**Document type**

CONFIDENTIAL | AUTHORIZED HPE PARTNER USE ONLY

**LAB 0: Lab Environment Setup**

1. **Install python on ubuntu using command prompt**

Run the following commands as root or user with sudo access to update the packages list and install the prerequisites:

Sudo apt update

sudo apt install software-properties-common

install python 3.8 with the following commands

sudo apt install python3.8

python3.8 –version

Output:

Python 3.8.0

Install necessary packages

Sudo apt install python3-pip

Sudo install matplotlib

Sudo install virtualenv

1. **Install java 8, using command prompt**

Run the below commands to install Java 8 on Ubuntu.

sudo apt update

sudo apt install openjdk-8-jdk openjdk-8-jre git

java -version

The output will be displayed as below

openjdk version "1.8.0\_252"

OpenJDK Runtime Environment (build 1.8.0\_252-8u252-b09-1ubuntu1-b09)

OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)

As the installation is on Linux System, we need to set JAVA\_HOME and JRE\_HOME, the environment variable, which is used to find Java libraries during runtime. We can set the variables in /etc/environment file, using the following command.

$cat >> /etc/environment <<EOL

The output will be displayed based on the file location as below.

JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

JRE\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

EOL

1. **Installation of spark 3.0.3 using command prompt**

From spark official website download spark 3.0.3. Copy the download link on the tool and it can be used with ‘wget’ command.

wget https://downloads.apache.org/spark/spark-3.0.3/spark-3.0.3-bin-hadoop2.7.tgz

Extract spark to /opt folder

sudo mkdir /opt/spark

sudo tar -xf spark\*.tgz -C /opt/spark --strip-component 1

Change the permission of the folder, so that spark has the permission to write inside it.

sudo chmod -R 777 /opt/spark

As we need to mention the whole path of spark each time we try to run on the terminal, it is recommended to configure the environment variable for spark by adding its home path to the system’s profile/bashrc file. This permits us to run the commands from the terminal regardless of whichever directory we are in at present.

echo "export SPARK\_HOME=/opt/spark" >> ~/.bashrc

echo "export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin" >> ~/.bashrc

echo "export PYSPARK\_PYTHON=/usr/bin/python3" >> ~/.bashrc

Reload the shell

source ~/.bashrc

1. **Installation of jupyter notebook**

Python3 is part of Ubuntu 20.04 version. We can now use the Python package manager ‘pip’, to install additional components.

First, we need to update the apt package index before downloading and installing the packages, using the following command.

sudo apt update

Now, we will install the pip and python header file using the following commands, as some of Jupyter’s dependencies use them.

sudo apt install python3-pip python3-dev

Let us now set up python virtual environment, into which the Jupyter notebook can be installed.

Upgrade pip and install the virtual environment packages using the following commands.

$ sudo -H pip3 install --upgrade pip

$ sudo -H pip3 install virtualenv

The -H flag makes sure that the security policy sets the home environment variable to the home directory of the user.

With virtulaenv installed and set up, we can form our own enoronment for the projects. We can create a directory and move into it to keep our project files. Let us name it as per our project requirements.

mkdir ~/my\_ML\_Projects

cd ~/my\_ ML\_Projects

Within the directory, we can create a Python virtual environment.

virtualenv ml\_project\_env

This creates a directory called ml\_project\_env within the my\_ML\_Projects directory. This environment will be having a local python and pip version which can be used for isolated python environment for Jupyter.

Let us now activate the environment using the following command

source ml\_project\_env/bin/activate

.

Your prompt should change to indicate that you are now operating within a Python virtual environment. Your command prompt will now read something like this:

(ml\_project\_env)user@host:~/ml\_project\_dir$

Now, let us install the Jypyter notebook.

pip install jupyter

Let us now run the notebook

jupyter notebook

The screen will give output as below

I 21:23:21.198 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook\_cookie\_secret

[I 21:23:21.361 NotebookApp] Serving notebooks from local directory: /home/sammy/my\_project\_dir

[I 21:23:21.361 NotebookApp] The Jupyter Notebook is running at:

[I 21:23:21.361 NotebookApp] http://localhost:8888/?token=1fefa6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72

[I 21:23:21.361 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[W 21:23:21.361 NotebookApp] No web browser found: could not locate runnable browser.

[C 21:23:21.361 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time, to login with a token:

<http://localhost:8888/?token=1fefa6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72>

jupyter notebook

If the Jupyter Notebook is run on a local computer, we can navigate to the displayed URL to connect to the notebook. If it is run on a server, we need to connect to it through SSH tunneling as explained in the next section

. To stop the notebook, CTRL+C command can be used and type Y and press enter to confirm the action. The output will be displayed as shown below.

[C 21:28:28.512 NotebookApp] Shutdown confirmed

[I 21:28:28.512 NotebookApp] Shutting down 0 kernels

**SSH Tunneling with Linux**

SSH tunneling can be performed with the following SSH command in a new local terminal window:

ssh -L 8888:localhost:8888 your\_server\_username@your\_server\_ip

The ssh command opens an SSH connection, -L on the otherhand specifies the given port on client is to be forwarded to the server-side host. So whatever is running on the second port number of server will appear as first port number on the client. The server username is your username (eg. Mary) on the server and IP address is the IP of your server, as shown in example below.

ssh -L 8888:localhost:8888 Mary @203.0.113.0

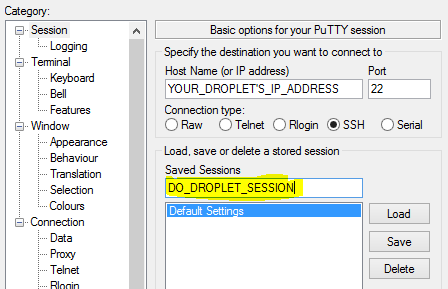
In case of no erros , we can run Jupyter notebook on the environment.

jupyter notebook

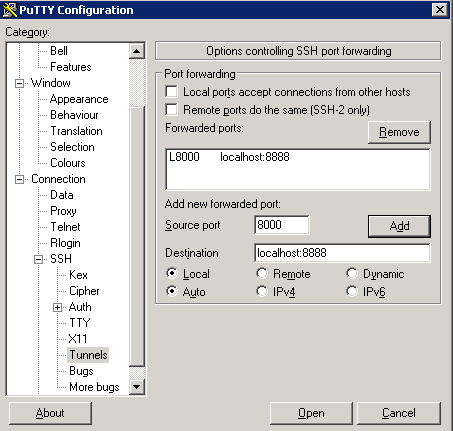
**SSH Tunneling with Windows and Putty**

If you are a Windows user, SSH tunnel can be created using Putty.

First, enter the server URL or IP address as the hostname as shown:



As the next step, click SSH on the bottom of the left pane to expand the menu and choose the option Tunnels. Enter the port number which u desire to use to access Jupyter on local machine.We may use 8000 or above as port number to avoid use of ports by other services. Set the destination as localhost:8888 , in which :8888 represents the port number of your notebook. Upon pressing add button, it willget listed in forwarded ports list as below.



Click on the open button and connect to the notebook. Make sure that token number is part of it.

**Lab Environment Setup for Elephas on AWS**

**High Level Procedure:**

1. Create 4 – EC2 Instances on AWS cloud.
2. Install Java v8, scala, git.
3. Download Spark 3.0.3 with Hadoop 2.7.
4. Set up Keyless SSH between Master and Slave nodes.
5. Install elephas.

**Detailed** **Lab Set Up Process**

**Step 1: Create 4 EC2 Instances**

* Create a new key pair.

a. Search for EC2 in the search bar

A screenshot of a computer

Description automatically generated

b. Click on EC2 Dashboard

Graphical user interface, application

Description automatically generated

c. Click on Key-pairs in the resource blade.

Graphical user interface

Description automatically generated

d. Click on Create key pair

Icon

Description automatically generated with low confidence

e. Provide a name for the key pair, select “.pem” for use with SSH and click on create key pair.

Graphical user interface, text, application, email

Description automatically generated

* Create 4 EC2 instances

- Ubuntu Server 20.04 LTS

- Instances = 4

- Subnet in US-west

- Tags: Key = Name and value=Hadoop

- select the key pair generated

- Launch Instances

**Step 2: Install Java**

* Update the Ubuntu distribution

sudo apt-get update && sudo apt-get -y dist-upgrade

* Add the Java repository

sudo add-apt-repository -y ppa:openjdk-r/ppa

sudo apt-get -qq update

* Install Java v8

sudo apt-get install -y openjdk-8-jdk --no-install-recommends

sudo update-java-alternatives -s java-1.8.0-openjdk-amd64

* Add Java to the Environment

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

**Step 3: Install Spark**

* Install Scala and git

sudo apt install scala git -y

* Download Spark

wget <https://downloads.apache.org/spark/spark-3.0.3/spark-3.0.3-bin-hadoop2.7.tgz>

* Untar the contents

tar xvf spark-\*

* Move the contents to the folder spark

sudo mv spark-3.0.3-bin-hadoop2.7 /opt/spark

* Add the spark to Environment

echo "export SPARK\_HOME=/opt/spark" >> ~/.profile

echo "export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin" >> ~/.profile

echo "export PYSPARK\_PYTHON=/usr/bin/python3" >> ~/.profile

**Step 4: Set up Keyless SSH**

* Install SSH on the Master only

sudo apt install openssh-server openssh-client

* Generate the public and private keys

cd ~/.ssh

ssh-keygen -t rsa -P ""

* Restart the ssh service

sudo service ssh restart

* Manually copy the contents of the id\_rsa.pub file into ~/.ssh/authorized\_keys file in each of the worker. (To be pasted as a single line)
* Test the ssh connection

ssh -i ~/.ssh/id\_rsa ubuntu@<private IP>

* Create a copy of the spark-env.sh.template file and rename it to spark-env.sh

cd /opt/spark/conf/

cp spark-env.sh.template spark-env.sh

* Add the following contents in the spark-env.sh file

# contents of conf/spark-env.sh

export SPARK\_MASTER\_HOST=<master-private-ip>

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# For PySpark use

export PYSPARK\_PYTHON=python3

* Create a copy of the slaves.template file and rename it to slaves

cp slaves.template slaves

* Add the following content in the file.

# contents of conf/slaves

<worker-private-ip1>

<worker-private-ip2>

<worker-private-ip3>

* Set the memory for the driver and the executor nodes.

cp spark-defaults.conf.template spark-defaults.conf

* Add the following lines in the file.

spark.driver.memory 5g

spark.driver.maxResultSize 2g

spark.executor.memory 5g

* Start the spark Master and slaves using the below command

sh /opt/spark/sbin/start-all.sh

**Step 5: Install elephas and execute the ipynb file.**

* Install elephas - in master and workers

sudo apt install python3-pip

pip install elephas

* Reboot the instances and execute the experiments

LAB 1A: Python Pandas Essentials

**Step 1**: - Clone Lab 1A from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%201/Python%20Pandas%20Essentials.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook.

**Step 3**: - Run each code block in ipynb file.

Lab 1B: - Python Scikit-learn Essentials

**Step 1**: - Clone Lab 1B from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%201/Python%20sklearn%20Essentials.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Run each code block in ipynb file

Lab 2: - create spark standalone cluster and test using word count problem - Basic Pyspark

**Step 1: -** Luanch EC2 Instance

**Step 2: -** Open Command Prompt

**Step 3: -** Change or create new directory to /home/sparkml (or select current working directory of your choice)

**Step 4: -** Clone Lab 2 from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%202/Word%20Count%20Example%20PySpark.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Run each code block in ipynb file

Lab 3: - Data preprocessing with pyspark

This lab aims to

1. Read CSV data using url
2. Detect Null Values and Imputation
3. Grouping Numerical Columns
4. Feature Selection
5. convert categorical data into binary
6. scale down values of feature between 0 to 1

**Step 1**: - Clone Lab 2 from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%203/Feature%20Engineering%20using%20PySpark.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Run each code block in ipynb file

Lab 4: - Train and Evaluate Classification Models

**Step 1**: - Clone Lab 4 Train and Evaluate Classification Models from the github repository.

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%204/Train%20and%20Evaluate%20Classification%20Model.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook.

**Step 3**: - Import the diabetes dataset using wget <https://raw.githubusercontent.com/siyad-CT/HPE_ML_WS/main/Lab%204/diabetes.csv>

**Step 4**: - For multi class classification import IRIS dataset using  wget <https://raw.githubusercontent.com/siyad-CT/HPE_ML_WS/main/Lab%204/IRIS.csv>

**Step 5**: - Run each code block in ipynb file.

Lab 5: - Train and Evaluate Regression Models

**Step 1**: - Clone Lab 5 Train and Evaluate Regression Models from the GitHub repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%205/Train%20and%20Evaluate%20Regression%20Models%20.ipynb>

**Step 2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Import the realestate dataset using wget <https://raw.githubusercontent.com/siyad-CT/HPE_ML_WS/main/Lab%205/Real%20estate.csv>

**Step 4**: - Run each code block in ipynb file

Lab 6: - Train and Evaluate Clustering Models

**Step 1**: - Clone Lab 6 Train and Evaluate Clustering Models from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/blob/main/Lab%206/Train%20and%20Evaluate%20Clustering%20Models.ipynb>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Import the retail\_loyalty\_rfm  dataset using wget <https://raw.githubusercontent.com/siyad-CT/HPE_ML_WS/main/Lab%206/retail_loyalty_rfm.csv>

**Step 4**: - Run each code block in ipynb file

Lab 7: - Hyperparameter tuning

**Step 1**: - Clone Lab 7 Hyperparameter Tuning from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%207>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Import the dataset using wget <https://raw.githubusercontent.com/siyad-CT/HPE_ML_WS/main/Lab%207/toyota.csv>

**Step 4**: - Run each code block in ipynb file

Lab 8: - Creating a Neural Network in Spark

**Step 1**: - Clone Lab 8 Creating a Neural Network from the github repository.

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%208>

**Step2**: - Open the ipynb file on Jupyter notebook

**Step 3**: - Import the dataset using wget

https://github.com/siyad-CT/HPE\_ML\_WS/blob/main/Lab%208/IRIS.csv

**Step 4**: - Run each code block in ipynb file.

Lab 9: - Predicting Fire Department Calls with Spark ML

**Step 1**: - Clone Lab 7 Hyperparameter Tuning from the github repository .

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%209>

**Step2**: - Open the ipynb file on Jupyter notebook .

**Step 3**: - Import the dataset using wget

https://github.com/siyad-CT/HPE\_ML\_WS/blob/main/Lab%209/Fire\_Department\_Calls\_for\_Service.csv

**Step 4**: - Run each code block in ipynb file

**Lab 10: - Predicting Apple Stock Market Cost with LSTM**

**Step 1**: - Clone Lab 10 from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%2010>

Step 2:Open the ipynb file on Jupyter notebook

**Step 3**: - Import the dataset using wget

[https://github.com/siyad-CT/HPE\_ML\_WS/tree/main/Lab%2010/data](https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%2010/data%0b )

**Step 4**: - Run each code block in ipynb file

**Lab 11: - Distributed deep learning on spark**

**Step 1**: - Clone Lab 11 from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%2011>

**Step 2**:Open the ipynb file on Jupyter notebook

**Step 3**: - Import the dataset using wget

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%2011/ml-latest-small>

**Step 4**: - Run each code block in ipynb file

**LAB 12:** **Lab Environment Set up for running determined.ai on local machine**

**High Level Procedure:**

1. Install Ubuntu 20.04 from Microsoft Store.
2. Configure the conda environment and determined.ai
3. Install Docker on windows 10.
4. Execute the experiments

**Detailed Procedure:**

**Step 1: Install Ubuntu**

Graphical user interface, text

Description automatically generated

**Step 2: Install Miniconda in WSL**

* Download Miniconda

sudo wget -c <https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh>

* Provide the access permission

sudo chmod +x Miniconda3-latest-Linux-x86\_64.sh

* Install Miniconda

./Miniconda3-latest-Linux-x86\_64.sh

* Activate Miniconda

conda activate

* Disable auto activation of base environment

conda config --set auto\_activate\_base false

* Create a conda environment

conda create -n determined python=3.8

* Activate the environment

conda activate determined

* Install the determined packages

pip install determined

* Clone the github repository for accessing the examples

git clone <https://github.com/determined-ai/determined>

* Shut Down WSL.

* Go to Poweshell on Windows

* Run the command to list the WSLs

wsl.exe -l -v

* Change the WSL version to 2 for the Ubuntu distribution

wsl.exe --set-version Ubuntu-20.04 2

**Step 3: Install Docker on Windows**

* Go the link [Get Docker | Docker Documentation](https://docs.docker.com/get-docker/)

Graphical user interface, text, application

Description automatically generated

* Open Docker on Windows
* Go to setting
* In the General tab enable “Use the WSL 2 based engine”
* In the Resources tab go to WSL Integration
* Check the option “Enable integration with my default WSL distro”
* Switch on docker integration for the WSL Distribution
* Apply and Restart

**Step 4: Deploy the cluster and run the experiments using determined.**

* Set up and deploy a cluster locally (run the command in WSL)

det deploy local cluster-up --no-gpu

* Verify Master and Agent are running using the Container and Images tab in the Docker

Graphical user interface, text, application

Description automatically generated

* Access the web UI of the determined by typing “localhost:8080” in a browser
* username is admin
* password empty

Graphical user interface, application, website

Description automatically generated

* Navigate to the desired experiment in the examples folder in WSL.

cd determined/examples/computer\_vision/cifar10\_pytorch

* Run the experiment using the below command

det experiment create const.yaml .

* Check the training activity in the Web UI.

**Distributed deep learning on determined**

**Step 1**: - Clone Lab 12 from the github repository

<https://github.com/siyad-CT/HPE_ML_WS/tree/main/Lab%2012>

**Step 2**:Open the ipynb file on Jupyter notebook

**Step 4**: - Run each code block in ipynb file