W04 –
SETTING UP SPARK ENVIRONMENT
LAB01 PRACTICE

2024

The Hoang

LEARNING OBJECTIVES – W04 & W05

- Lab 00: Setting Up PySpark with Jupyter Notebook
- Lab 01: Introduction to SparkContext and RDDs
- Lab 02: Introduction to SparkSession and DataFrames APIs
- Lab 03: SparkSQL and DataFrames Integration
- Lab 04: Advanced RDDs and Pair RDDs
- Lab 05: DataFrames vs RDDs: Performance and Use Cases
- Notes and Recommendations

LAB 00
SETTING UP PYSPARK WITH JUPYTER NOTEBOOK

2024

The Hoang

Setup Spark environment

- Objective
  - Setup Jupyter Notebooks with Apache Spark
- Prerequisites
  - If you are using Windows OS, setup WSL2 for Windows PC. Follow this instruction or this official link from Microsoft.
- Setup Docker Desktop
  - Link Download and Setup Docker Desktop for Windows PCs
  - Link Download and Setup Docker Desktop for MacOS / Linux
- [*Optional*] Install Docker Toolbox for PCs which don't support Docker Desktop

[Windows] Setup & verify WSL

- Check if the WSL is installed
  - `wsl --list –verbose`
- List all the online distributions
  - `wsl --list --online`
- Set default version to 2
  - `wsl --set-default-version 2`
- Install Ubuntu
  - `wsl --update`
- Check if the WSL is installed
  - `wsl --install`

```
tedd@THEHN: ~
C:\Users\HOANG>
C:\Users\HOANG>wsl --update
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.

C:\Users\HOANG>wsl --install
Installing: Ubuntu
Ubuntu has been installed.
Launching Ubuntu...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: tedd
New password:
Retype new password:
passwd: password updated successfully
The operation completed successfully.
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This message is shown once a day. To disable it please create the
/home/tedd/.hushlogin file.
tedd@THEHN:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
tedd@THEHN:~$
```

[Windows] Setup & verify WSL

The WSL --install command performs the following actions:

- Enables the optional WSL and Virtual Machine Platform components
- Downloads and installs the latest Linux kernel
- Sets WSL 2 as the default
- Downloads and installs the Ubuntu Linux distribution (reboot may be required)

[Windows] Setup & verify WSL

- Update and upgrade packages
  - `sudo apt update && sudo apt upgrade`

```
tedd@THEHN: ~
Setting up libpython3.10:amd64 (3.10.12-1~22.04.6) ...
Setting up systemd-sysv (249.11-0ubuntu3.12) ...
Setting up vim (2:8.2.3995-1ubuntu2.18) ...
Setting up python3.10 (3.10.12-1~22.04.6) ...
Setting up libnss-systemd:amd64 (249.11-0ubuntu3.12) ...
Setting up python3 (3.10.6-1~22.04.1) ...
running python rtupdate hooks for python3.10...
running python post-rtupdate hooks for python3.10...
Setting up binutils (2.38-4ubuntu2.6) ...
Setting up python3-zipp (1.0.0-3ubuntu0.1) ...
Setting up netplan.io (0.106.1-7ubuntu0.22.04.4) ...
Setting up python3-cryptography (3.4.8-1ubuntu2.2) ...
Setting up libpam-systemd:amd64 (249.11-0ubuntu3.12) ...
Setting up bind9-dnsutils (1:9.18.28-0ubuntu0.22.04.1) ...
Setting up python3-distro-info (1.1ubuntu0.2) ...
Setting up python3-pkg-resources (59.6.0-1.2ubuntu0.22.04.2) ...
Setting up python3-problem-report (2.20.11-0ubuntu82.6) ...
Setting up python3-apt (2.4.0ubuntu4) ...
Setting up ubuntu-standard (1.481.3) ...
Setting up python3-distupgrade (1:22.04.20) ...
Setting up python3-apport (2.20.11-0ubuntu82.6) ...
Setting up python3-software-properties (0.99.22.9) ...
Setting up ubuntu-release-upgrader-core (1:22.04.20) ...
Setting up ubuntu-pro-client (33.2~22.04) ...
Setting up ubuntu-pro-client-l10n (33.2~22.04) ...
Setting up software-properties-common (0.99.22.9) ...
Setting up apport (2.20.11-0ubuntu82.6) ...
apport-autoreport.service is a disabled or a static unit, not starting it.
Setting up ubuntu-advantage-tools (33.2~22.04) ...
Setting up ubuntu-minimal (1.481.3) ...
Setting up ubuntu-wsl (1.481.3) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for plymouth-theme-ubuntu-text (0.9.5+git20211018-1ubuntu3) ...
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
tedd@THEHN:~$
```

Setup Docker

Docker

- An open-source platform that enables the development, deployment, and management of applications using containers

Docker core concepts - Quick introduction

Image

- A Docker image is a read-only template that contains the instructions for creating a Docker container.

Container

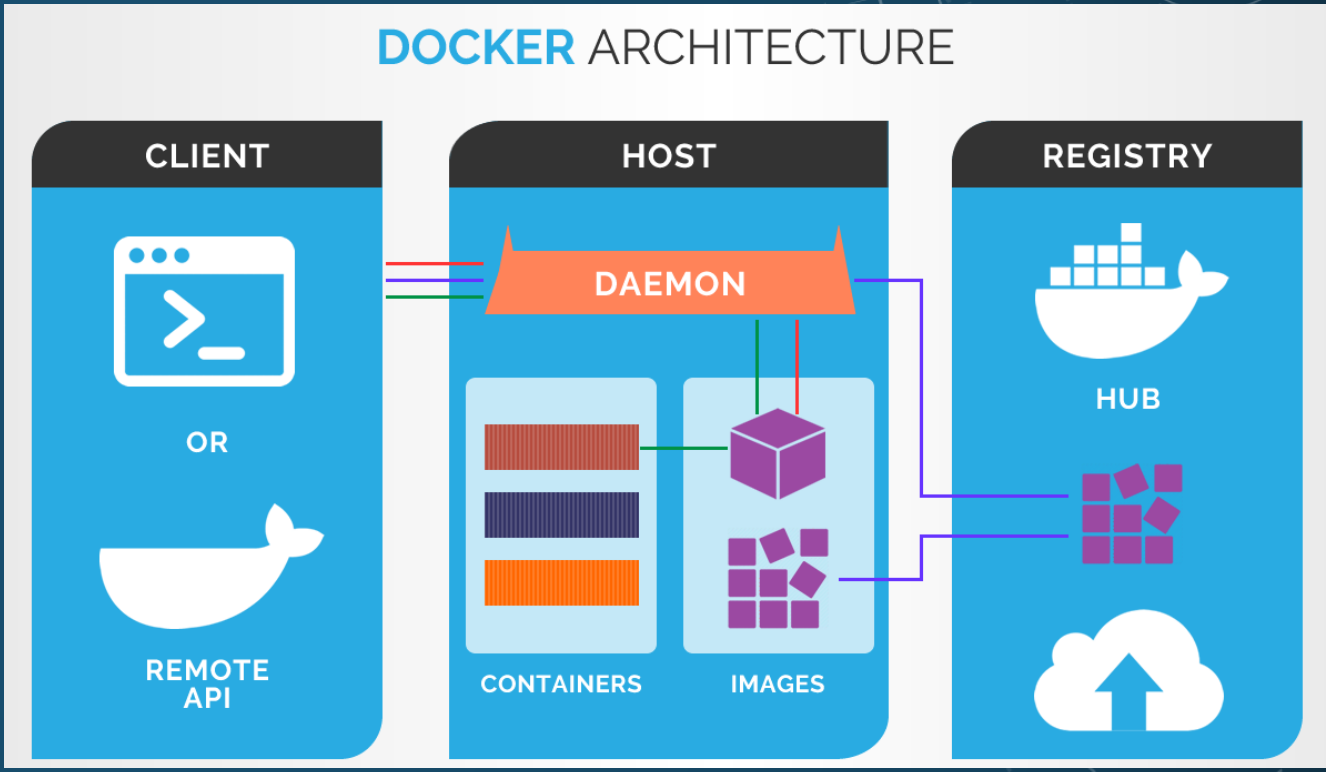
- A lightweight, standalone, and executable software package that includes everything needed to run an application - the code, runtime, system tools, libraries, and settings.
- Isolate applications from their environment, ensuring that the application runs consistently across different computing environments (e.g., development, testing, and production).

Containerization

- Containerization is the process of packaging an application and its dependencies into a standardized unit called a container.

Docker compose

- Define and run multi-container applications.
- Allows defining the services, networks, and volumes that make up an application in a single YAML file, making it easier to manage and deploy the application.



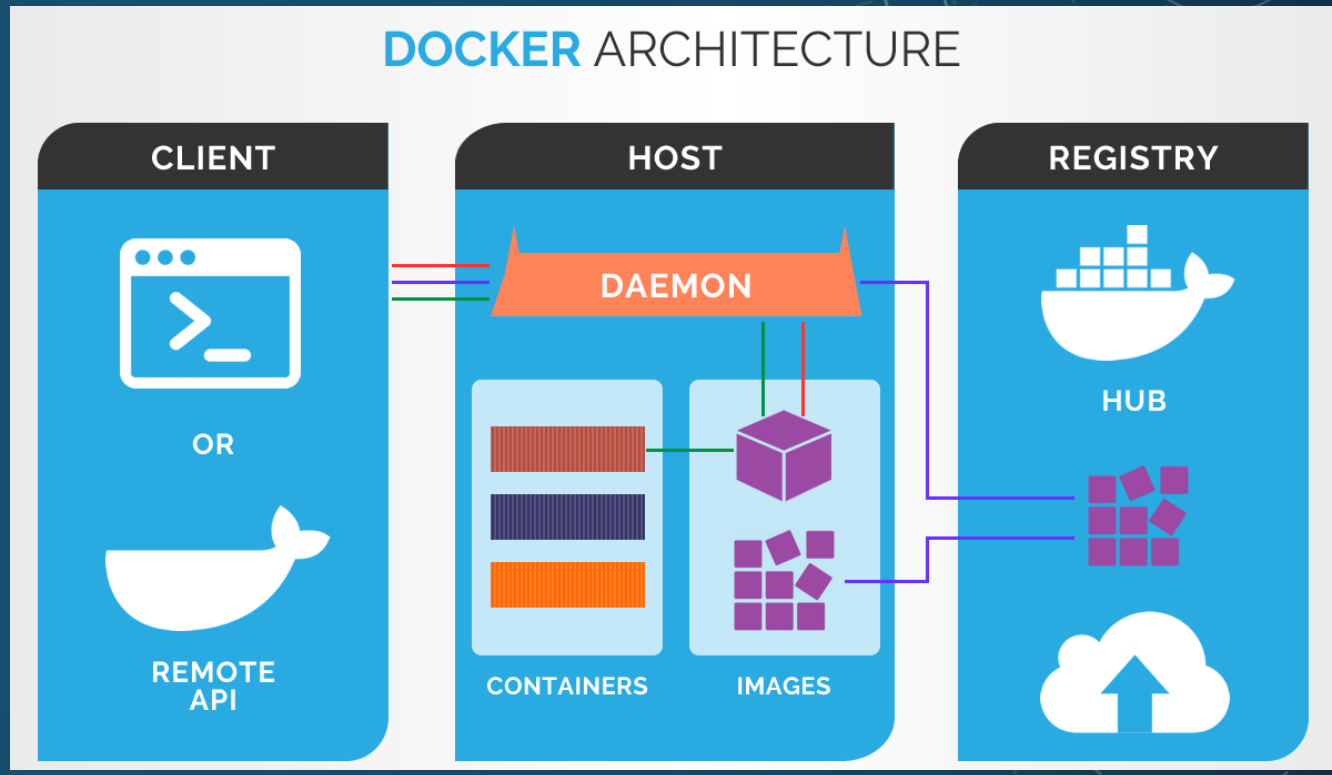
Docker core concepts - Quick introduction

Docker Networking

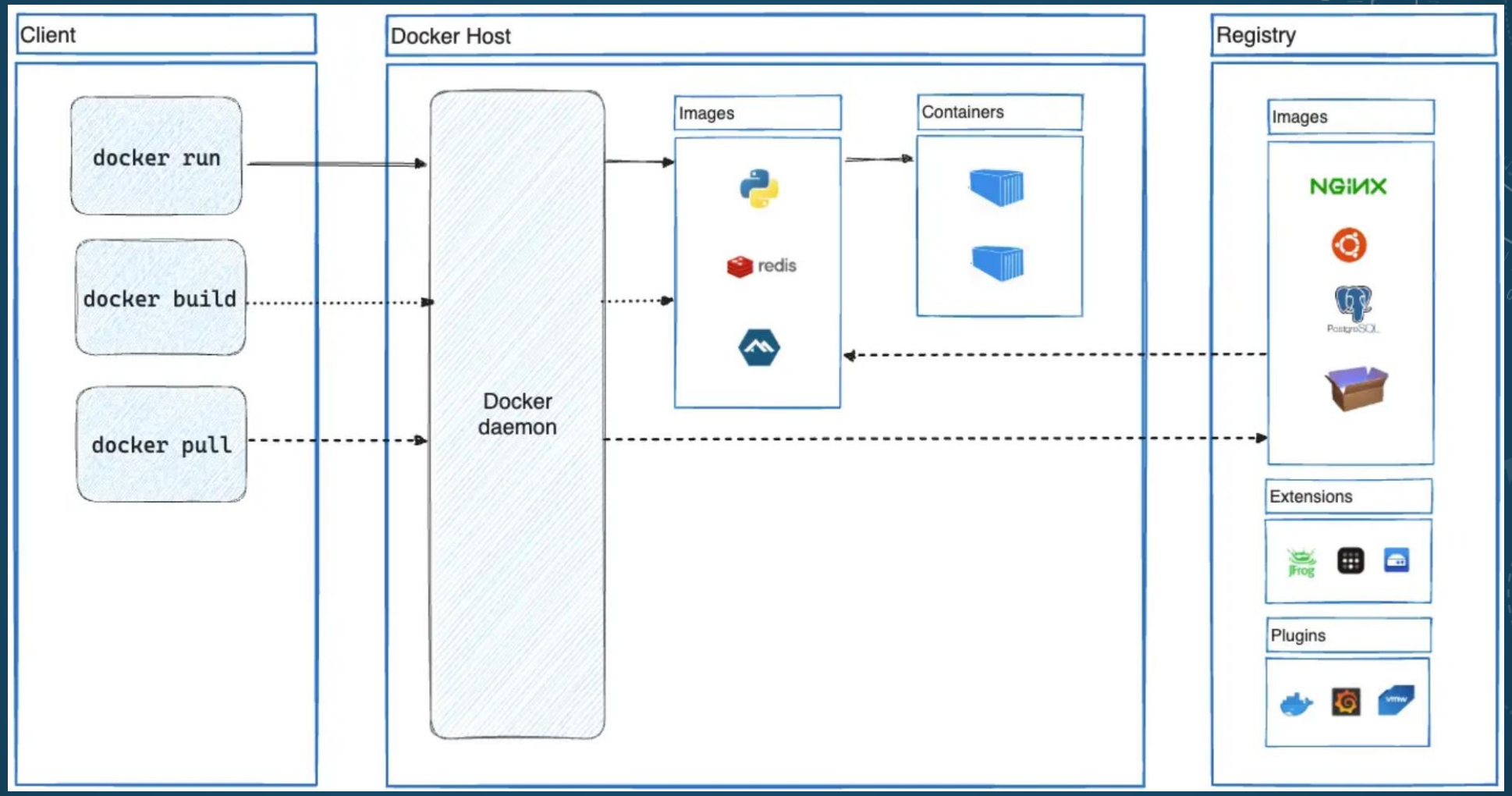
- Docker provides built-in networking capabilities that allow containers to communicate with each other and with the host system.
- Containers can be connected to one or more networks, and each network has its own IP address range and DNS resolution.

Docker Volumes

- Docker volumes are used to persist data generated by a container.
- Volumes are independent of the container's lifecycle and can be shared between containers.
- Volumes provide a way to store and manage data outside of the container's file system, making it easier to backup, restore, and share data between different environments.
- Volumes can be used to store application data, configuration files, and other persistent data that needs to be accessed by the container

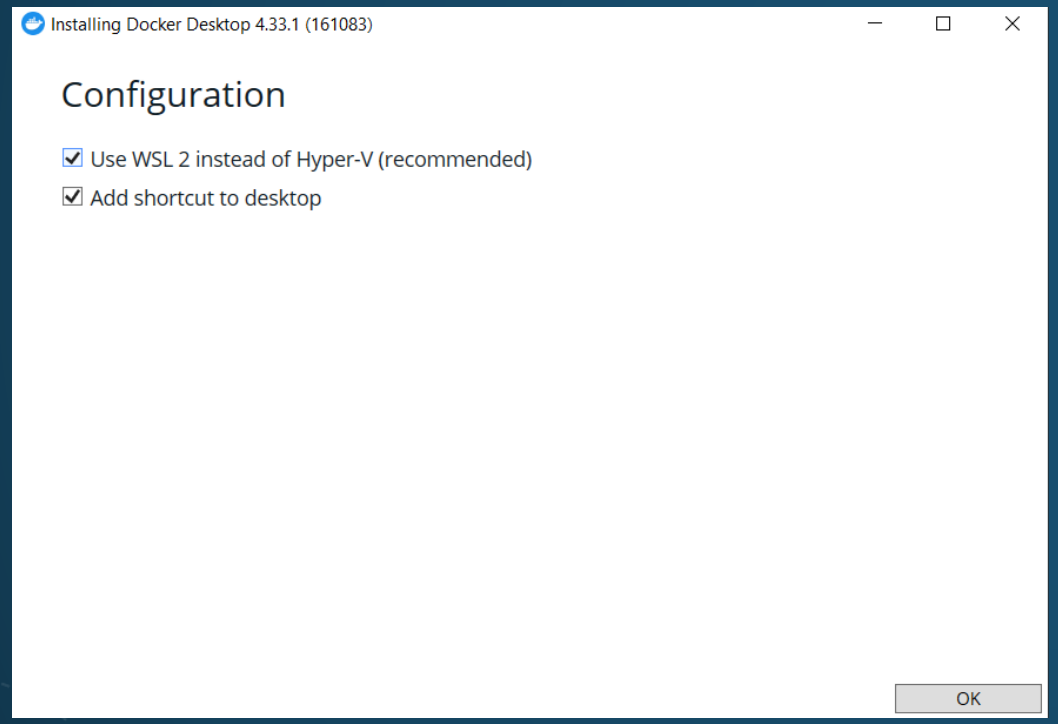


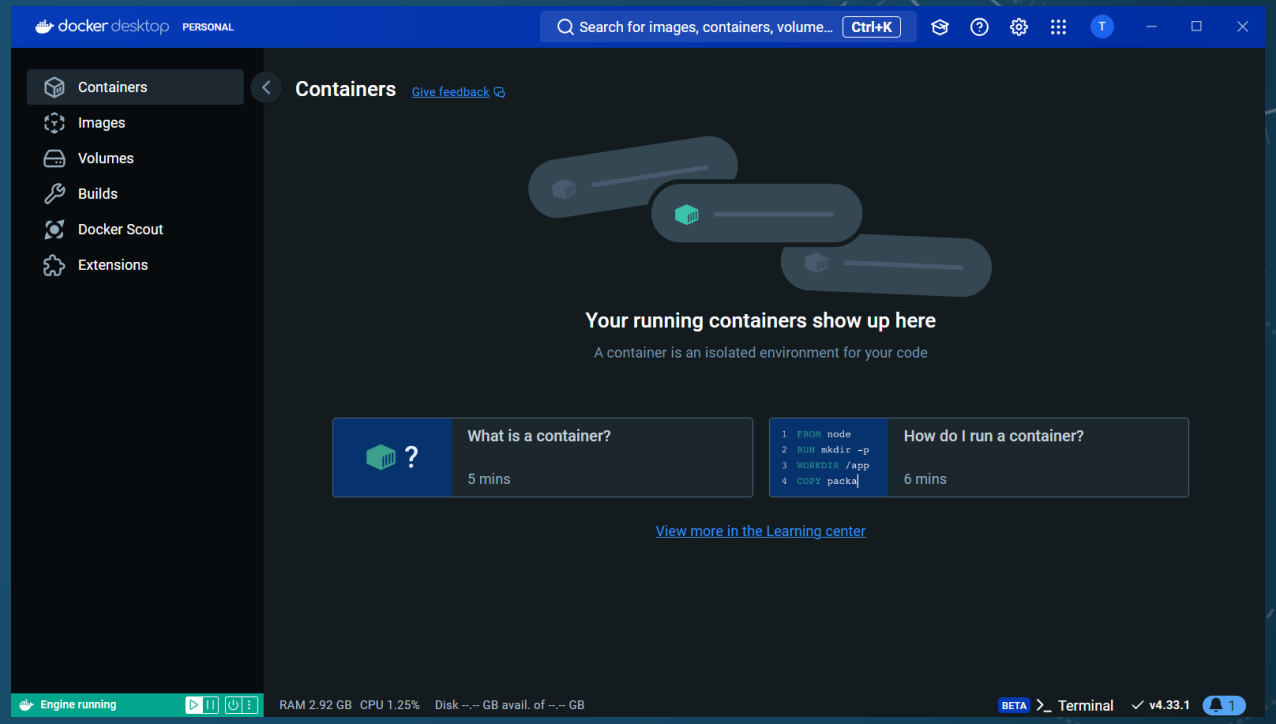
Docker – How it works?



Setup Docker Desktop

- Download and Install Docker Desktop





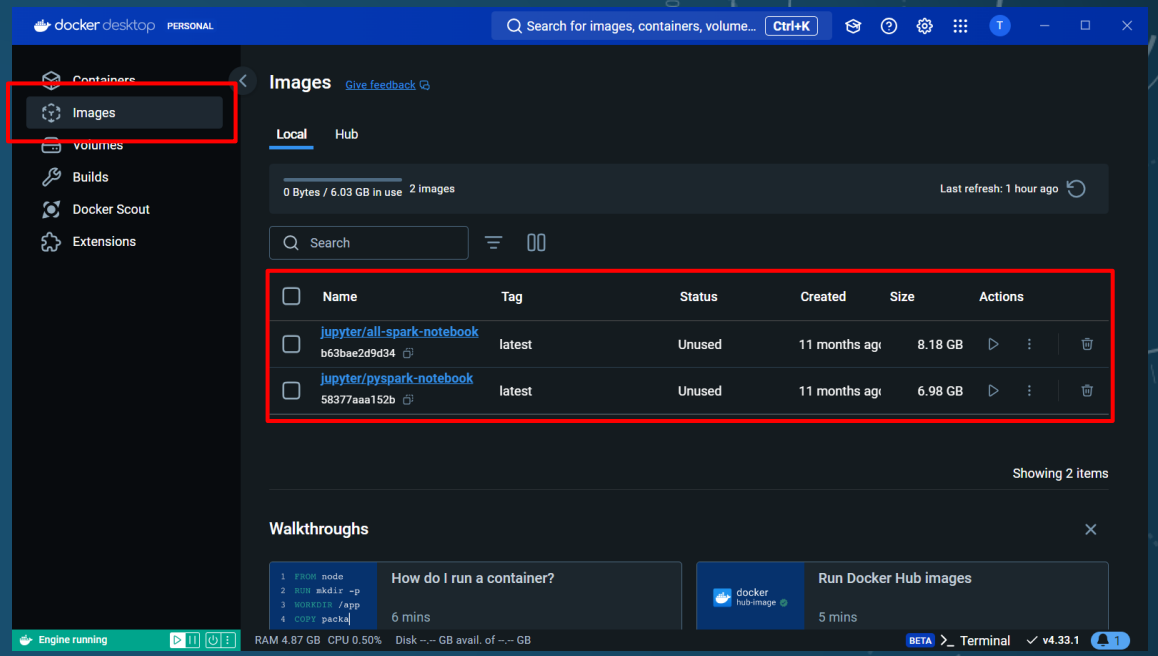
Launch Docker, Pull images

- Complete Docker setup
- Launch Docker Desktop to start Docker services
- (For Windows) Open WSL terminal to activate Linux shell
- Execute below command to pull docker image:

```
docker pull jupyter/pyspark-notebook
```

- Verify the image pulled successfully
  - Using Docker Desktop UI
  - Using docker command from terminal

    ```
    docker image ls
    ```




```
tedd@THEHN:~/courses/neu/data-analytics-with-spark/docker$ docker image ls
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
jupyter/all-spark-notebook    latest    b63bae2d9d34   11 months ago  8.19GB
jupyter/pyspark-notebook      latest    58377aaa152b   11 months ago  6.98GB
tedd@THEHN:~/courses/neu/data-analytics-with-spark/docker$
```

References:

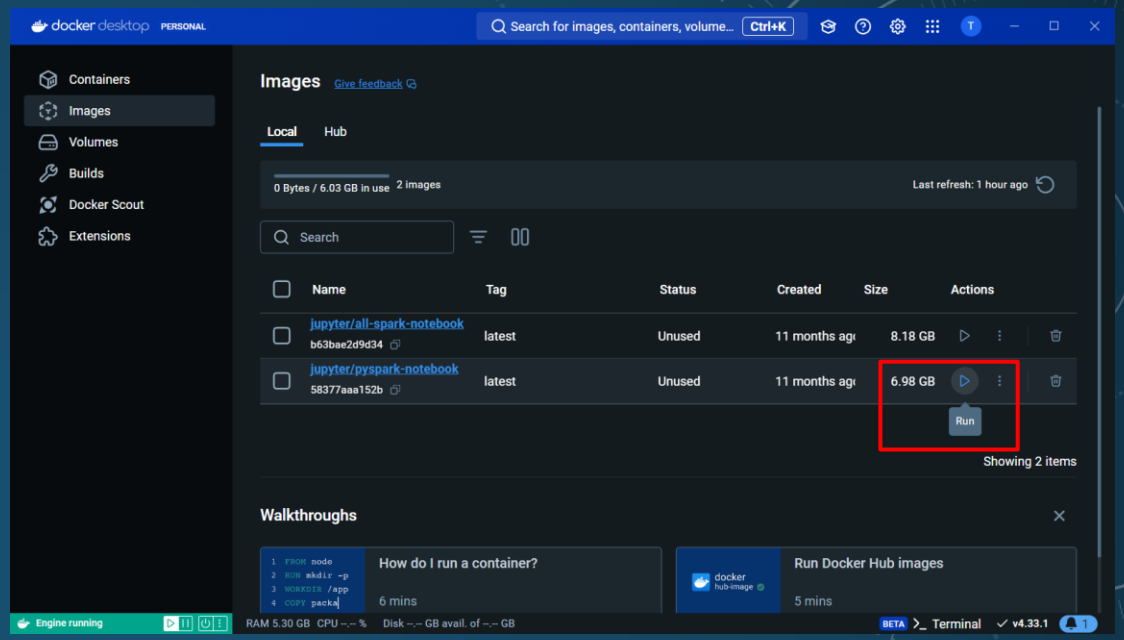
JUPYTER/DOCKER-STACKS: READY-TO-RUN DOCKER IMAGES CONTAINING JUPYTER APPLICATIONS (GITHUB.COM)

SELECTING AN IMAGE — DOCKER STACKS DOCUMENTATION (JUPYTER-DOCKER-STACKS.READTHEDOCS.IO)

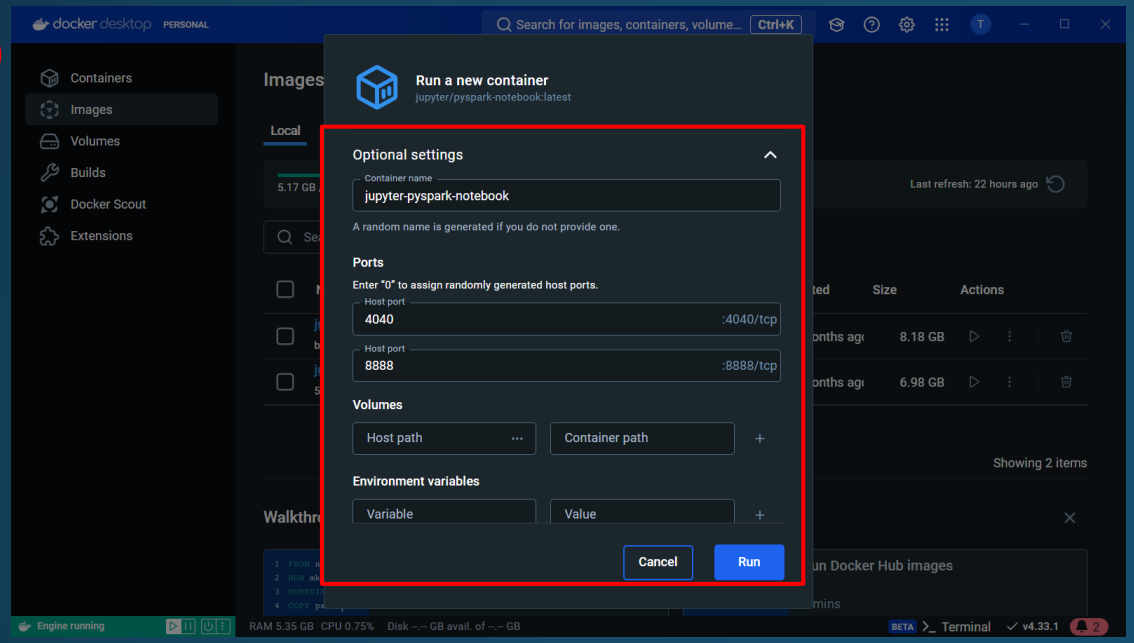
Run Docker Container

- Execute from Docker desktop UI

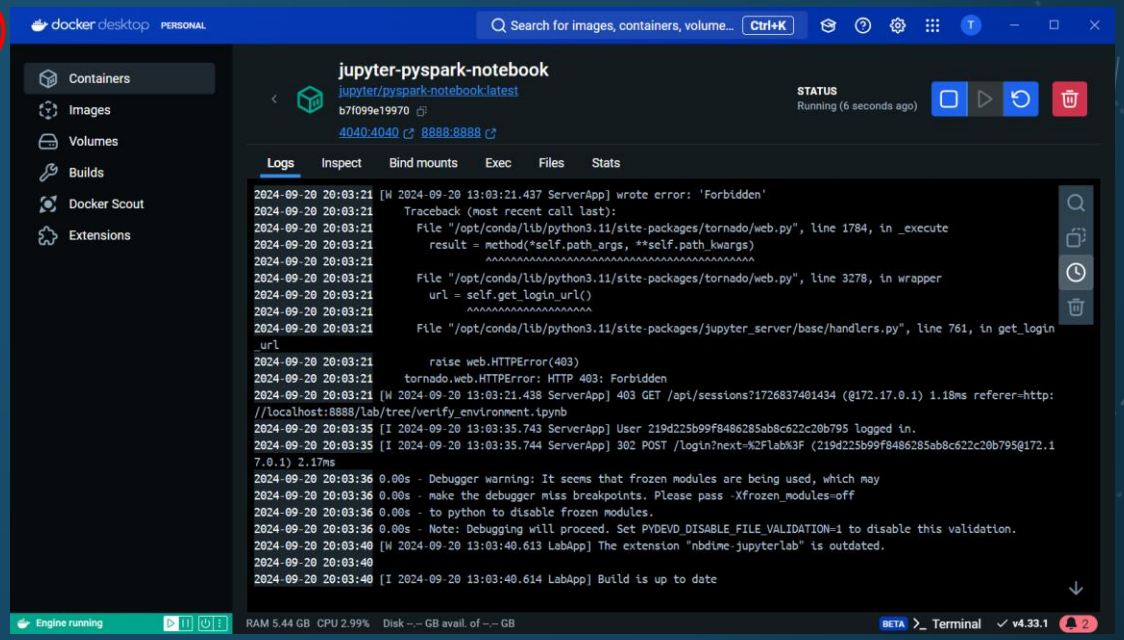
1



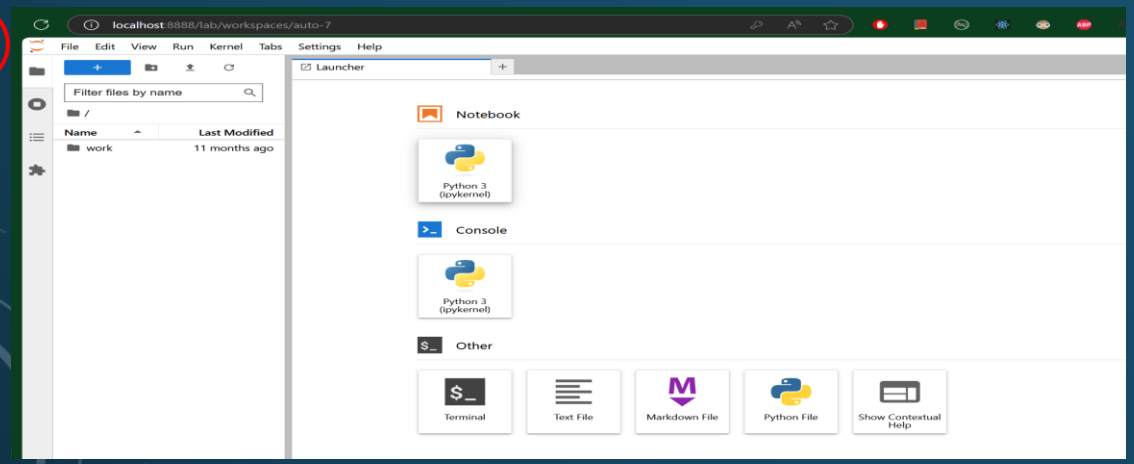
2



3



4



Run Docker Container

- Execute from Terminal using Docker commands
  - docker run --rm -p 4040:4040 -p 8888:8888 jupyter/pyspark-notebook
  - Copy token from Terminal and use it for login

```
p] notebook | extension was successfully loaded.
p] Serving notebooks from local directory: /home/jovyan
p] Jupyter Server 2.8.0 is running at:
p] http://ef891f369896:8888/lab?token=fc10cb48425d83283c8d6e6f01a65336954065ae5702d91b
p]     http://127.0.0.1:8888/lab?token=fc10cb48425d83283c8d6e6f01a65336954065ae5702d91b
p] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
p]

 file in a browser:
```

```
tedd@THEHN: ~/courses/neu/data-analytics-with-spark/docker
-rw-r--r-- 1 tedd tedd   373 Sep 19 23:33 docker-compose.yml
drwxr-xr-x 8 tedd users 4096 Sep 19 23:33 notebook_data/
tedd@THEHN:~/courses/neu/data-analytics-with-spark/docker$ docker run --rm -p 4040:4040 -p 8888:8888 jupyt
Entered start.sh with args: jupyter lab
Running hooks in: /usr/local/bin/start-notebook.d as uid: 1000 gid: 100
Done running hooks in: /usr/local/bin/start-notebook.d
Running hooks in: /usr/local/bin/before-notebook.d as uid: 1000 gid: 100
Sourcing shell script: /usr/local/bin/before-notebook.d/spark-config.sh
Done running hooks in: /usr/local/bin/before-notebook.d
Executing the command: jupyter lab
[I 2024-09-20 13:11:06.746 ServerApp] Package jupyterlab took 0.0000s to import
[I 2024-09-20 13:11:06.761 ServerApp] Package jupyter_lsp took 0.0142s to import
[W 2024-09-20 13:11:06.762 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyt
and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2024-09-20 13:11:06.764 ServerApp] Package jupyter_server_mathjax took 0.0016s to import
[I 2024-09-20 13:11:06.771 ServerApp] Package jupyter_server_terminals took 0.0070s to import
[I 2024-09-20 13:11:06.801 ServerApp] Package jupyterlab_git took 0.0288s to import
[I 2024-09-20 13:11:06.804 ServerApp] Package nbclassic took 0.0025s to import
[W 2024-09-20 13:11:06.806 ServerApp] A `_jupyter_server_extension_points` function was not found in nbcla
d will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2024-09-20 13:11:06.807 ServerApp] Package nbdime took 0.0000s to import
[I 2024-09-20 13:11:06.807 ServerApp] Package notebook took 0.0000s to import
[I 2024-09-20 13:11:06.811 ServerApp] Package notebook_shim took 0.0000s to import
[W 2024-09-20 13:11:06.811 ServerApp] A `_jupyter_server_extension_points` function was not found in noteb
d and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2024-09-20 13:11:06.811 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-09-20 13:11:06.817 ServerApp] jupyter_server_mathjax | extension was successfully linked.
[I 2024-09-20 13:11:06.822 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-09-20 13:11:06.830 ServerApp] jupyterlab | extension was successfully linked.
[I 2024-09-20 13:11:06.830 ServerApp] jupyterlab_git | extension was successfully linked.
```

Run Docker Container

- Create docker-compose application
  - File name: docker-compose.yml
  - Services
  - Image
  - Ports
  - User
  - Environment
  - Working Directory
  - Command
  - Restart
  - External Volumes

.gitignore | docker-compose.yml

docker > docker-compose.yml

```yaml
version: '3'
services:
  jupyter:
    image: jupyter/pyspark-notebook
    ports:
      - 8888:8888
      - 4040:4040
    user: root
    environment:
      - NB_USER=thehn
      - CHOWN_HOME=yes
    working_dir: /home/thehn
    command: ["start-notebook.sh"]
    restart: always
    volumes:
      - ./notebook_data:/home/thehn

volumes:
  notebook_data:
    external: true
```

Run Docker Container

- Execute docker-compose application
  - Navigate to directory which contains the docker-compose.yml
  - Execute: `docker-compose up`
  - Optional parameter: `-d`
  - Change password for the next logins

Select tedd@THEHN: ~/courses/neu/data-analytics-with-spark/docker

```
drwxr-xr-x 4 tedd tedd  4096 Sep 19 23:35 ../
-rw-r--r-- 1 tedd tedd   373 Sep 19 23:33 docker-compose.yml
drwxr-xr-x 8 tedd users 4096 Sep 19 23:33 notebook_data/
tedd@THEHN:~/courses/neu/data-analytics-with-spark/docker$ docker-compose up
WARN[0000] /home/tedd/courses/neu/data-analytics-with-spark/docker/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potentia
l confusion
[+] Running 2/2
 ✔ Network docker_default      Created                                                                                                                                    0.2s
 ✔ Container docker-jupyter-1  Created                                                                                                                                    0.7s
Attaching to jupyter-1
jupyter-1  | WARNING: Use start-notebook.py instead
jupyter-1  | Entered start.sh with args: jupyter lab
jupyter-1  | Running hooks in: /usr/local/bin/start-notebook.d as uid: 0 gid: 0
jupyter-1  | Done running hooks in: /usr/local/bin/start-notebook.d
jupyter-1  | Updated the jovyan user:
jupyter-1  | - username: jovyan       -> thehn
jupyter-1  | - home dir: /home/jovyan -> /home/thehn
jupyter-1  | Ensuring /home/thehn is owned by 1000:100
jupyter-1  | Running hooks in: /usr/local/bin/before-notebook.d as uid: 0 gid: 0
jupyter-1  | Sourcing shell script: /usr/local/bin/before-notebook.d/spark-config.sh
jupyter-1  | Done running hooks in: /usr/local/bin/before-notebook.d
jupyter-1  | Running as thehn: jupyter lab
jupyter-1  | [I 2024-09-20 13:24:29.947 ServerApp] Package jupyterlab took 0.0000s to import
jupyter-1  | [I 2024-09-20 13:24:29.987 ServerApp] Package jupyter_lsp took 0.0398s to import
jupyter-1  | [W 2024-09-20 13:24:29.988 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` functi
on was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
jupyter-1  | [I 2024-09-20 13:24:30.001 ServerApp] Package jupyter_server_mathjax took 0.0128s to import
jupyter-1  | [I 2024-09-20 13:24:30.022 ServerApp] Package jupyter_server_terminals took 0.0189s to import
jupyter-1  | [I 2024-09-20 13:24:30.122 ServerApp] Package jupyterlab_git took 0.0997s to import
jupyter-1  | [I 2024-09-20 13:24:30.135 ServerApp] Package nbclassic took 0.0125s to import
```

Jupyter Pyspark Notebook

- Verify environment and setup

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ w04 /

Name	Last Modified
verify_envir...	20 hours ago

verify_environment.ipynb

Code

Open in...

Python 3 (ipykernel)

[1]:
```
print("hello world!!")
```

```
hello world!!
```

[2]:
```
from pyspark.sql import SparkSession
```

[3]:
```
spark = SparkSession.builder.appName("Data Analytics with Apache Spark").master("local").getOrCreate()
```

[4]:
```
spark
```

[4]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	`v3.5.0`
Master	`local`
AppName	`Data Analytics with Apache Spark`

[5]:
```
sc = spark.sparkContext
```

[8]:
```
sc
```

[8]: **SparkContext**

Spark UI

Version	`v3.5.0`
Master	`local`
AppName	`Data Analytics with Apache Spark`

LAB 01: INTRODUCTION TO SPARKCONTEXT AND RDDS

2024

The Hoang

SparkSession and SparkContext

- Initialize SparkSession
- Get SparkContext from SparkSession

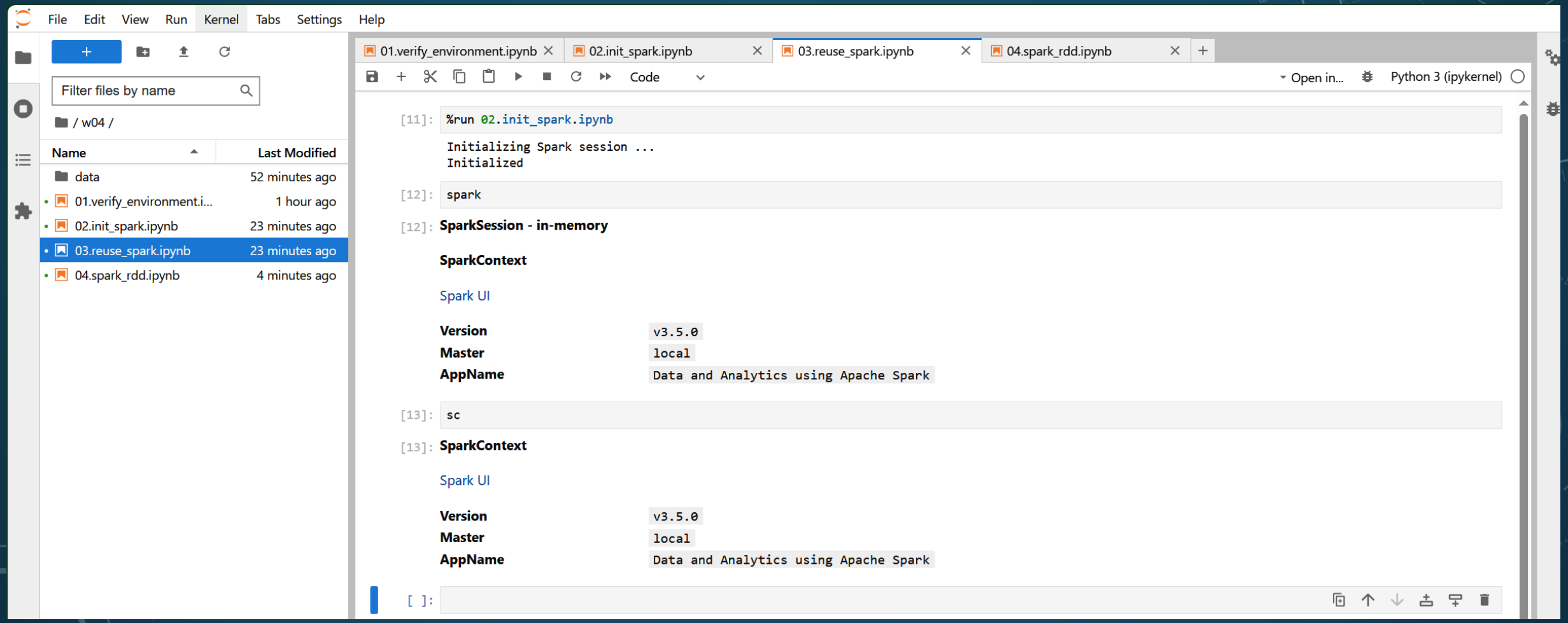
Code

[6]:
```
print('Initializing Spark session ...')
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Data and Analytics using Apache Spark").master("local").getOrCreate()
sc = spark.sparkContext
print('Initialized')
```

```
Initializing Spark session ...
Initialized
```

SparkSession and SparkContext

- Make the initialization reusable



Spark RDD APIs – Creation

- Creating RDD from a text file

Code Open in... Python 3 (ipykernel)

[2]:
```
%run 02.init_spark.ipynb
```

```
Initializing Spark session ...
Initialized
```

[10]:
```
# reading data from a text file (a book)
rdd = sc.textFile('data/Complete Works of William Shakespeare.txt')
```

[11]:
```
# number of lines in the book
rdd.count()
```

[11]:
```
196390
```

[16]:
```
# taking top ten lines of the rdd for previewing
rdd.take(10)
```

[16]:
```
['The Project Gutenberg eBook of The Complete Works of William Shakespeare',
 '    ',
 'This ebook is for the use of anyone anywhere in the United States and',
 'most other parts of the world at no cost and with almost no restrictions',
 'whatsoever. You may copy it, give it away or re-use it under the terms',
 'of the Project Gutenberg License included with this ebook or online',
 'at www.gutenberg.org. If you are not located in the United States,',
 'you will have to check the laws of the country where you are located',
 'before using this eBook.',
 '']
```

Spark RDD APIs – Creation

- Creating a new RDD from an existing RDD

```
[17]: # create a new rdd from the origin rdd - which created by reading the text file
      words_rdd = rdd.flatMap(lambda line: line.split())

[18]: words_rdd

[18]: PythonRDD[15] at RDD at PythonRDD.scala:53

[26]: # Get 10 words for previewing
      words_rdd.take(10)

[26]: ['The',
       'Project',
       'Gutenberg',
       'eBook',
       'of',
       'The',
       'Complete',
       'Works',
       'of',
       'William']
```

Spark RDD APIs – Creation

- Creating from a sequence (list)

```
[50]: arr = [i for i in range(100)]

[52]: arr_rdd = sc.parallelize(arr)

[53]: arr_rdd.take(10)

[53]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[54]: arr_rdd.count()

[54]: 100
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Narrow transformation: **flatMap**, map, filter

```
[17]: # create a new rdd from the origin rdd - which created by reading the text file
      words_rdd = rdd.flatMap(lambda line: line.split())

[18]: words_rdd

[18]: PythonRDD[15] at RDD at PythonRDD.scala:53

[26]: # Get 10 words for previewing
      words_rdd.take(10)

[26]: ['The',
       'Project',
       'Gutenberg',
       'eBook',
       'of',
       'The',
       'Complete',
       'Works',
       'of',
       'William']
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Narrow transformation: flatMap, **map**, filter

```
[27]: word_map_rdd = words_rdd.map(lambda word: (word, 1))

[28]: word_map_rdd.take(10)

[28]: [('The', 1),
       ('Project', 1),
       ('Gutenberg', 1),
       ('eBook', 1),
       ('of', 1),
       ('The', 1),
       ('Complete', 1),
       ('Works', 1),
       ('of', 1),
       ('William', 1)]
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Narrow transformation: flatMap, map, **filter**

```
[44]: # filter words by conditions: get all words which have length longer than 5 chars
      long_words_rdd = words_rdd.filter(lambda x: len(x) > 5)

[45]: long_words_rdd.take(10)

[45]: ['Project',
       'Gutenberg',
       'Complete',
       'William',
       'Shakespeare',
       'anyone',
       'anywhere',
       'United',
       'States',
       'almost']
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Wide transformation: **reduceByKey**, sortByKey

```
[28]: word_map_rdd.take(10)

[28]: [('The', 1),
       ('Project', 1),
       ('Gutenberg', 1),
       ('eBook', 1),
       ('of', 1),
       ('The', 1),
       ('Complete', 1),
       ('Works', 1),
       ('of', 1),
       ('William', 1)]

[29]: word_count_rdd = word_map_rdd.reduceByKey(lambda a, b: a + b)

[30]: word_count_rdd.take(10)

[30]: [('The', 4630),
       ('Project', 79),
       ('Gutenberg', 22),
       ('eBook', 4),
       ('of', 17111),
       ('Complete', 3),
       ('Works', 4),
       ('William', 35),
       ('Shakespeare', 5),
       ('This', 1218)]
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Wide transformation: reduceByKey, **sortByKey**

```
[38]: swap_word_count_rdd = word_count_rdd.map(lambda x: (x[1], x[0]))

[39]: swap_word_count_rdd.take(10)

[39]: [(4630, 'The'),
       (79, 'Project'),
       (22, 'Gutenberg'),
       (4, 'eBook'),
       (17111, 'of'),
       (3, 'Complete'),
       (4, 'Works'),
       (35, 'William'),
       (5, 'Shakespeare'),
       (1218, 'This')]

[41]: swap_word_count_rdd.sortByKey(ascending=False).take(10)

[41]: [(25689, 'the'),
       (20717, 'I'),
       (19849, 'and'),
       (17111, 'of'),
       (17075, 'to'),
       (13730, 'a'),
       (11397, 'my'),
       (10943, 'in'),
       (9527, 'you'),
       (8361, 'is')]
```

Spark RDD APIs – Transformations

- Manipulate RDD transformation APIs to work with text data:
  - Wide transformation: reduceByKey, **sortByKey**

```
[38]: swap_word_count_rdd = word_count_rdd.map(lambda x: (x[1], x[0]))

[39]: swap_word_count_rdd.take(10)

[39]: [(4630, 'The'),
       (79, 'Project'),
       (22, 'Gutenberg'),
       (4, 'eBook'),
       (17111, 'of'),
       (3, 'Complete'),
       (4, 'Works'),
       (35, 'William'),
       (5, 'Shakespeare'),
       (1218, 'This')]

[41]: swap_word_count_rdd.sortByKey(ascending=False).take(10)

[41]: [(25689, 'the'),
       (20717, 'I'),
       (19849, 'and'),
       (17111, 'of'),
       (17075, 'to'),
       (13730, 'a'),
       (11397, 'my'),
       (10943, 'in'),
       (9527, 'you'),
       (8361, 'is')]
```

Spark RDD APIs – Actions

- Manipulate RDD transformation APIs to work with text data:
  - rdd.collect(): Return all elements as an array in the driver program.
  - rdd.count(), rdd.countByValue(): Return the number of elements or count of each unique value.
  - rdd.take(n), rdd.takeSample(withReplacement, num, seed): Return the first n elements or a random sample.
  - rdd.reduce(func), rdd.fold(zeroValue)(func): Aggregate elements using a function.
  - rdd.saveAsTextFile(path), rdd.saveAsSequenceFile(path): Save the RDD to a file.

Spark RDD APIs – Persistence

- Manipulate RDD transformation APIs to work with text data:
  - rdd.cache(), rdd.persist(storageLevel): Cache the RDD in memory or on disk for reuse.
  - rdd.checkpoint(): Mark the RDD for checkpointing to fault-tolerant storage.

Spark RDD APIs – Persistence

Feature	Checkpointing	Caching
Purpose	To save the state of RDDs/DataFrames to reliable storage for fault tolerance and lineage truncation.	To store intermediate data in memory for faster access during subsequent operations.
Storage	Writes data to a reliable storage system (e.g., HDFS, S3).	Keeps data in memory (RAM) for quick retrieval.
Data Materialization	Materializes the data and breaks the lineage, meaning subsequent operations don't need to recompute previous transformations.	Keeps data in memory but retains lineage, allowing Spark to recompute if needed.
Performance Impact	Involves I/O overhead since data is written to disk, which can slow down performance.	Provides faster access due to in-memory storage, significantly speeding up repeated queries.
Use Cases	Best for long-running applications, iterative algorithms, or when data might be lost due to failures.	Ideal for iterative algorithms or repeated queries where the same dataset is accessed multiple times.
Eviction	Checkpointed data remains until explicitly removed or until the application ends.	Cached data can be evicted based on memory pressure (i.e., if Spark needs memory for other tasks).

Spark RDD APIs – Summary

Category	APIs
Narrow Transformations	map(), flatMap(), filter(), mapPartitions(), mapPartitionsWithIndex(), sample(), union(), intersection(), cartesian(), zip()
Wide Transformations	reduceByKey(), groupByKey(), join(), cogroup(), subtractByKey(), sortByKey(), aggregateByKey(), foldByKey()
Actions	collect(), take(), count(), first(), reduce(), saveAsTextFile(), countByKey(), foreach()
Others	cache(), persist(), unpersist(), coalesce(), repartition(), glom(), getNumPartitions()

DataFrames APIs – Creation

1. From a physical data source:
    - File storage: CSV, JSON, Parquet, Delta Lake, HUDI, Iceberg …
    - Message Queue: Apache Kafka, …
2. From a sequence / list
3. From another DataFrames

DataFrames APIs – Narrow Transformations

1. .select()
2. .withColumn()
3. .filter()
4. .drop()
5. .distinct()
6. .limit()
7. .selectExpr()
8. .replace()
9. .fillna() or .na.fill()
10. .dropna()

DataFrames APIs – Wide Transformations

1. .groupBy()
2. .join()
3. .orderBy() or sort()
4. .coalesce()
5. .repartition()
6. .rollup()
7. .cube()
8. .dropDuplicates()

DataFrames APIs – Actions

1. .show()
2. .count()
3. .collect()
4. .first()
5. .head()
6. .take()
7. .describe()
8. .toPandas()
9. .write()

DataFrames APIs – Others

1. .explain()
2. .cache()
3. .persist()
4. .unpersist()
5. .printSchema()
6. .schema
7. .sample()
8. .corr() - Computes the correlation between two columns
9. .cov() - Computes the covariance between two columns
10. .approxQuantile() - Computes approximate quantiles
11. .crossJoin() - Performs a cartesian product of two DataFrames
12. createOrReplaceTempView() - Creates a temporary SQL view

DataFrames APIs – Summary

Category	APIs
Narrow Transformations	select(), withColumn(), filter(), drop(), distinct(), limit(), selectExpr(), replace(), fillna(), dropna(), na.fill()
Wide Transformations	groupBy(), agg(), join(), orderBy(), coalesce(), repartition(), rollup(), cube(), dropDuplicates()
Actions	show(), count(), collect(), first(), head(), take(), describe(), toPandas(), write()
Others	explain(), cache(), persist(), unpersist(), printSchema(), schema(), createOrReplaceTempView(), sample(), corr(), cov(), approxQuantile(), crossJoin()

Spark SQL – Data Definition Language (DDL)

1. createOrReplaceTempView(): Creates or replaces a temporary view using a DataFrame
2. createGlobalTempView(): Creates a global temporary view accessible across all sessions.
3. createGlobalTempView(): Creates a global temporary view accessible across all sessions.
4. dropGlobalTempView(): Removes a global temporary view.
5. sql(): Executes a SQL query.
6. createExternalTable(): Defines an external table using the data stored outside of Spark.

Spark SQL – Data Manipulation Language (DML)

1. insertInto(): Inserts the content of a DataFrame into an existing table.
2. df.write.mode(): Specifies the save mode when writing data (e.g., append, overwrite).
3. saveAsTable(): Saves the content of the DataFrame as a table.
4. drop(): Deletes a table or view.

Spark SQL – Query Execution and SQL Functions

Similar to Spark DataFrames APIs in the previous sections

Spark SQL – Catalog and Metadata Operations

1. listDatabases(): Lists all databases.
2. listTables(): Lists all tables in the current database.
3. listColumns(): Lists all columns of a specific table.
4. listFunctions(): Lists all available SQL functions.
5. currentDatabase(): Returns the current database name.
6. setCurrentDatabase(): Sets the current database.
7. isCached(): Checks if a table is cached.

Spark SQL – DataFrame to SQL Interoperability

1. table(): Loads a table as a DataFrame

Spark SQL – Summary

Category	APIs
Data Definition Language (DDL)	createOrReplaceTempView(), createGlobalTempView(), dropTempView(), dropGlobalTempView(), sql(), createExternalTable()
Data Manipulation Language (DML)	insertInto(), mode(), saveAsTable(), drop()
Query Execution and SQL Functions	select(), filter(), groupBy(), agg(), join(), orderBy(), explain(), show(), cache(), unpersist()
Catalog and Metadata Operations	listDatabases(), listTables(), listColumns(), listFunctions(), currentDatabase(), setCurrentDatabase(), isCached()
DataFrame to SQL Interoperability	table(), createDataFrame(), toPandas(), toJSON()
Caching and Persistence	cache(), persist(), unpersist()
Configuration and Optimization	explain(), describe(), corr(), approxQuantile()
I/O Operations (Read/Write)	read(), write(), parquet(), csv(), json(), jdbc()

Spark SQL – Online materials

- *Official document*
- *Spark SQL Built-in Functions*

References and Online Resources

1) Setup Pyspark environment – must be ready before tutorial sessions
    1) *Setup for Windows*
    2) *Setup for MacOS (Linux)*
    3) *Advanced: Setup Pyspark Notebook using Docker*
    4) *Advanced: Setup a Spark cluster using Docker*
2) Launch simple interactive Pyspark-shell
3) Starting point: Get SparkSession
4) Creating DataFrames
5) DataFrame Operations
6) Running SQL Queries
7) Global Temporary View
8) Interoperating with RDDs
9) Scalar Functions
10) Aggregate Functions
11) UDFs – User Defined Functions
12) UDAFs - User Defined Aggregation Functions
13) View & Understand SparkUI

Notes and Recommendations

Tools and Libraries:

- Use Jupyter Notebook for all labs.
- Make heavy use of the PySpark DataFrame API for customer analysis.

Datasets:

- For datasets, you can explore freely available data on Kaggle and UCI Data Repository. These repositories contain numerous datasets related to customer analysis, e-commerce, and economics.
- By completing these labs, you'll gain hands-on experience in using Apache Spark for customer analysis, which is highly valuable for economic research.