket for use within your team. You will learn:\n",
    "\n",
    "- Setting up an S3 bucket\n",
    "- Moving data into your bucket\n",
    "- Reading data from your bucket"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "a2621106",
   "metadata": {},
   "source": [
    "\n",
    "## Storage\n",
    "\n",
    "- EBS - Elastic Block Store\n",
    "- EFS - Elastic File System\n",
    "- S3 -  Simple Storage Service\n",
    "\n",
    "\n",
    "### EBS\n",
    "Elastic Block Storage(EBS) is block storage, which means the disk utilization happens in blocks or data stored in blocks.  A ***file is stored in blocks***, so if we want to change one character in a one GB file, we just want to change the block that contains that one bit. This is
```

```
how data is stored, and it functions very similar to your laptop hard disk, flash drive, or any external disk. We use this as storage for our EC2 instance.\n",
    "\n",
    "So we can use EBS as a boot volume or attach it to an existing EC2 instance, just like how we connect an external hard disk to our laptop.\n",
    "\n",
    "```{note}\n",
    "EC2 instances need an Amazon EBS volume to boot.\n",
    "```\n",
    "\n",
    "Here are some of the properties of EBS\n",
    "\n",
    "- Persistent storage (non-volatile storage)\n",
    "- Automatically replicated within the availability zones \n",
    "- High availability and durability\n",
    "- More durability by backing up (taking snapshots) data to S3. \n",
    "- Low latency\n",
    "- Scale storage up or down\n",
    "- Data encryption \n",
    "\n",
    "### EFS\n",
    "Provides storage for EC2 instances; unlike EBS, EFS storage can be accessed by multiple EC2 instances simultaneously. EFS offers all the advantages that we mentioned with EBS but also provides the following on top of it\n",
    "\n",
    "- Auto-scaling \n",
    "- Replication on multiple availability zones\n",
    "- Sharing with other EC2 instance\n",
    "\n",
    "This is mainly used in industry to provide home directories for workers.\n",
    "\n",
    "```{note}\n",
    "EFS won't be replacing EBS; both have their use cases. For example, we need EBS storage for an EC2 instance as a boot drive. But, EFS, you can think more in a corporate environment performing centralized shared storage—uses like media processing or shared code repositories.\n",
    "```\n",
    "\n",

"[Here](https://medium.com/awesome-cloud/aws-difference-between-efs-and-ebs-8c0d72a348ad) is an article that shows the difference between EBS and EFS."
   ]
```

```
  },
  {
   "cell_type": "markdown",
   "id": "45070f5e",
   "metadata": {},
   "source": [
    "### S3\n",
    "\n",
    "S3 stands for Simple Storage Service. This is one of the first storage services provided by AWS in 2006. It is an object-level storage, where a ***file is stored as an entire object***, so if we want to change one character in a one GB file, we want to update the file, and then we have to replace that entire file. By the object storage, you can think of it as the ***Key-Value store***, where the key is the filename, and the Value is the contents of the object or the file itself. So you store an object with a Key and retrieve the object with the key. \n",
    "\n",
    "Summarizing some properties of S3:\n",
    "\n",
    "- Data is stored as objects in buckets\n",
    "- Virtually unlimited storage (Single object is limited to 5 TB)\n",
    "- Designed for 99.999999999% of durability\n",
    "- Granular access to buckets and objects\n",
    "\n",
    "You don't need to specify the availability zones, as AWS will take care of all these in replicating your data across different availability zones."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "087dec40",
   "metadata": {},
   "source": [
    "\n",
    "## Setting up S3 bucket\n",
    "\n",
    "Before attempting to read or upload to your S3 bucket, you should ensure that the bucket exist and have access to it. Due to the limitations of the student account, we cannot dig into the permissions and policies required for this. However, in our case, we can simply make the bucket public so that anyone, including your team members, can access it. \n",
    "\n",
    "Click the toggle below for instructions to create and setup your bucket and make it public:\n",
    "\n",
```

```
    "- Go to the S3 Service in AWS console.\n",
    "- Create a bucket here. (e.g.:I gave an example name
`mds-s3-14-gittu`).\n",
    "- Make sure AWS region is `us-west-2`.\n",
    "- When creating the root bucket, make it public. Make sure you
unchecked \"Block all public access\".\n",
    "- (Default encryption) Bucket key make sure you check radio button for
`disable`\n",
    "- All other options leave as it is. And click on `Create bucket`.\n",
    "- After your bucket is created make sure to change the policies of the
bucket. \n",
    "    - Click on your buckets\n",
    "    - Click on Permissions tab\n",
    "    - Edit bucket policy and paste the following policy and then Save
changes. Make sure you replace the bucket name with your bucket name, in my
case it is `mds-s3-14-gittu`."
   ]
  },
  {
   "cell_type": "raw",
   "id": "000b13de",
   "metadata": {},
   "source": [
    "{\n",
    "    \"Version\": \"2012-10-17\",\n",
    "    \"Id\": \"Policy1649284381437\",\n",
    "    \"Statement\": [\n",
    "        {\n",
    "            \"Sid\": \"Stmt1649284379371\",\n",
    "            \"Effect\": \"Allow\",\n",
    "            \"Principal\": {\n",
    "                \"AWS\": \"*\"\n",
    "            },\n",
    "            \"Action\": \"s3:*\",\n",
    "            \"Resource\": [\n",
    "                \"arn:aws:s3:::mds-s3-14-gittu\",\n",
    "                \"arn:aws:s3:::mds-s3-14-gittu/*\"\n",
    "            ]\n",
    "        }\n",
    "    ]\n",
    "}"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0b804540",
```

```
   "metadata": {},
   "source": [
    "### More about credentials\n",
    "\n",
    "You can create a folder in your S3 bucket and store any files in it. Since your bucket is now public, you and your team members can access it. Before moving on to the next session, please make sure you have correctly installed AWS CLI and updated your AWS credentials (as they change every time your lab restarts/starts) to `~/.aws/credentials`. You can find more details about this in our previous lecture [here](awscli).\n",
    "\n",
    "If you have already updated your credentials in `~/.aws/credentials`, you do not need to pass them explicitly anywhere. However, if you still want to pass them explicitly, I have included instructions in an optional section [here](explicitly).\n",
    "\n",
    "Here is the confusing part: I said you do not want to pass credentials explicitly if you have already updated them in ~/.aws/credentials. This is because, when you run the AWS CLI or any API, it looks for this credentials file in your home directory, finds it, and takes the credentials from there to execute. This process is straightforward when you are working on your laptop. However, when you work on a shared environment (like the EC2 instance we have), each user has their own home directory, and they must update it with their credentials in their home directory ~/.aws/credentials. When you access TLJH notebook, it does not execute the notebook as the same user in your Linux (maria is the user in unix), but as a different user (\"jupyter-maria\" is user in TLJH), such as \"jupyter-yourjupyterusername\" (in my case, \"jupyter-gittu\") or \"jupyter-maria\". Therefore, if you execute something from the Jupyter notebook in EC2, you want to make sure that your credentials are in the respective home directories. I have included a picture of all home directories to make this situation more understandable.\n",
    "\n",
    "Simply move credentials (here all users will be sharing the root user credentials) to a shared folder, and let aws know that this is now where the credentials are:\n",
    "\n",
    "1. Create a shared folder in the ```srv``` folder"
   ]
  },
  {
   "cell_type": "raw",
   "id": "86174e89",
   "metadata": {},
   "source": [
    "sudo mkdir -p /srv/keys"
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "814f449e",
   "metadata": {},
   "source": [
    "2.  Copy credentials to that folder and make it readable by other
(chmod)"
   ]
  },
  {
   "cell_type": "raw",
   "id": "5f36e915",
   "metadata": {},
   "source": [
    "sudo cp ~/.aws/credentials /srv/keys/\n",
    "sudo chmod -R 777 /srv/keys"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "b49de982",
   "metadata": {},
   "source": [
    "3. In your Jupyter Hub notebook file, add this cell:"
   ]
  },
  {
   "cell_type": "raw",
   "id": "d53221f3",
   "metadata": {},
   "source": [
    "os.environ[\"AWS_SHARED_CREDENTIALS_FILE\"] =
\"/srv/keys/credentials\""
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9815690c",
   "metadata": {},
   "source": [
    "\n"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "c4550f7d",
   "metadata": {},
   "source": [
    "## How to transfer data to S3 \n",
    "\n",
    "There are many ways you can put data into S3. You can do it
using...\n",
    "\n",
    "### Web interface \n",
    "\n",
    "You can upload tiny files using UI and bigger files using CLI. I will
show this in class.\n",
    "\n",
    "### SDK \n",
    "\n",
    "Using [pandas](pandass3) and [AWS wrangler](awswrangler).\n",
    "\n",
    "### Using CLI\n",
    "\n",
    "You can use AWS CLI what you installed previously to upload data. If
you want to upload a folder (or a parquet file) best is to use AWS CLI.\n"
   ]
  },
  {
   "cell_type": "raw",
   "id": "edc8672b",
   "metadata": {},
   "source": [
    "eg: aws s3 cp aws/ s3://mds-s3-gittu/output/ --recursive\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "64b8d886",
   "metadata": {},
   "source": [
    "Here in the above eg I am moving a folder named \"aws/\" to the S3
path \"s3://mds-s3-gittu/output/\" ; adding --recursive as it's a folder.
This is similar to the ```cp - R``` in linux."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "40a11d34",
   "metadata": {},
```

```
   "source": [
    "\n",
    "You can also use the AWS CLI for moving local data to AWS. Refer to
[this](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-com
mands.html#using-s3-commands-managing-objects-move) document."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "516af748",
   "metadata": {},
   "source": [
    "\n",
    "## How to read data from S3\n",
    "\n",
    "Popular data processing libraries like pandas and arrow offer built-in
features for retrieving data from Amazon S3 cloud storage. You can access
the data just as you would access it from your local computer by simply
adding \"s3://<path_to_file_in_s3>\" at the beginning of the file path.\n",
    "\n",
    "To enable this functionality, most of these libraries use either the
`boto3` or `s3fs` library in the background. For example, pandas use s3fs,
while awswrangler uses boto3. To make it pretty straightforward you can use
pandas to read data from S3 and I will be showing you that below. But it is
bit buggy, and if you get into any `forbidden error` [despite of making
sure that you made bucket public and updated credentials correctly in
`~/.aws/credentials` (remember your credential gets updated every time it
restarts)](credentials), then this could be with some issues with pandas
reader, if that is the case please switch to awswrangler. I am giving
instruction on how to use awswrangler at the [end as
optional](awswrangler).\n",
    "\n",
    "To start, let's install the necessary packages using pip. Then, we can
easily read files from S3 and start analyzing the data. If you want to see
how to install please check above section on How to install packages in
TLJH.\n"
   ]
  },
  {
   "cell_type": "raw",
   "id": "17c38c41",
   "metadata": {},
   "source": [
    "sudo -E pip install pandas\n",
    "sudo -E pip install s3fs\n",
    "sudo -E pip install pyarrow"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "211a878b",
   "metadata": {},
   "outputs": [],
   "source": [
    "import json\n",
    "import urllib.parse\n",
    "import pandas as pd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "7bef9264",
   "metadata": {
    "tags": [
     "skip-execution"
    ]
   },
   "outputs": [],
   "source": [
    "%%time\n",
    "df = pd.read_parquet(\"s3://testmds/combined_data.parquet\", \n",
    "    "        filters=[('year','=', 2004)], \n",
    "    "                columns=['year', 'UniqueCarrier', 'ArrDelay'])\n",
    "print(df[(df.year== 2004) & (df.ArrDelay >
10)][\"UniqueCarrier\"].value_counts())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "bac4713f",
   "metadata": {
    "tags": [
     "skip-execution"
    ]
   },
   "outputs": [],
   "source": [
    "## just getting 1000 rows to save time\n",
    "df = pd.read_csv(\"s3://testmds/combined_data.csv\",nrows = 1000)\n",
```

```
    "print(df[(df.year== 1996) & (df.ArrDelay >
10)][\"UniqueCarrier\"].value_counts())"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c4a02dc9",
   "metadata": {},
   "source": [
    "## S3 Glacier (optional)\n",
    "\n",
    "Amazon S3 glacier is an extremely low-cost cloud service for long-term
backup (data archiving service). The drawback is that it takes several
hours to retrieve the stored data and should only be used for archiving.
The retrieval time depends on the retrieval option that the user goes for.
There are mainly 3 retrieval options...\n",
    "\n",
    "- Bulk (5 - 12 hours)\n",
    "- Standard (3- 5 hours)\n",
    "- Expedited (1 - 5 minutes)\n",
    "\n",
    "Cost increases as the speed of retrieval increases.\n",
    "\n",
    "There are 3 main elements of glacier...\n",
    "\n",
    "- Archive\n",
    "All files (Any object such as a photo, video file, or document) should
be zipped before uploading to the glacier. This is considered the base unit
of storage. All have a unique ID and also have a description \n",
    "- Vault\n",
    "It is a container for storing archives, and you can specify the region
where you want to locate your vault. \n",
    "- Access policies\n",
    "This is how you control access to the vault. You can give permissions
to individuals and what kind of operation they can do.\n",
    "\n",
    "Can you think of use-cases for storing data in a glacier?\n",
    "\n",
    "In industries, data usually follows a lifecycle, and we can achieve
this life cycle using S3 and glacier. \n",
    "\n",

"[Here](https://aws.amazon.com/blogs/storage/aws-reinvent-recap-best-practi
ces-with-amazon-s3/) you can read some best practices with S3."
   ]
  },
```

```
  {
   "cell_type": "markdown",
   "id": "c77f67b2",
   "metadata": {},
   "source": [
    "(awswrangler)=\n",
    "## How to read data from S3 using awswrangler (optional)\n",
    "\n",
    "First install it;\n",
    "\n",
    "```bash\n",
    "!conda install -c conda-forge -y awswrangler\n",
    "```\n",
    "\n",
    "If you are planning to install in EC2 instance then; \n",
    "\n",
    "```\n",
    "sudo -E conda install -y awswrangler\n",
    "```"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "e79cbada",
   "metadata": {},
   "outputs": [],
   "source": [
    "import awswrangler as wr"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "65e6308e",
   "metadata": {
    "tags": [
     "skip-execution"
    ]
   },
   "outputs": [],
   "source": [
    "## just getting 100 rows to save time\n",
    "df = wr.s3.read_csv(\"s3://testmds/combined_data.csv\")\n",
    "print(df[(df.year== \"2004\") & (df.ArrDelay > 10)][\"UniqueCarrier\"].value_counts())"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "384e19d2",
   "metadata": {
    "tags": [
     "skip-execution"
    ]
   },
   "outputs": [],
   "source": [
    "df = wr.s3.read_parquet(path=\"s3://testmds/combined_data.parquet\",
dataset=True)\n",
    "print(df[(df.year== \"2004\") & (df.ArrDelay >
10)][\"UniqueCarrier\"].value_counts())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "c7711b27",
   "metadata": {
    "tags": [
     "skip-execution"
    ]
   },
   "outputs": [],
   "source": [
    "df =
wr.s3.read_parquet(path=\"s3://testmds/combined_data.parquet\",\n",
    "      dataset=True,\n",
    "      columns=['year', 'UniqueCarrier', 'ArrDelay'],\n",
    "      # We have to give it this way to filter on partitions\n",
    "      partition_filter=lambda x: x[\"year\"] == \"2004\",\n",
    "      # I don't see this filters getting pushdown internally maybe bug
https://github.com/aws/aws-sdk-pandas/issues/1032\n",
    "      pyarrow_additional_kwargs={\"filters\":[('ArrDelay', '>',
10)]}\n",
    "    ) \n",
    "print(df[(df.year== \"2004\") & (df.ArrDelay >
10)][\"UniqueCarrier\"].value_counts())"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "c0a7eef8",
   "metadata": {},
   "source": [
    "If you want upload a file using awswrangler;\n",
    "\n",
    "```python\n",
    "# install first\n",
    "# sudo -E conda install -y s3fs\n",
    "# sudo -E conda install -y awswrangler\n",
    "import awswrangler as wr\n",
    "wr.s3.upload(local_file='img/task.jpeg',
path='s3://testmdsprivate/task.jpeg')\n",
    "```\n",
    "\n",
    "```{note}\n",
    "The above usage is to upload a file. If you want to upload a folder
(or a parquet file) best is to use AWS CLI.\n",
    "```"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "787ce823",
   "metadata": {},
   "source": [
    "(explicitly)=\n",
    "## Passing credentials explicitly (optional)\n",
    "\n",
    "### In pandas\n",
    "\n",
    "```python\n",
    "import pandas as pd\n",
    "aws_credentials ={\"key\": \"paste_key\",\"secret\":
\"paste_secret\",\"token\":\"paste_token\"} \n",
    "## dont include you secret and key when submitting the notebook\n",
    "df = pd.read_parquet(\"s3://testmds/combined_data.parquet\", \n",
    "        filters=[('year','=', 2004)], \n",
    "                columns=['year', 'UniqueCarrier', 'ArrDelay'],
storage_options=aws_credentials)\n",
    "```\n",
    "```python\n",
    "df =
pd.read_csv(\"s3://testmds/combined_data.csv\",storage_options=aws_credenti
als)\n",
    "print(df[(df.year== 2004) & (df.ArrDelay >
```

```
10)][\"UniqueCarrier\"].value_counts())\n",
    "```\n",
    "\n",
    "### In awswrangler\n",
    "\n",
    "```python\n",
    "## install boto if you plan to do pass it explicitly\n",
    "import boto3\n",
    "import awswrangler as wr\n",
    "session = boto3.Session(\n",
    "    aws_access_key_id=\"xxx\",\n",
    "    aws_secret_access_key=\"xxx\",\n",
    "  aws_session_token=\"xxxx\"\n",
    ")\n",
    "df = wr.s3.read_parquet(path=\"s3://testmds/combined_data.parquet\",
dataset=True, boto3_session= session)\n",
    "```\n",
    "\n",
    "```\n",
    "df = wr.s3.read_csv(\"s3://testmds/combined_data.csv\", boto3_session=
session)\n",
    "```\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "609edb30",
   "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"
  },
  "vscode": {
   "interpreter": {
    "hash":
"6e9e0baa62560f8a3b402c12d339bdad33c58a25305700ec7e7682c0b6251f68"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```