Snigdha Sathineni
160119733143

Data Science and Big Data Analytics

Experiment 1: Installation of required software, Programs using Numpy, Pandas and Matplotlib

AIM: Identification and Installation of required software and technology (python modules)

DESCRIPTION:
The jupyter notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document centric experience.

Libraries for python:

- Tensorflow
- Numpy
- Scipy
- Pandas
- Matplotlib
- Keras
- Scikit-learn
- Pytorch
- Scrapy
- BeautifulSoup

Numpy:
NumPy is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

Scipy:
SciPy is a scientific computation library that uses NumPy underneath.SciPy stands for Scientific Python.It provides more utility functions for optimization, stats and signal processing.Like NumPy, SciPy is open source so we can use it freely.SciPy was created by NumPy's creator Travis Olliphant.If SciPy uses NumPy underneath, why can we not just use NumPy?SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

Pandas:
Pandas is a Python library used for working with data sets.It has functions for analyzing, cleaning, exploring, and manipulating data.The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.Pandas allows us to analyze big data and make conclusions based on statistical theories.Pandas can clean messy data sets, and make them readable and relevant.Relevant data is very important in data science.

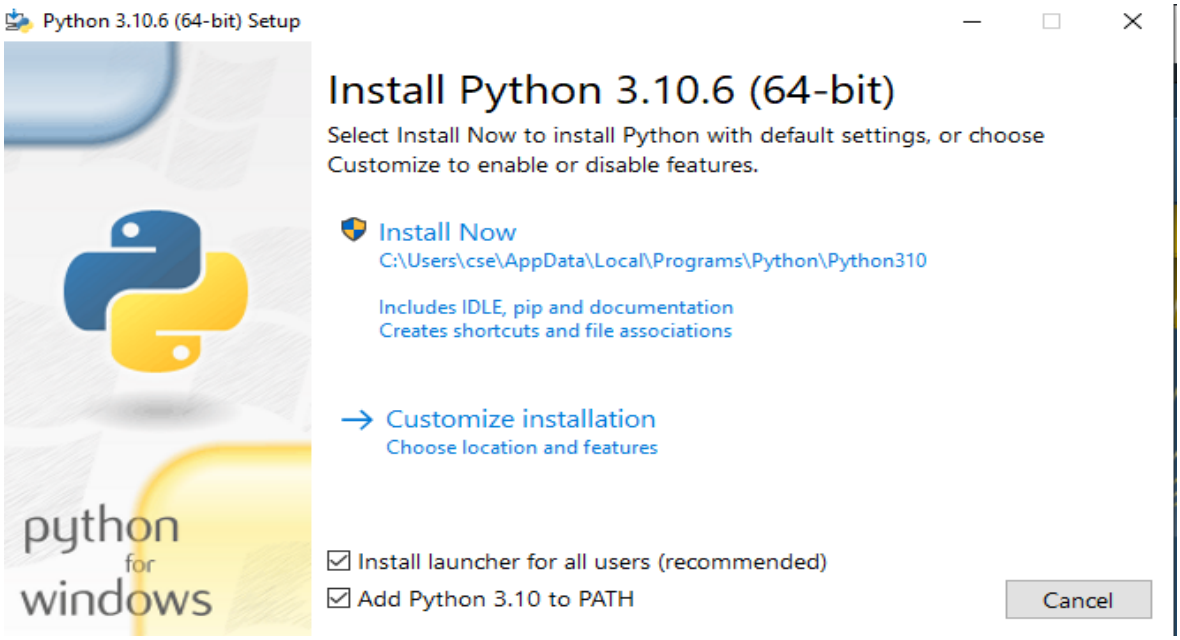
Matplotlib:
Matplotlib is a low level graph plotting library in python that serves as a visualization utility.Matplotlib was created by John D. Hunter.Matplotlib is open source and we can use it

freely.Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

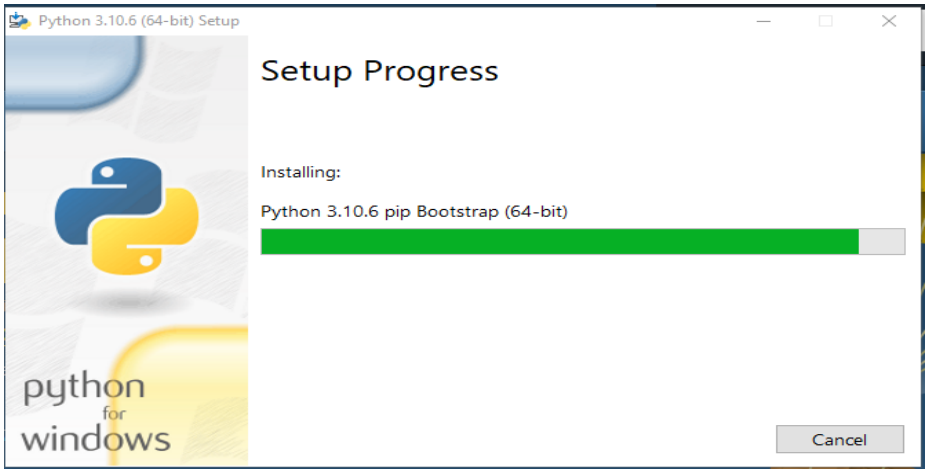
PROCEDURE:

Install python:

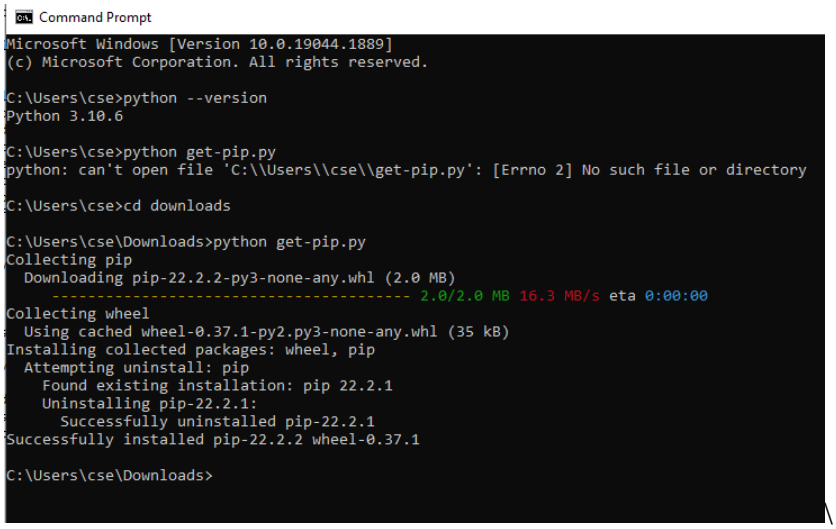
Go to python.org/downloads and download the latest version of python.



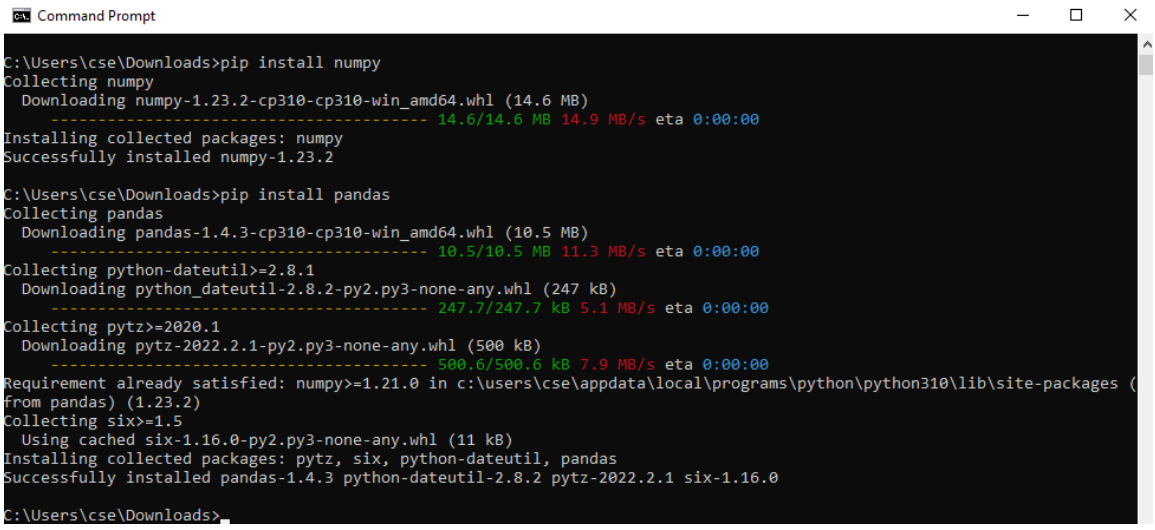
Snigdha Sathineni
160119733143



Install pip:
python get-pip.py

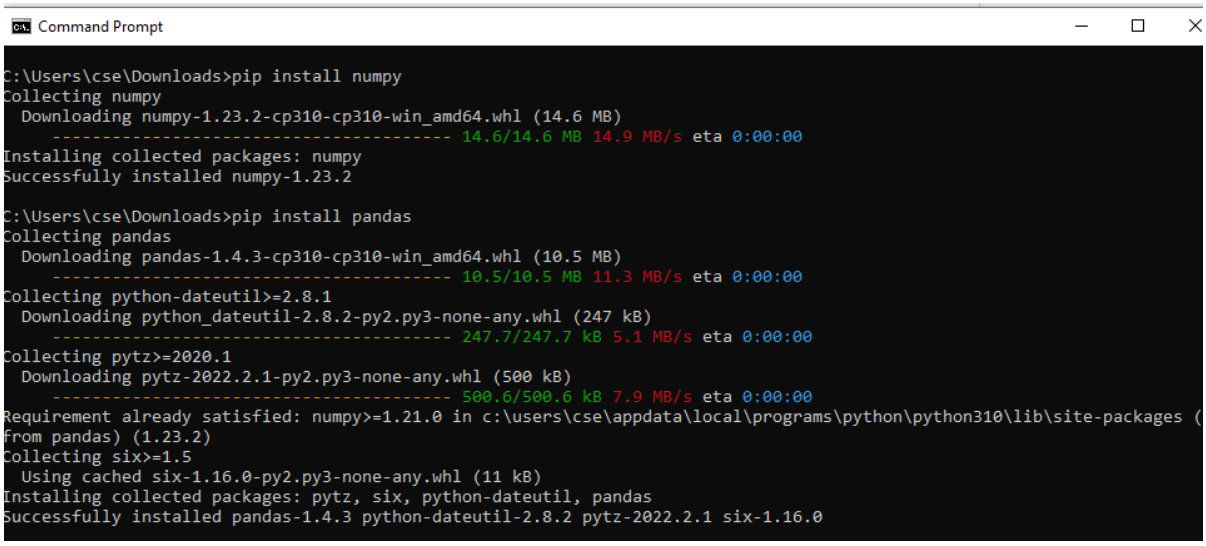


To install Numpy:
pip install numpy

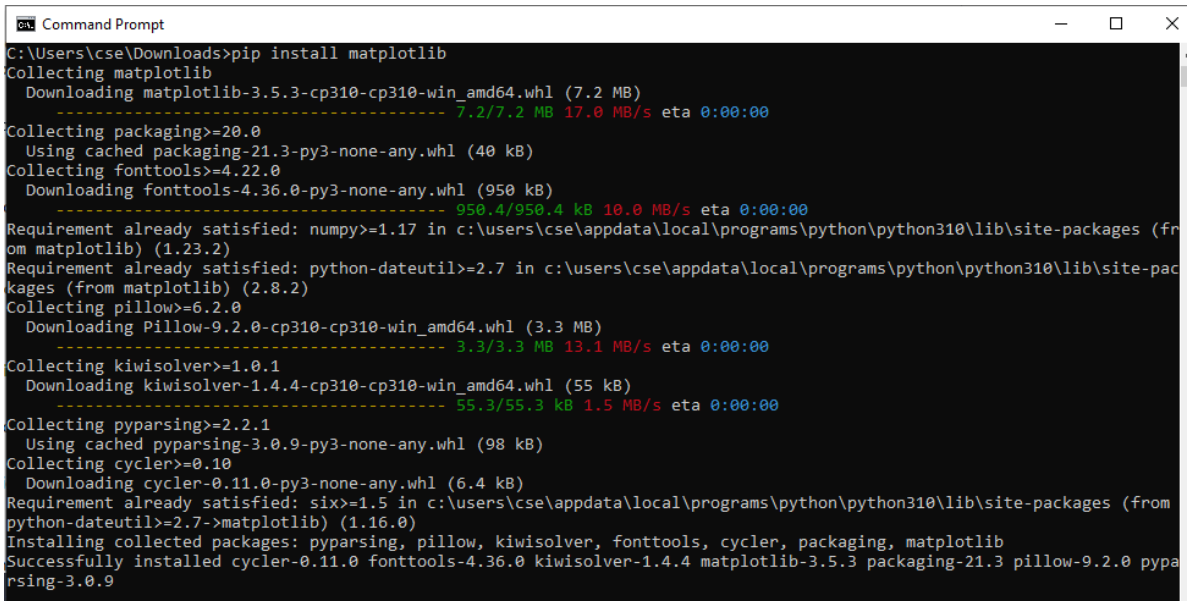


To install Pandas:

pip install pandas



To install Matplotlib:
pip install matplotlib



To install scipy:
pip install scipy

Snigdha Sathineni
160119733143

```
Command Prompt
Requirement already satisfied: numpy>=1.17 in c:\users\cse\appdata\local\programs\python\python310\lib\site-packages (fr
om matplotlib) (1.23.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\cse\appdata\local\programs\python\python310\lib\site-pac
kages (from matplotlib) (2.8.2)
Collecting pillow>=6.2.0
  Downloading Pillow-9.2.0-cp310-cp310-win_amd64.whl (3.3 MB)
     ---------------------------------------- 3.3/3.3 MB 13.1 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp310-cp310-win_amd64.whl (55 kB)
     ---------------------------------------- 55.3/55.3 kB 1.5 MB/s eta 0:00:00
Collecting pyparsing>=2.2.1
  Using cached pyparsing-3.0.9-py3-none-any.whl (98 kB)
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Requirement already satisfied: six>=1.5 in c:\users\cse\appdata\local\programs\python\python310\lib\site-packages (from
python-dateutil>=2.7->matplotlib) (1.16.0)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, packaging, matplotlib
Successfully installed cycler-0.11.0 fonttools-4.36.0 kiwisolver-1.4.4 matplotlib-3.5.3 packaging-21.3 pillow-9.2.0 pypa
rsing-3.0.9

C:\Users\cse\Downloads>pip install scipy
Collecting scipy
  Downloading scipy-1.9.0-cp310-cp310-win_amd64.whl (38.6 MB)
     ---------------------------------------- 38.6/38.6 MB 15.2 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.25.0,>=1.18.5 in c:\users\cse\appdata\local\programs\python\python310\lib\site-pa
ckages (from scipy) (1.23.2)
Installing collected packages: scipy
Successfully installed scipy-1.9.0
```

To install jupyter notebook(linux):
sudo snap install jupyter
To install jupyter notebook(windows):
pip install jupyter notebook

```
Command Prompt - pip  install jupyter notebook
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cse>pip install jupyter notebook
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting notebook
  Downloading notebook-6.4.12-py3-none-any.whl (9.9 MB)
     ---------------------------------------- 9.9/9.9 MB 17.6 MB/s eta 0:00:00
Collecting nbconvert
  Downloading nbconvert-6.5.3-py3-none-any.whl (563 kB)
     ---------------------------------------- 563.8/563.8 kB 5.9 MB/s eta 0:00:00
Collecting jupyter-console
  Downloading jupyter_console-6.4.4-py3-none-any.whl (22 kB)
Collecting qtconsole
  Downloading qtconsole-5.3.1-py3-none-any.whl (120 kB)
     ---------------------------------------- 120.8/120.8 kB 1.8 MB/s eta 0:00:00
Collecting ipykernel
  Downloading ipykernel-6.15.1-py3-none-any.whl (132 kB)
     ---------------------------------------- 132.9/132.9 kB 7.7 MB/s eta 0:00:00
Collecting ipywidgets
  Downloading ipywidgets-7.7.1-py2.py3-none-any.whl (123 kB)
     ---------------------------------------- 123.4/123.4 kB 3.6 MB/s eta 0:00:00
Collecting argon2-cffi
  Downloading argon2_cffi-21.3.0-py3-none-any.whl (14 kB)
Collecting Send2Trash>=1.8.0
  Downloading Send2Trash-1.8.0-py3-none-any.whl (18 kB)
Collecting nest-asyncio>=1.5
  Downloading nest_asyncio-1.5.5-py3-none-any.whl (5.2 kB)
Collecting pyzmq>=17
```

```
C:\Users\cse>python3
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import scipy
>>> import matplotlib
>>> import pandas
>>>
```

Snigdha Sathineni
160119733143

PROGRAMS USING THE INSTALLED MODULES:

A) Creating and printing a one-dimensional array

```
In [2]: import numpy
        arr=numpy.array([4,8,12])
        arr
Out[2]: array([ 4,  8, 12])

In [3]: type(arr)
Out[3]: numpy.ndarray

In [4]: arr.dtype
Out[4]: dtype('int32')

In [ ]:
```

B) Creating and printing a two-dimensional array

```
In [6]: arr=numpy.array([(1.1,2,3,4),(5,6,7,8)])
        arr
Out[6]: array([[1.1, 2. , 3. , 4. ],
               [5. , 6. , 7. , 8. ]])

In [7]: type(arr)
Out[7]: numpy.ndarray

In [9]: arr.dtype
Out[9]: dtype('float64')

In [ ]:

In [ ]:
```

C) Product of a two-dimensional array

```
In [11]: arr=numpy.array([(1,3,1),(2,2,2)])
         numpy.product(arr)
Out[11]: 24

In [ ]:
```

Snigdha Sathineni
160119733143

D) Indexing and slicing

```
In [12]: arr=numpy.arange(10)
         arr
Out[12]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [13]: sli=arr[3:10:2]
         sli
Out[13]: array([3, 5, 7, 9])

In [15]: c=arr>2
         d=arr[c]
         d
Out[15]: array([3, 4, 5, 6, 7, 8, 9])

In [ ]:
```

E) Iterating

```
In [15]: c=arr>2
         d=arr[c]
         d
Out[15]: array([3, 4, 5, 6, 7, 8, 9])

In [17]: import numpy as np
         a=np.arange(0,50,5)
         for i in np.nditer(a):
             print(i)

         0
         5
         10
         15
         20
         25
         30
         35
         40
         45

In [ ]:
```

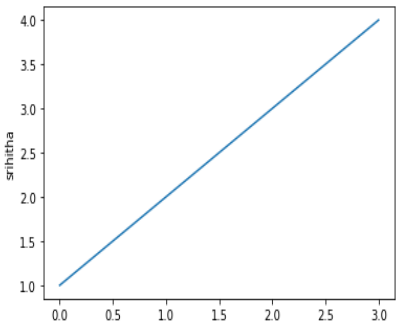
F) Reshaping

```
In [18]: a=np.arange(0,60,5)
         b=a.reshape(6,2)
         b
Out[18]: array([[ 0,  5],
                [10, 15],
                [20, 25],
                [30, 35],
                [40, 45],
                [50, 55]])

In [ ]:
```

Snigdha Sathineni
160119733143

G) Line plot using Matplotlib

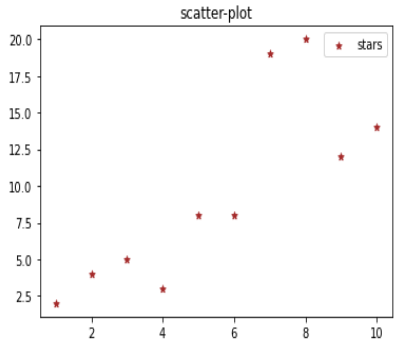
```
In [23]: import matplotlib.pyplot as plt
         plt.plot([1,2,3,4])
         plt.ylabel("srihitha")
         plt.show()
```




```
In [ ]:
```

H) Scatterplot using Matplotlib

```
In [25]: x=[1,2,3,4,5,6,7,8,9,10]
         y=[2,4,5,3,8,8,19,20,12,14]
         plt.scatter(x,y,label="stars",color='brown',marker="*",s=30)
         plt.title('scatter-plot')
         plt.legend()
         plt.show()
```

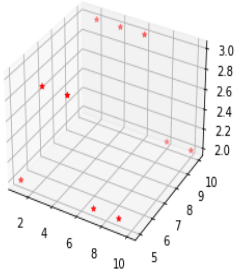



```
In [ ]:
```

Snigdha Sathineni
160119733143

I) 3D Plot using Matplotlib

```
In [28]: from mpl_toolkits.mplot3d import Axes3D
         fig=plt.figure()
         ax=fig.add_subplot(111,projection='3d')
         x=[1,2,3,4,5,6,7,8,9,10]
         y=[5,10,5,10,5,10,5,10,5,10]
         z=[2,3,3,3,3,3,2,2,2,2]
         ax.scatter(x,y,z,c='r',marker='*')
         plt.show()
```



J) DataFrame Implementation using pandas

```
In [29]: import pandas as pd
         lst=['hello','I am','Srihitha']
         df=pd.DataFrame(lst)
         print(df)
```

```
          0
0     hello
1      I am
2  Srihitha
```

```
In [ ]:
```

H) Series Implementation using python

```
In [30]: lst=['hello','I am','Srihitha']
         data=pd.Series(lst)
         print(type(data))
         print(data)
```

```
<class 'pandas.core.series.Series'>
0       hello
1        I am
2    Srihitha
dtype: object
```

```
In [ ]:
```

Snigdha Sathineni
160119733143

Data Science and Big Data Analytics

Experiment 2: Exploratory Data Analysis

AIM: To do exploratory data analysis on heart disease UCI dataset.

DESCRIPTION:

Exploratory Data Analysis (EDA) is a pre-processing step to understand the data. There are numerous methods and steps in performing EDA, however, most of them are specific, focusing on either visualization or distribution, and are incomplete. Therefore, we have to understand, explore, and extract the information from the data to answer the questions or assumptions.

Data Set Explanations:

Here we will be using the dataset consisting of 303 patients with 14 features set.

Features explanation:

1. age (Age in years)
2. sex : (1 = male, 0 = female)
3. cp (Chest Pain Type): [0: asymptomatic, 1: atypical angina, 2: non-anginal pain, 3: typical angina]
4. trestbps (Resting Blood Pressure in mm/hg)
5. chol (Serum Cholesterol in mg/dl)
6. fps (Fasting Blood Sugar > 120 mg/dl): [0 = no, 1 = yes]
7. restecg (Resting ECG): [0: showing probable or definite left ventricular hypertrophy by Estes' criteria, 1: normal, 2: having ST-T wave abnormality]
8. thalach (maximum heart rate achieved)
9. exang (Exercise Induced Angina): [1 = yes, 0 = no]
10. oldpeak (ST depression induced by exercise relative to rest)
11. slope (the slope of the peak exercise ST segment): [0: downsloping; 1: flat; 2: upsloping]
12. ca [number of major vessels (0–3)
13. thal : [1 = normal, 2 = fixed defect, 3 = reversible defect]
14. target: [0 = disease, 1 = no disease]

CODE AND ANALYSIS:

1. Import and get to know the data

Snigdha Sathineni
160119733143

In [2]:
```
# Libraries for Exploratory Data Analysis
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
df = pd.read_csv('/content/drive/My Drive/csv/github/heart.csv')
df.head(3)
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1

In [4]:
```
df.shape
```

Out[4]:
```
(303, 14)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

Snigdha Sathineni
160119733143

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

there are no nulls

Here we have 303 rows with 14 variables.

2.Data Cleaning

a) Check the data type**.**

The variables types are

- Binary: sex, fbs, exang, target
- Categorical: cp, restecg, slope, ca, thal
- Continuous: age, trestbps, chol, thalac, oldpeak

Snigdha Sathineni
160119733143

In [7]:
```
# to know the type of variable
df.nunique()
```

Out[7]:
```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

In [8]:
```
df.dtypes
```

Out[8]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

Note here that the binary and categorical variable are classified as different integer type by python. We will need to change them to ‘object’ type.

In [9]:
```
# change the categorical type to categorical variables
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
df.dtypes
```

Out[9]:
```
age           int64
sex          object
cp           object
trestbps      int64
chol          int64
fbs          object
restecg      object
thalach       int64
exang        object
oldpeak     float64
slope        object
ca           object
thal         object
target        int64
dtype: object
```

b. Check for the data characters mistakes

1.Feature ‘ca’ ranges from 0–3, however, df.nunique**()** listed 0–4. So lets find the ‘4’ and change them to NaN.

In [10]:
```
df['ca'].unique()
```

Out[10]:
```
array([0, 2, 1, 3, 4], dtype=object)
```

In [11]:
```
# to count the number in of each category decending order
df.ca.value_counts()
```

Out[11]:
```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

In [12]:
```
# to check missing values
df.isnull().sum()
```

Out[12]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

c) Check for missing values and replace them

In [12]:
```
# to check missing values
df.isnull().sum()
```

Out[12]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

d) Statistics summary

Snigdha Sathineni
160119733143

In [13]:
```
# change the labelling for better interpretation/ visualization understanding
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({1: "typical_angina",
                          2: "atypical_angina",
                          3:"non-anginal pain",
                          4: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['slope'] = df.cp.replace({1: "upsloping",
                          2: "flat",
                          3:"downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect", 3:"normal"})
```

In [14]:
```
# to know the basic stats
df.describe()
```

Out[14]:

	age	trestbps	chol	thalach	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	131.623762	246.264026	149.646865	1.039604
std	9.082101	17.538143	51.830751	22.905161	1.161075
min	29.000000	94.000000	126.000000	71.000000	0.000000
25%	47.500000	120.000000	211.000000	133.500000	0.000000
50%	55.000000	130.000000	240.000000	153.000000	0.800000
75%	61.000000	140.000000	274.500000	166.000000	1.600000
max	77.000000	200.000000	564.000000	202.000000	6.200000

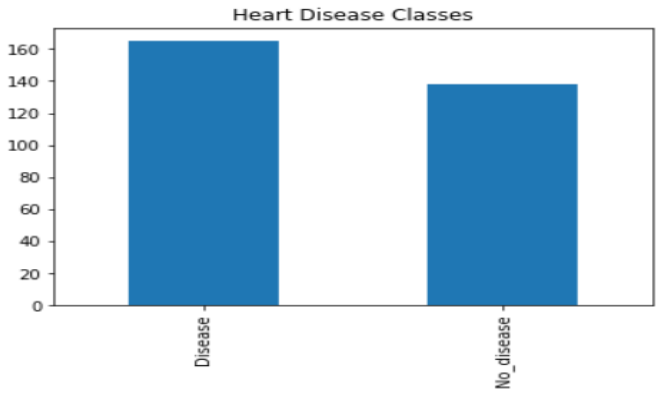
BARPLOTS:
target variable distribution

In [17]:
```
df.columns
```

Out[17]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [35]:
```
print(df.target.value_counts())
df['target'].value_counts().plot(kind='bar').set_title('Heart Disease Classes')
```

Out[35]:
```
Disease       165
No_disease    138
Name: target, dtype: int64
Text(0.5, 1.0, 'Heart Disease Classes')
```



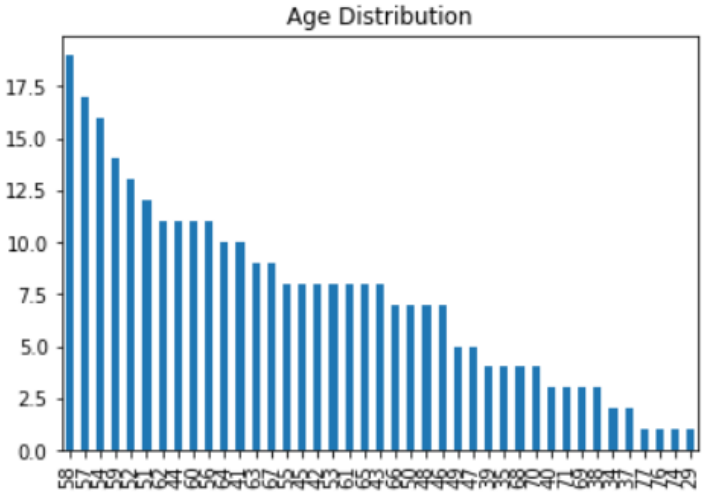
There are more diseased than healthy patients.

Age variable distribution

Snigdha Sathineni
160119733143

In [23]:
```
# print(df.age.value_counts())
df['age'].value_counts().plot(kind='bar').set_title('Age Distribution')
```

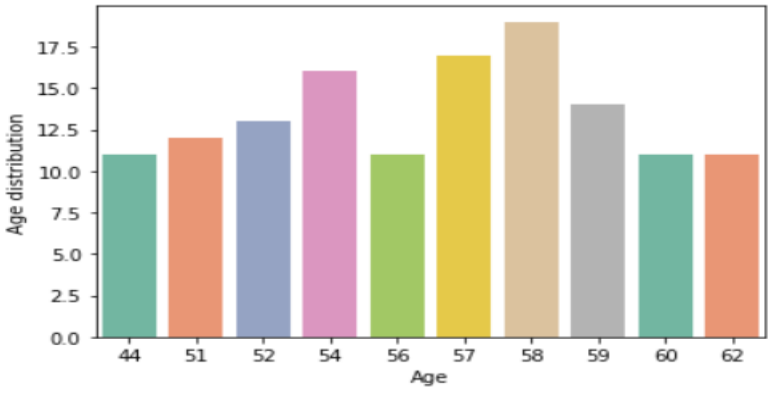
Out[23]:
```
Text(0.5, 1.0, 'Age Distribution')
```



In [24]:
```
# Analyze distribution in age in range 10
print(df.age.value_counts()[:10])
sns.barplot(x=df.age.value_counts()[:10].index,
            y=df.age.value_counts()[:10].values,
            palette='Set2')
plt.xlabel('Age')
plt.ylabel('Age distribution')
```

```
58    19
57    17
54    16
59    14
52    13
51    12
62    11
44    11
60    11
56    11
Name: age, dtype: int64
```

Out[24]:
```
Text(0, 0.5, 'Age distribution')
```



The age are normally distributed. Most of the patients are in the age between 50s to 60s. Let's take a quick look basic stats. The mean age is about 54 years with ±9.08 std, the youngest is at 29 and the oldest is at 77.

Snigdha Sathineni
160119733143

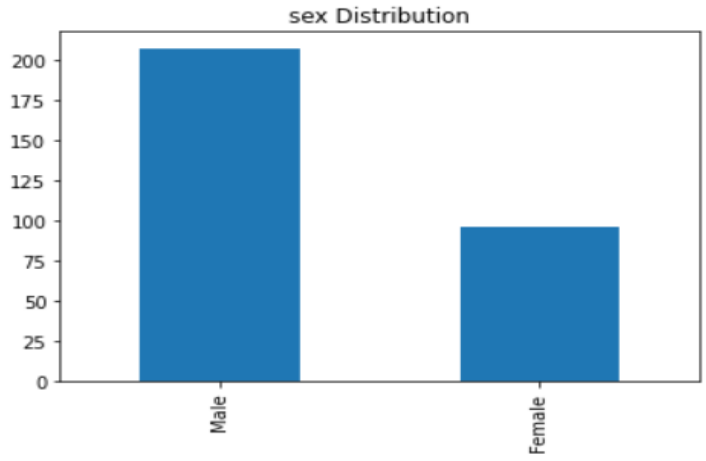
In [ ]:
```
# to know the youngest or oldest in age
print(min(df.age))
print(max(df.age))
print(df.age.mean())
```

```
29
77
54.366336633663366
```

In [25]:
```
print(df.sex.value_counts())
df['sex'].value_counts().plot(kind='bar').set_title('sex Distribution')
```

```
Male      207
Female     96
Name: sex, dtype: int64
```

Out[25]:
```
Text(0.5, 1.0, 'sex Distribution')
```

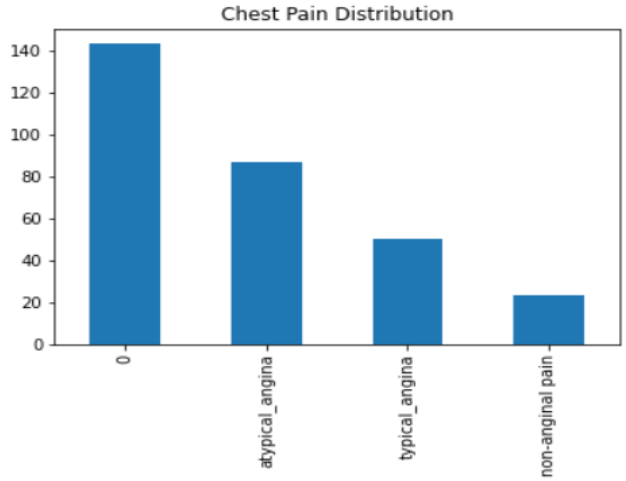


Snigdha Sathineni
160119733143

In [26]:
```
print(df.cp.value_counts())
df['cp'].value_counts().plot(kind='bar').set_title('Chest Pain Distribution')
```

```
0                  143
atypical_angina     87
typical_angina      50
non-anginal pain    23
Name: cp, dtype: int64
```

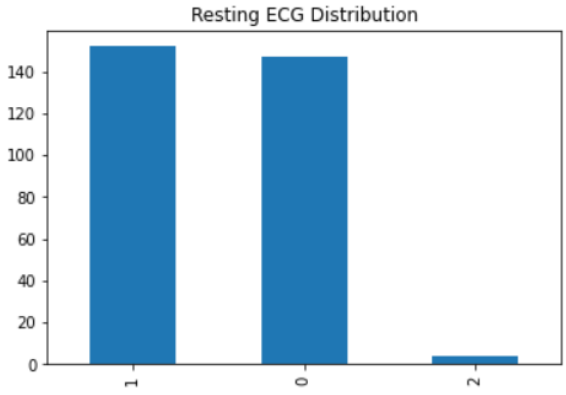
Out[26]:
```
Text(0.5, 1.0, 'Chest Pain Distribution')
```



In [43]:
```
print(df.restecg.value_counts())
df['restecg'].value_counts().plot(kind='bar').set_title('Resting ECG Distribution')
```

```
1    152
0    147
2      4
Name: restecg, dtype: int64
```

Out[43]:
```
Text(0.5, 1.0, 'Resting ECG Distribution')
```

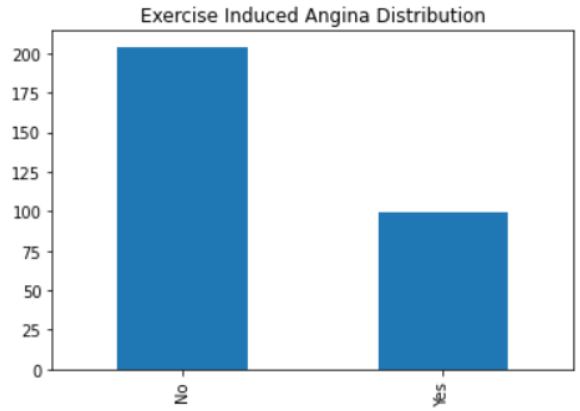


Snigdha Sathineni
160119733143

In [32]:
```
print(df.exang.value_counts())
df['exang'].value_counts().plot(kind='bar').set_title('Exercise Induced Angina Distribution')
```

```
No     204
Yes     99
Name: exang, dtype: int64
```

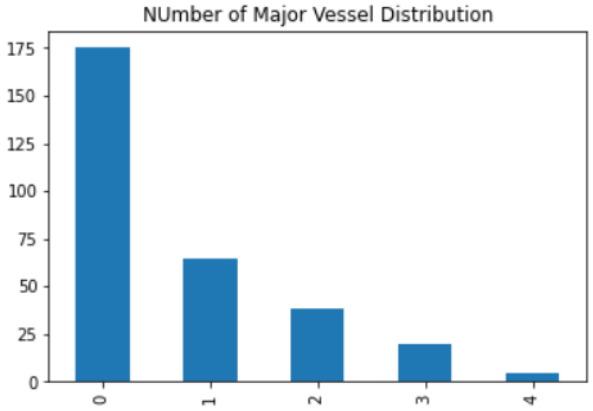
Out[32]:
```
Text(0.5, 1.0, 'Exercise Induced Angina Distribution')
```



In [33]:
```
print(df.ca.value_counts())
df['ca'].value_counts().plot(kind='bar').set_title('NUmber of Major Vessel Distribution')
```

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

Out[33]:
```
Text(0.5, 1.0, 'NUmber of Major Vessel Distribution')
```

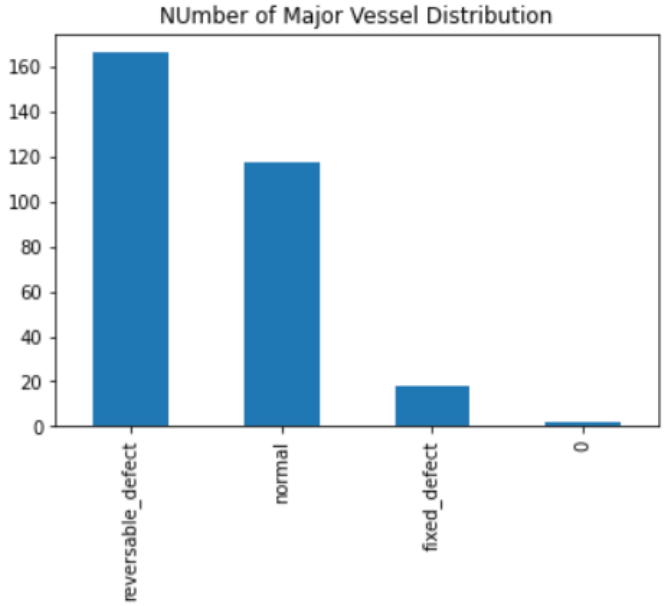


Snigdha Sathineni
160119733143

In [34]:
```
print(df.thal.value_counts())
df['thal'].value_counts().plot(kind='bar').set_title('thal Distribution')
```

```
reversable_defect    166
normal               117
fixed_defect          18
0                      2
Name: thal, dtype: int64
```

Out[34]: Text(0.5, 1.0, 'NUmber of Major Vessel Distribution')

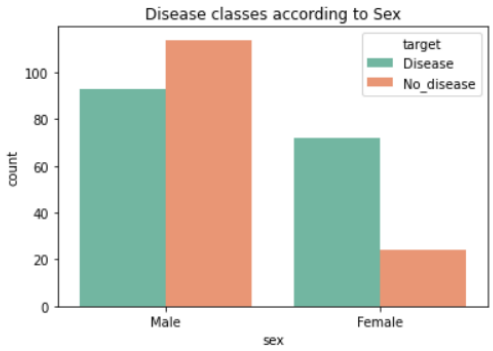


Visualize categorical data distribution -COUNTPLOTS

Gender distribution according to target variable

In [38]:
```
sns.countplot(x='sex', hue='target', data=df, palette='Set2').set_title('Disease classes according to Sex')
```

Out[38]: Text(0.5, 1.0, 'Disease classes according to Sex')



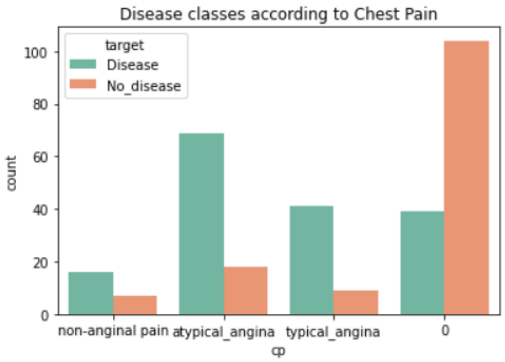
From the bar graph, we can observe that among disease patients, male are higher than female.

Snigdha Sathineni
160119733143

Chest pain distribution according to target variable

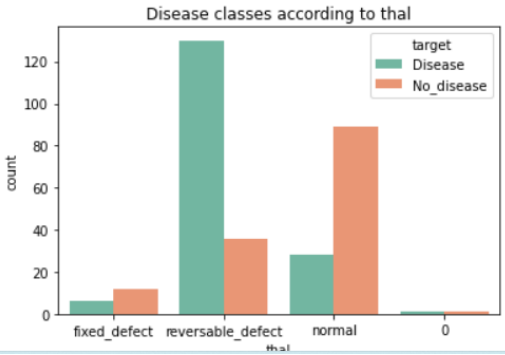
In [40]:
```
sns.countplot(x='cp', hue='target', data=df, palette='Set2').set_title('Disease classes according to Chest Pain')
```

Out[40]:
```
Text(0.5, 1.0, 'Disease classes according to Chest Pain')
```



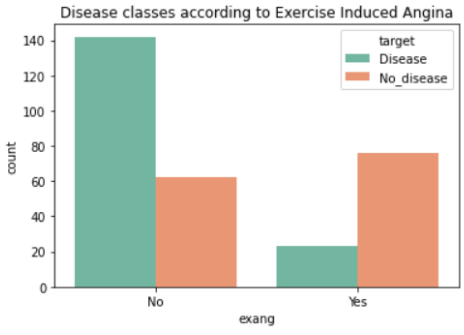
In [41]:
```
sns.countplot(x='thal', hue='target', data=df, palette='Set2').set_title('Disease classes according to thal')
```

Out[41]:
```
Text(0.5, 1.0, 'Disease classes according to thal')
```



In [44]:
```
sns.countplot(x='exang', hue='target', data=df, palette='Set2').set_title('Disease classes according to Exercise Induced Angina')
```

Out[44]:
```
Text(0.5, 1.0, 'Disease classes according to Exercise Induced Angina')
```

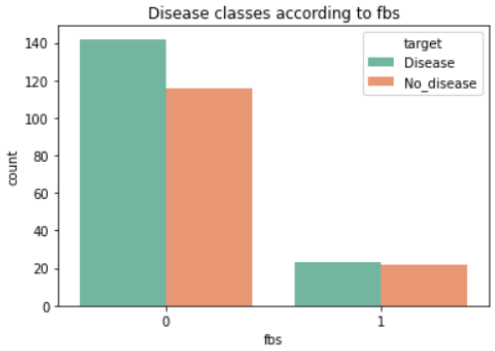


Fasting blood sugar distribution according to target variable

Snigdha Sathineni
160119733143

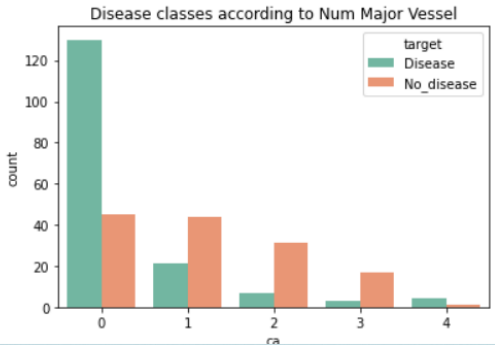
In [45]:
```
sns.countplot(x='fbs', hue='target', data=df, palette='Set2').set_title('Disease classes according to fbs')
```

Out[45]:
```
Text(0.5, 1.0, 'Disease classes according to fbs')
```



In [46]:
```
sns.countplot(x='ca', hue='target', data=df, palette='Set2').set_title('Disease classes according to Num Major Vessel')
```

Out[46]:
```
Text(0.5, 1.0, 'Disease classes according to Num Major Vessel')
```

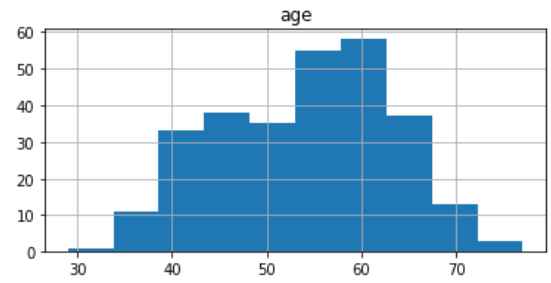


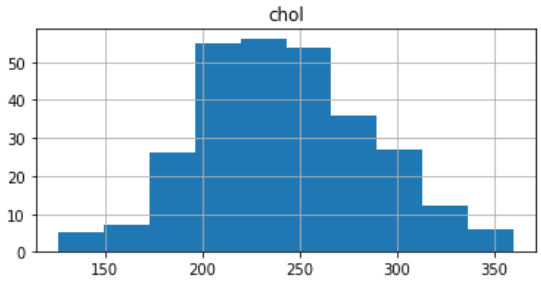
Looking at the bar graph above, its raised a question of higher number of healthy subject having typical_angina. Or in other word, most of the healthy subject having chest pain.

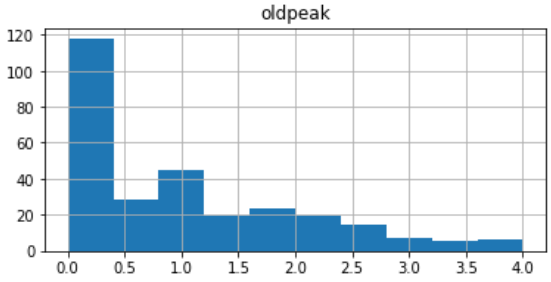
Fasting blood sugar or fbs is a diabetes indicator with fbs >120 mg/d is considered diabetic (True class). Here, we observe that the number for class true, is lower compared to class false. However, if we look closely, there are higher number of heart disease patient without diabetes. This provide an indication that fbs might not be a strong feature differentiating between heart disease an non-disease patient.

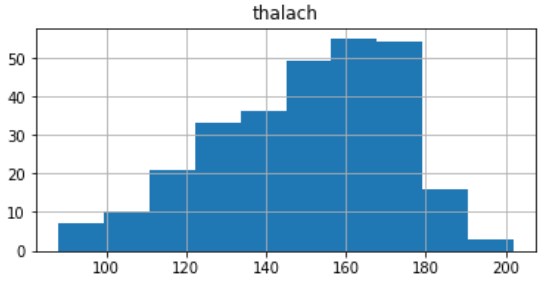
Distribution plot on continuous variables.

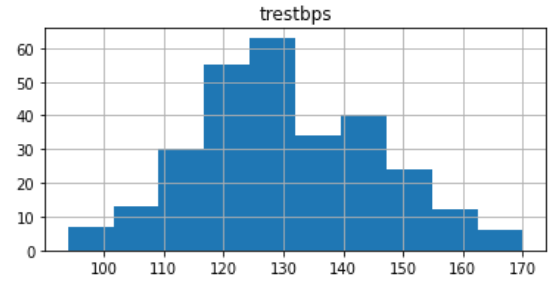
Snigdha Sathineni
160119733143











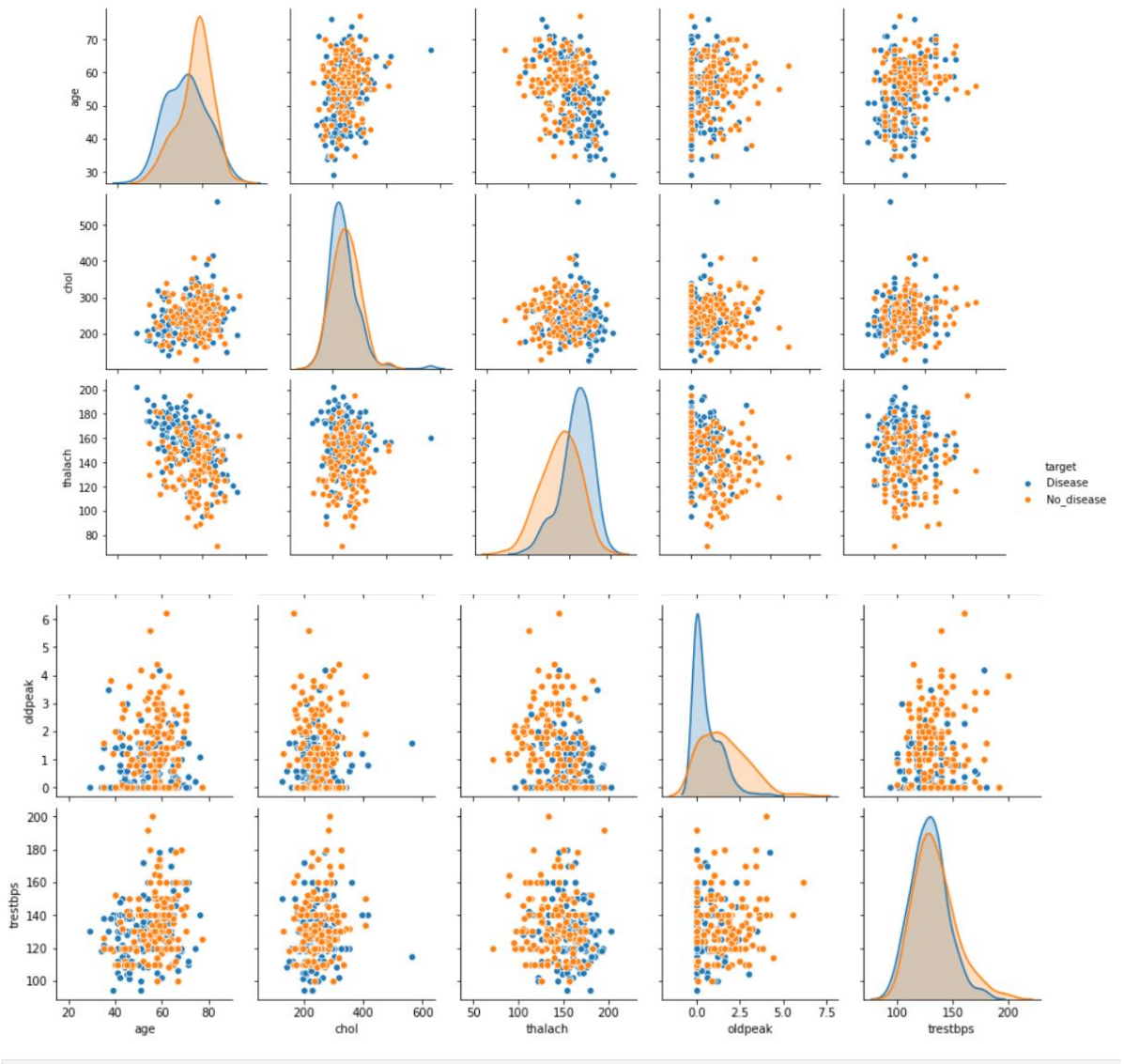
- normal distribution for: age, trestbps and almost for chol
- oldpeak is left-skewed
- thalac is right-skewed

Visualize the distribution of continuous variable across target variable-PAIRPLOTs

Snigdha Sathineni
160119733143

In [55]:
```
# define continuous variable & plot
continous_features = ['age', 'chol', 'thalach', 'oldpeak','trestbps']
sns.pairplot(df[continous_features + ['target']], hue='target')
```

Out[55]:
```
<seaborn.axisgrid.PairGrid at 0x7f562cdf38d0>
```

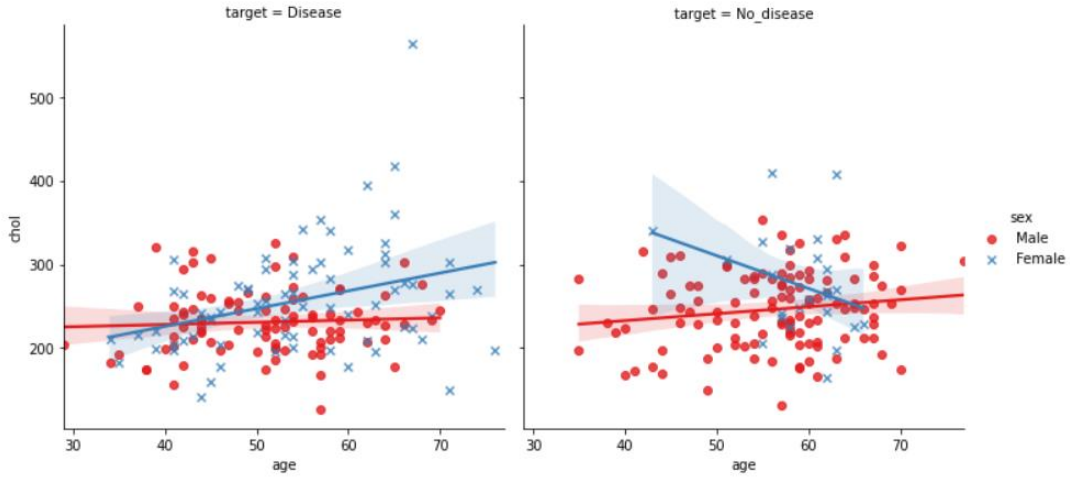


- oldpeak having a linear separation relation between disease and non-disease.
- thalach having a mild separation relation between disease and non-disease.
- Other features don’t form any clear separation

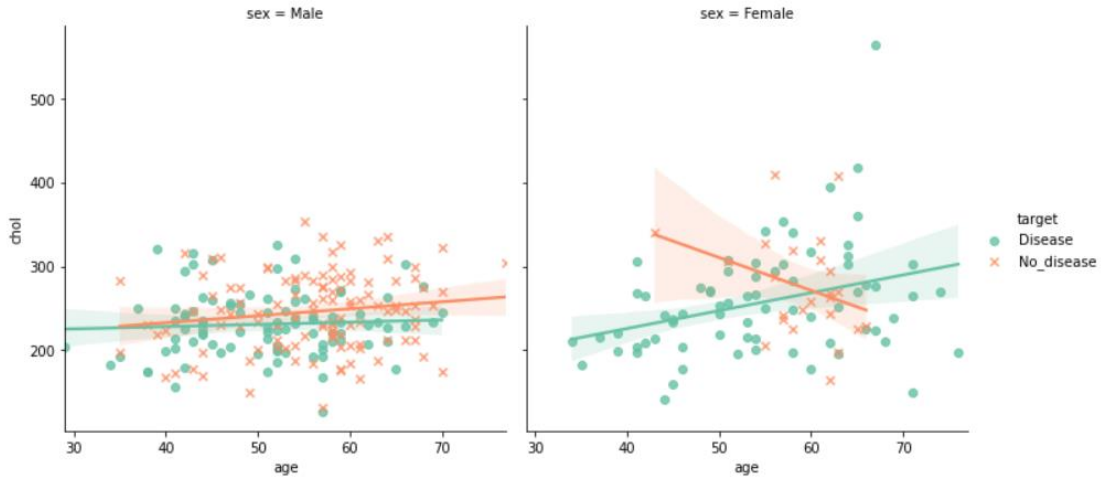
Lineplots:

Snigdha Sathineni
160119733143

In [56]:
```
# to understand the relationship between age and chol in each of the target based on sex.
sns.lmplot(x="age", y="chol", hue="sex", col="target",
           markers=["o", "x"],
           palette="Set1",
           data=df)
plt.show()
```



In [ ]:
```
# to understand the relationship between age and chol in each of the sex, based on target.
sns.lmplot(x="age",
           y="chol",
           hue="target",
           col="sex",
          #  row="target",
          #  order=2,
           markers=["o", "x"],
           palette="Set2",
           data=df)
plt.show()
```



correlation

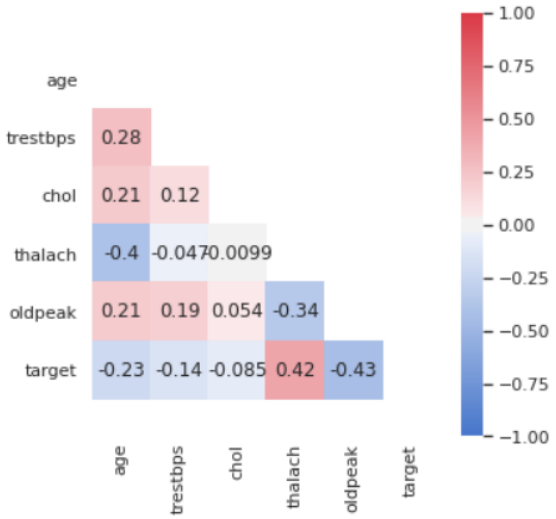
Snigdha Sathineni
160119733143

In [ ]:

```
# Correlation with Heatmap Visualization
sns.set(style="white")
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

fig, ax = plt.subplots(figsize=(5,5))
cmap = sns.diverging_palette(255, 10, as_cmap=True)
sns.heatmap(df.corr(), mask=mask, annot=True, square=True, cmap=cmap,vmin=-1, vmax=1, ax=ax)

bottom, top = ax.get_ylim()
ax.set_ylim(bottom+0.5, top-0.5)
```

Out[ ]:

```
(6.5, -0.5)
```



- ‘cp’, ‘thalach’, ‘slope’ shows good positive correlation with target
- ‘oldpeak’, ‘exang’, ‘ca’, ‘thal’, ‘sex’, ‘age’ shows a good negative correlation with target
- ‘fbs’ ‘chol’, ‘trestbps’, ‘restecg’ has low correlation with our target

Snigdha Sathineni
160119733143

Data Science and Big Data Analytics

Experiment 3: Mean, Median, Mode, Variance, Standard Deviation, Hypothesis Testing

AIM: To calculate the mean, median, mode, variance, standard deviation and perform Hypothesis Testing.

DESCRIPTION:

Hypothesis testing is a key concept in statistics, analytics, and data science. There are four steps to perform Hypothesis Testing:

- Set the Hypothesis
- Set the Significance Level, Criteria for a decision
- Compute the test statistics
- Make a decision

z -tests are a statistical way of testing a hypothesis when either:

- We know the population variance, or
- We do not know the population variance but our sample size is large $n \geq 30$

t-tests are a statistical way of testing a hypothesis when:

- We do not know the population variance
- Our sample size is small, $n < 30$

CODE AND ANALYSIS:

1. Import and get to know the data

```
In [1]: from scipy.stats import ttest_1samp
        import statistics
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from scipy import stats
        from statsmodels.stats import weightstats as stests
```

```
In [3]: salaries = pd.read_csv('Downloads/Salary.csv')
```

Snigdha Sathineni
160119733143

```
In [4]: salaries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  35 non-null     float64
 1   Salary           35 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 688.0 bytes
```

OUTPUT ANALYSIS: There are no null values in any of the fields and the data types are also correctly set, therefore code cleaning can be skipped.

2. Calculate the mean, median, mode, variance, standard deviation and plot the graphs

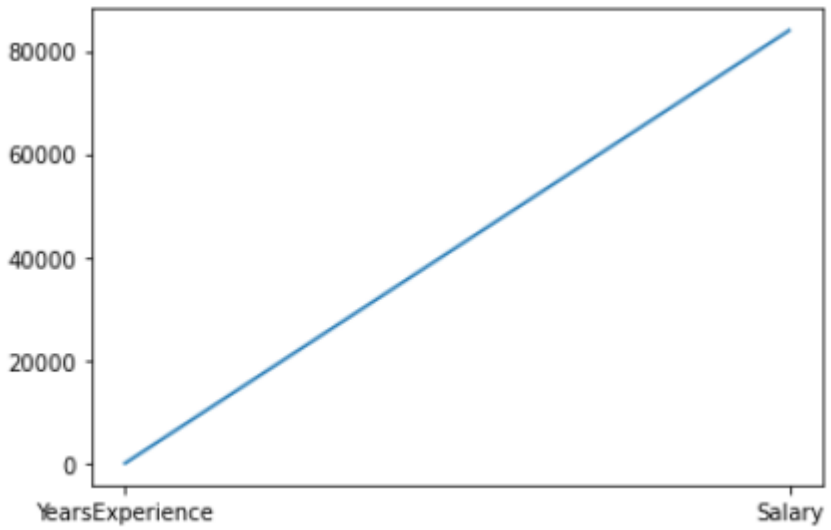
```
In [ ]: salaries = pd.read_csv('Downloads/Salary.csv')
salaries_mean = np.mean(salaries)
salaries_median = salaries.median()
salaries_mode = salaries['Salary'].mode()
salaries_variance = np.var(salaries)
salaries_deviation = np.std(salaries)
```

Snigdha Sathineni
160119733143

In [24]:
```
print('Mean:')
print(salaries_mean)
plt.plot(salaries_mean)
```

```
Mean:
YearsExperience        6.308571
Salary             83945.600000
dtype: float64
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x13cbc62fe80>]
```

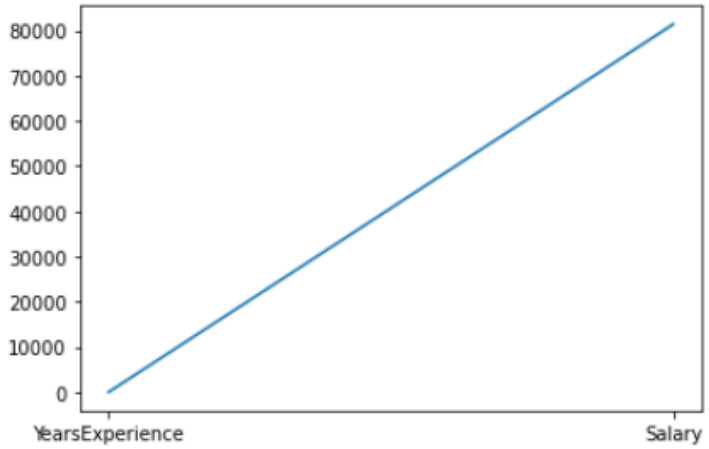


Snigdha Sathineni
160119733143

In [25]:
```
print('Median:')
print(salaries_median)
plt.plot(salaries_median)
```

```
Median:
YearsExperience        5.3
Salary             81363.0
dtype: float64
```

Out[25]: `[<matplotlib.lines.Line2D at 0x13cbc695430>]`

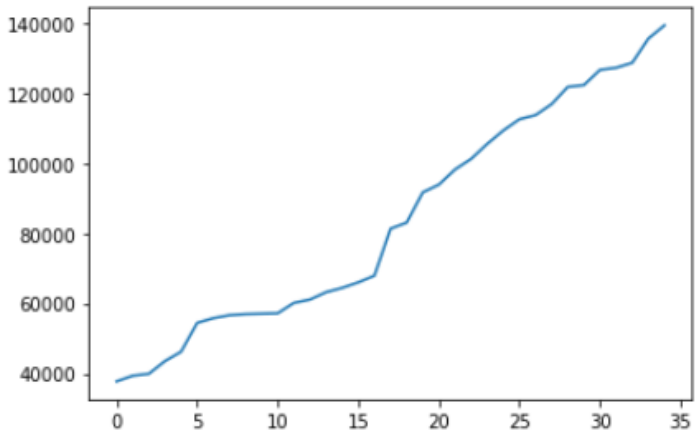


In [26]:
```
print('Mode:')
print(salaries_mode)
plt.plot(salaries_mode)
```

```
Mode:
0      37731
1      39343
2      39891
3      43525
4      46205
5      54445
6      55794
7      56642
8      56957
9      57081
10     57189
11     60150
12     61111
13     63218
14     64445
15     66029
16     67938
17     81363
18     83088
19     91738
```

Snigdha Sathineni
160119733143

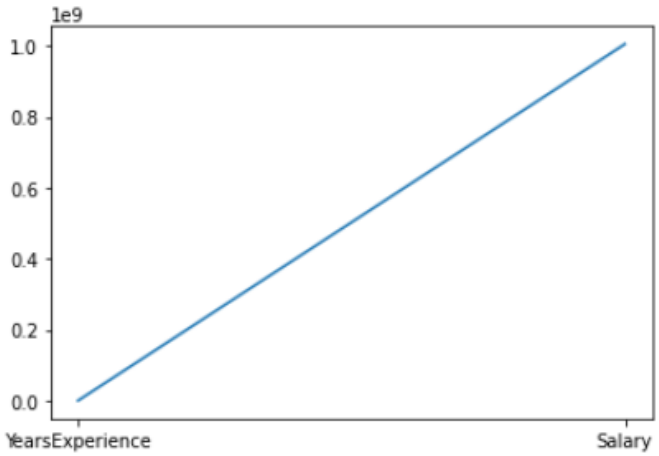
Out[26]: [<matplotlib.lines.Line2D at 0x13cbc6ed9a0>]



In [28]:
```
print('Variance:')
print(salaries_variance)
plt.plot(salaries_variance)
```

```
Variance:
YearsExperience    1.272021e+01
Salary             1.004882e+09
dtype: float64
```

Out[28]: [<matplotlib.lines.Line2D at 0x13cba51b5e0>]

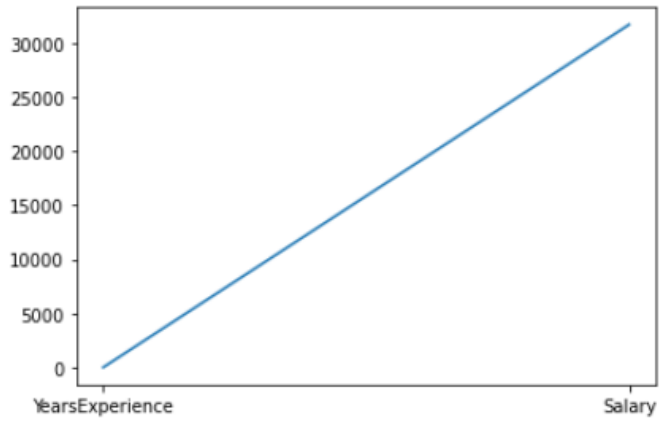


Snigdha Sathineni
160119733143

```
In [29]: print('Standard Deviation:')
         print(salaries_deviation)
         plt.plot(salaries_deviation)
```

```
Standard Deviation:
YearsExperience        3.566541
Salary             31699.876602
dtype: float64
```

```
Out[29]: [<matplotlib.lines.Line2D at 0x13cbc7b8280>]
```



3. Perform Hypothesis Testing – one sample t-test, two independent sample t-test, z-test

```
In [34]: #take 20 salaries and checking whether avg salary is 55000
         #1 sample t-test popmean=55000 df<30 variance unknown

         df = pd.read_csv("Downloads/Salary.csv")
         print(df.Salary[0:20])
         salary_mean = np.mean(df.Salary[0:20])
         print(salary_mean)
         tset, pval = ttest_1samp(df.Salary[0:20], 55000)
         print('p-values',pval)
         if pval < 0.05:    # alpha value is 0.05 or 5%
            print(" we are rejecting null hypothesis")
         else:
           print("we are accepting null hypothesis")
```

```
0     39343
1     46205
2     37731
3     43525
4     39891
5     56642
6     60150
7     54445
8     64445
9     57189
10    63218
11    55794
12    56957
13    57081
Name: Salary, dtype: int64
59304.25
p-values 0.20762262668116735
we are accepting null hypothesis
```

Snigdha Sathineni
160119733143

OUTPUT ANALYSIS: Since the sample size is less than 30 and we do not know the population variance, t-test is suitable for hypothesis testing. For the 20 salaries, the average is indeed less than 55000, hence we are accepting the null hypothesis.

```
In [5]: #take 20 salaries and checking whether avg salary is 25000
        #1 sample t-test popmean=55000 df<30 variance unknown

        from scipy.stats import ttest_1samp
        import numpy as np
        df = pd.read_csv("Downloads/Salary.csv")
        print(df.Salary[0:20])
        salary_mean = np.mean(df.Salary[0:20])
        print(salary_mean)
        tset, pval = ttest_1samp(df.Salary[0:20], 25000)
        print('p-values',pval)
        if pval < 0.05:    # alpha value is 0.05 or 5%
           print(" we are rejecting null hypothesis")
        else:
          print("we are accepting null hypothesis")
```

```
0     39343
1     46205
2     37731
3     43525
4     39891
5     56642
6     60150
7     54445
8     64445
9     57189
10    63218
11    55794
Name: Salary, dtype: int64
59304.25
p-values 2.800037525837218e-09
 we are rejecting null hypothesis
```

OUTPUT ANALYSIS: Since the sample size is less than 30 and we do not know the population variance, t-test is suitable for hypothesis testing. For the 20 salaries, the average is not less than 25000, hence we are rejecting the null hypothesis.

Snigdha Sathineni
160119733143

In [6]:
```
#2 independent sample t-test  df<30 variance unknown

from scipy.stats import ttest_ind
import numpy as np
df = pd.read_csv("Downloads/Salary.csv")
sal1=df.Salary[0:15]
sal2=df.Salary[10:35]
sal1_mean = np.mean(sal1)
sal2_mean = np.mean(sal2)
print("sal1 mean value:",sal1_mean)
print("sal2 mean value:",sal2_mean)
sal1_std = np.std(sal1)
sal2_std = np.std(sal2)
print("sal1 std value:",sal1_std)
print("sal2 std value:",sal2_std)
ttest,pval = ttest_ind(sal1,sal2)
print("p-value",pval)
if pval <0.05:
  print("we reject null hypothesis")
else:
  print("we accept null hypothesis")
```

```
sal1 mean value: 52915.13333333333
sal2 mean value: 97541.2
sal1 std value: 8766.096173072456
sal2 std value: 26942.53837039116
p-value 4.666024428230427e-07
we reject null hypothesis
```

OUTPUT ANALYSIS: Since the sample size is less than 30 and we do not know the population variance, t-test is suitable for hypothesis testing.

In [33]:
```
#z test

ztest ,pval = stests.ztest(salaries['Salary'], x2=None, value=79000)
print(float(pval))
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.362977782994247
accept null hypothesis
```

OUTPUT ANALYSIS: Since the sample size is greater than 30, z-test is suitable for hypothesis testing. For the given salaries, the average is less than 79000, hence we are accepting the null hypothesis.

Snigdha Sathineni
160119733143

Snigdha Sathineni
160119733143

Data Science and Big Data Analytics

Experiment 4: Chi-Square Test, ANOVA, Correlation

Aim: To perform Chi-square test, ANOVA, Pearson Correlation, Correlation Heatmap.

Description: An Analysis of Variance (ANOVA) is a statistical test employed to compare twoor more means together, which are determined through the analysis of variance. One-way ANOVA tests are utilized to analyze differences between groups and determine if the differences are statistically significant.

One-way ANOVAs compare two or more independent group means, though in practice theyare most oftenly used when there are at least three independent groups.

In order to carry out an ANOVA on the Gapminder dataset, we'll need to transform some ofthe features, as these values in the dataset are continuous but ANOVA analyses are appropriate for situations where one variable is categorical and one variable is quantitative.

The Chi-Square test of independence is utilized when both explanatory and response variables are categorical. You likely also want to use the Chi-Square test when the explanatory variable is quantitative and the response variable is categorical, which you cando by dividing the explanatory variable into categories.

The Chi-Square test of independence is a statistical test used to analyze how significant a relationship between two categorical variables is. When a Chi-Square test is run, every category in one variable has its frequency compared against the second variable's categories. This means that the data can be displayed as a frequency table, where the rowsrepresent the independent variables and the columns represent the dependent variables.

The Pearson Correlation test is used to analyze the strength of a relationship between twoprovided variables, both quantitative in nature. The value, or strength of the Pearson correlation, will be between +1 and -1.

A correlation of 1 indicates a perfect association between the variables, and the correlation is either positive or negative. Correlation coefficients near 0 indicate very weak, almost non-existent, correlations. While there are other ways of measuring correlations between two variables, such as Spearman Correlation or Kendall Rank Correlation, Pearson correlation is probably the most commonly used correlational test.

STEP – 1:

Import Modules

```
import statsmodels.formula.api as smf

import statsmodels.stats.multicomp as multi
import scipy
```

Snigdha Sathineni
160119733143

```
from scipy.stats import pearsonr
import pandas as pd
from seaborn import regplot
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

HeatMap

CODE:

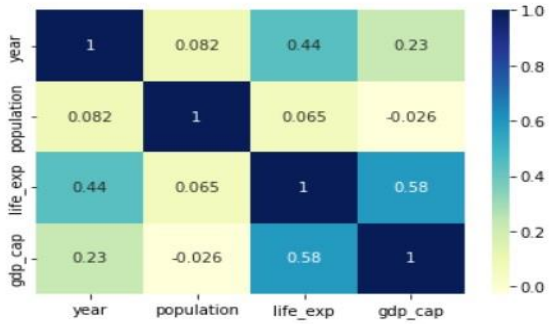
```
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sb

# import file with data
data = pd.read_csv("countrylifgdp.csv")
print(data.info())


print(data.corr())

# plotting correlation heatmap
dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)

# displaying heatmap
mp.show()
```

OUTPUT:

Snigdha Sathineni
160119733143

```
In [4]: # prints data that will be plotted
        # columns shown here are selected by corr() since
        # they are ideal for the plot
        print(data.corr())

        # plotting correlation heatmap
        dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)

        # displaying heatmap
        mp.show()
```

```
                year  population  life_exp   gdp_cap
year        1.000000    0.082308  0.435611  0.227318
population  0.082308    1.000000  0.064955 -0.025600
life_exp    0.435611    0.064955  1.000000  0.583706
gdp_cap     0.227318   -0.025600  0.583706  1.000000
```



Output Analysis:

Life_exp and gdp_cap have the highest positive correlation, population and gdp_cap has negative correlation.

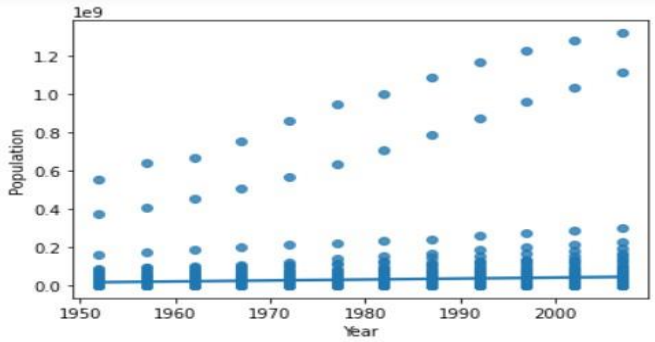
2) Pearson Correlation

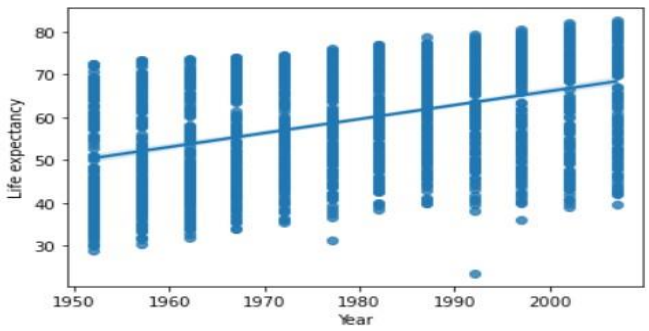
CODE:

```
df_clean = data.dropna()

def plt_regression(x, y, data, label_1, label_2):
    reg_plot = regplot(x=x, y=y, fit_reg=True,
    data=data)plt.xlabel(label_1)
    plt.ylabel(label_2)
    plt.show()

plt_regression('year', 'population', df_clean, 'Year', 'Population')
```

```
plt_regression('year', 'life_exp', df_clean, 'Year', 'Life expectancy')

print('Assoc. - life expectancy and internet use rate')
print(pearsonr(df_clean['year'], df_clean['population']))

print('Assoc. - between employment rate and internet use rate')
print(pearsonr(df_clean['year'], df_clean['life_exp']))
```

OUTPUT:






```
Assoc. - life expectancy and internet use rate
(0.08230807783830922, 0.0006716006496141012)
Assoc. - between employment rate and internet use rate
(0.4356112240540735, 7.54679462559958e-80)
```

Output Analysis:

As we saw in the heatmap year, population and year, life_Exp have positive correlation in the same way pearson graph also has a positive slope.

3) ANOVA

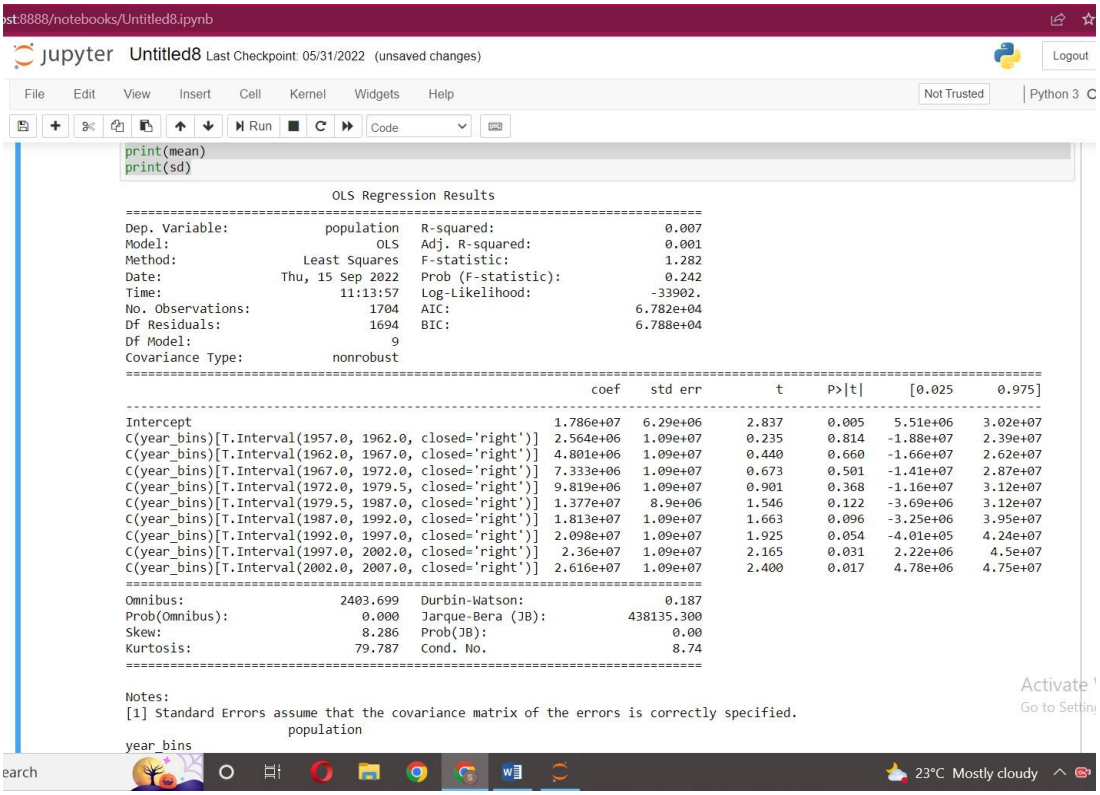
Snigdha Sathineni
160119733143

CODE:

```
def bin(dataframe, cols):
        # Create new columns that store the binned data
   for col in cols:
      new_col_name = "{}_bins".format(col)
      dataframe[new_col_name] = pd.qcut(dataframe[col], 10)

df3 = data.copy()
cols = ['year']
cols1 = ['population']
norm_cols = ['population' , 'gdp_cap' , 'life_exp']
# This creates new columns filled with the binned column data
bin(df3, cols)
bin(df3, norm_cols)
bin(df3, cols1)

anova_df = df3[['year_bins', 'population']].dropna()

relate_df = df3[['year_bins', 'population']]

anova = smf.ols(formula='population ~ C(year_bins)', data=anova_df).fit()

print(anova.summary())

# We may also want to check the mean and standard deviation for the groups
mean = relate_df.groupby("year_bins").mean()
sd = relate_df.groupby("year_bins").std()
print(mean)
print(sd)
```

Snigdha Sathineni
160119733143

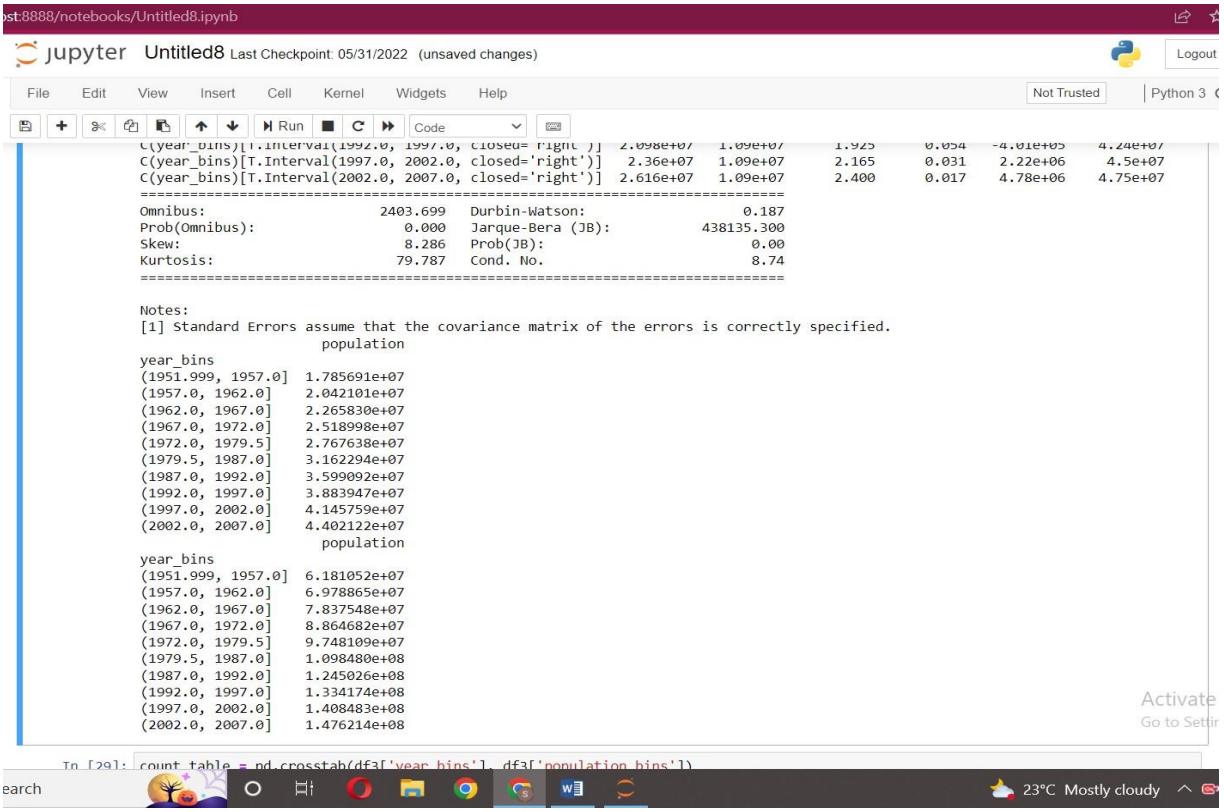
OUTPUT:



```
print(mean)
print(sd)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             population   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.282
Date:                Thu, 15 Sep 2022   Prob (F-statistic):              0.242
Time:                        11:13:57   Log-Likelihood:                -33902.
No. Observations:                1704   AIC:                         6.782e+04
Df Residuals:                    1694   BIC:                         6.788e+04
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================================================
                                                                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------------
Intercept                                                      1.786e+07   6.29e+06      2.837      0.005    5.51e+06    3.02e+07
C(year_bins)[T.Interval(1957.0, 1962.0, closed='right')]       2.564e+06   1.09e+07      0.235      0.814   -1.88e+07    2.39e+07
C(year_bins)[T.Interval(1962.0, 1967.0, closed='right')]       4.801e+06   1.09e+07      0.440      0.660   -1.66e+07    2.62e+07
C(year_bins)[T.Interval(1967.0, 1972.0, closed='right')]       7.333e+06   1.09e+07      0.673      0.501   -1.41e+07    2.87e+07
C(year_bins)[T.Interval(1972.0, 1979.5, closed='right')]       9.819e+06   1.09e+07      0.901      0.368   -1.16e+07    3.12e+07
C(year_bins)[T.Interval(1979.5, 1987.0, closed='right')]       1.377e+07    8.9e+06      1.546      0.122   -3.69e+06    3.12e+07
C(year_bins)[T.Interval(1987.0, 1992.0, closed='right')]       1.813e+07   1.09e+07      1.663      0.096   -3.25e+06    3.95e+07
C(year_bins)[T.Interval(1992.0, 1997.0, closed='right')]       2.098e+07   1.09e+07      1.925      0.054   -4.01e+05    4.24e+07
C(year_bins)[T.Interval(1997.0, 2002.0, closed='right')]        2.36e+07   1.09e+07      2.165      0.031    2.22e+06     4.5e+07
C(year_bins)[T.Interval(2002.0, 2007.0, closed='right')]       2.616e+07   1.09e+07      2.400      0.017    4.78e+06    4.75e+07
==============================================================================
Omnibus:                     2403.699   Durbin-Watson:                   0.187
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           438135.300
Skew:                           8.286   Prob(JB):                         0.00
Kurtosis:                      79.787   Cond. No.                         8.74
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                  population
year_bins
```

Snigdha Sathineni
160119733143



OUTPUT Analysis:

P-Value is 0.242 > 0.05 [Level of Significance] which implies that we don't have a significantrelationship between population and year.

4) Chi Square Test

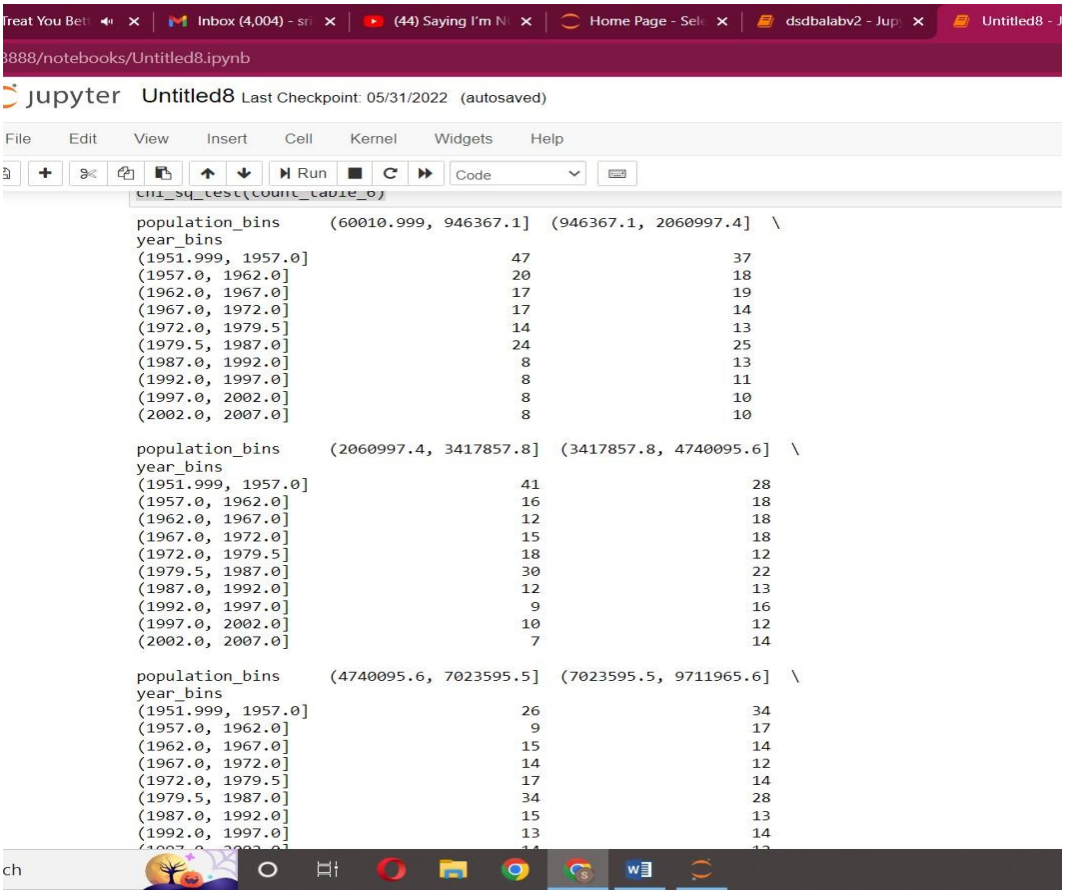
CODE:

```
count_table = pd.crosstab(df3['year_bins'], df3['population_bins'])

print(count_table)


def chi_sq_test(table):

    print("Results for:")

    print(str(table))


    # Get column percentages
```

```
    col_sum = table.sum(axis=0)
    col_percents = table/col_sum
    print(col_percents)

    chi_square = scipy.stats.chi2_contingency(table)
    print("Chi-square value, p-value, expected_counts")
    print(chi_square)

    print()

print("Initial Chi-
square:")
chi_sq_test(count_table)
print(" ")

chi_sq_test(count_table_3)
chi_sq_test(count_table_4)
chi_sq_test(count_table_5)
chi_sq_test(count_table_6)
```

OUTPUT:

Snigdha Sathineni
160119733143



population_bins	(60010.999, 946367.1]	(946367.1, 2060997.4]	\
year_bins			
(1951.999, 1957.0]	47	37	
(1957.0, 1962.0]	20	18	
(1962.0, 1967.0]	17	19	
(1967.0, 1972.0]	17	14	
(1972.0, 1979.5]	14	13	
(1979.5, 1987.0]	24	25	
(1987.0, 1992.0]	8	13	
(1992.0, 1997.0]	8	11	
(1997.0, 2002.0]	8	10	
(2002.0, 2007.0]	8	10	

population_bins	(2060997.4, 3417857.8]	(3417857.8, 4740095.6]	\
year_bins			
(1951.999, 1957.0]	41	28	
(1957.0, 1962.0]	16	18	
(1962.0, 1967.0]	12	18	
(1967.0, 1972.0]	15	18	
(1972.0, 1979.5]	18	12	
(1979.5, 1987.0]	30	22	
(1987.0, 1992.0]	12	13	
(1992.0, 1997.0]	9	16	
(1997.0, 2002.0]	10	12	
(2002.0, 2007.0]	7	14	

population_bins	(4740095.6, 7023595.5]	(7023595.5, 9711965.6]	\
year_bins			
(1951.999, 1957.0]	26	34	
(1957.0, 1962.0]	9	17	
(1962.0, 1967.0]	15	14	
(1967.0, 1972.0]	14	12	
(1972.0, 1979.5]	17	14	
(1979.5, 1987.0]	34	28	
(1987.0, 1992.0]	15	13	
(1992.0, 1997.0]	13	14	

DATA SCIENCE AND BIG DATA ANALYTICS LAB

16011933143
Snigdha Reddy

EXPERIMENT 5

AIM: To demonstrate Associative rule mining using apriori algorithm

DESCRIPTION:

The Association rule is a learning technique that helps identify the dependencies between two data items. The apriori algorithm is to create the association rule between different objects. It describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. It is used to gain insight into the structured relationships between different items involved. The most prominent practical application of the algorithm is to recommend products based on the products already present in the user's cart.

CODE:

```
!pip install apyori
from apyori import apriori
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('Market_Basket_Optimisation.csv',header=None)
df.fillna(0,inplace=True)
df.head()

records = []
for i in range(1, len(df)):
    records.append([str(df.values[i, j]) for j in range(0, len(df.columns))])

association_rules = apriori(records, min_support=0.0045, min_confidence=0.2,
min_lift=3, min_length=2)
association_results = list(association_rules)

#for i in range(0, len(association_results)):
#    print(association_results[i])


for item in association_results:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + str(list(item[0])) + " -> " + str(list(item[2][0][1])))
```

```
print("Support: " + str(item[1]))
print("Confidence: " + str(item[2][0][2]))
print("Lift: " + str(item[2][0][3]))
print("====================================")
```

```
Rule: ['light cream', 'chicken'] -> ['chicken']
Support: 0.004533333333333334
Confidence: 0.2905982905982906
Lift: 4.843304843304844
====================================
Rule: ['mushroom cream sauce', 'escalope'] -> ['escalope']
Support: 0.005733333333333333
Confidence: 0.30069930069930073
Lift: 3.7903273197390845
====================================
Rule: ['escalope', 'pasta'] -> ['escalope']
Support: 0.005866666666666667
Confidence: 0.37288135593220345
Lift: 4.700185158809287
====================================
Rule: ['herb & pepper', 'ground beef'] -> ['ground beef']
Support: 0.016
Confidence: 0.3234501347708895
Lift: 3.2915549671393096
====================================
Rule: ['ground beef', 'tomato sauce'] -> ['ground beef']
Support: 0.005333333333333333
Confidence: 0.37735849056603776
Lift: 3.840147461662528
====================================
Rule: ['whole wheat pasta', 'olive oil'] -> ['olive oil']
Support: 0.008
Confidence: 0.2714932126696833
Lift: 4.130221288078346
====================================
Rule: ['shrimp', 'pasta'] -> ['shrimp']
Support: 0.005066666666666666
Confidence: 0.3220338983050848
Lift: 4.514493901473151
====================================
```

```
====================================
Rule: ['frozen vegetables', 'tomatoes', 'spaghetti'] -> ['tomatoes']
Support: 0.006666666666666667
Confidence: 0.23923444976076555
Lift: 3.497579674864993
====================================
Rule: ['ground beef', 'spaghetti', 'grated cheese'] -> ['ground beef']
Support: 0.005333333333333333
Confidence: 0.3225806451612903
Lift: 3.282706701098612
====================================
Rule: ['herb & pepper', 'ground beef', 'mineral water'] -> ['ground beef']
Support: 0.006666666666666667
Confidence: 0.390625
Lift: 3.975152645861601
====================================
Rule: ['herb & pepper', 'ground beef', 'spaghetti'] -> ['ground beef']
Support: 0.0064
Confidence: 0.3934426229508197
Lift: 4.003825878061259
====================================
Rule: ['ground beef', 'milk', 'olive oil'] -> ['olive oil']
Support: 0.004933333333333333
Confidence: 0.22424242424242424
Lift: 3.411395906324912
====================================
Rule: ['shrimp', 'ground beef', 'spaghetti'] -> ['spaghetti']
Support: 0.006
Confidence: 0.5232558139534884
Lift: 3.004914704939635
====================================
Rule: ['spaghetti', 'milk', 'olive oil'] -> ['olive oil']
Support: 0.0072
Confidence: 0.20300751879699247
Lift: 3.0883496774390333
====================================
```

DATA SCIENCE AND BIG DATA ANALYTICS LAB

160119733143

Snigdha Reddy

EXPERIMENT 6

AIM: To perform time Series Forecasting with ARIMA model

DESCRIPTION:

A time series is simply a series of data points ordered in time. In a time series, time is often the independent variable and the goal is usually to make a forecast for the future. Time-Series Forecasting is the process of using a statistical model to predict future values of a time-series based on past results.

Components of a Time-Series are :

1. Trend - Shows a general direction of the time series data over a long period of time.
2. Seasonality - Exhibits a trend that repeats with respect to timing, direction, and magnitude.
3. Cyclical Component - These are the trends with no set repetition over a particular period of time. A cycle refers to the period of ups and downs.
4. Irregular Variation - These are the fluctuations in the time series data which become evident when trend and cyclical variations are removed.

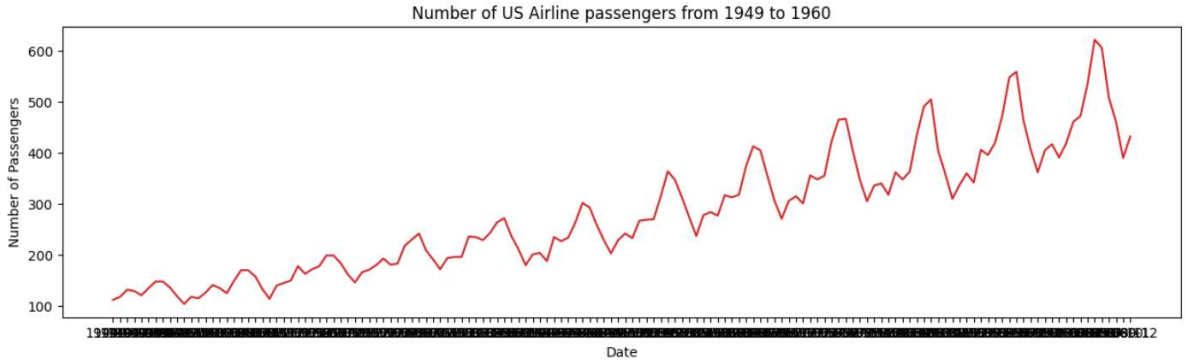
CODE:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("AirPassengers.csv")
df.head()
```

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

```
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Number of Passengers', dpi=100):
    plt.figure(figsize=(15,4), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
```

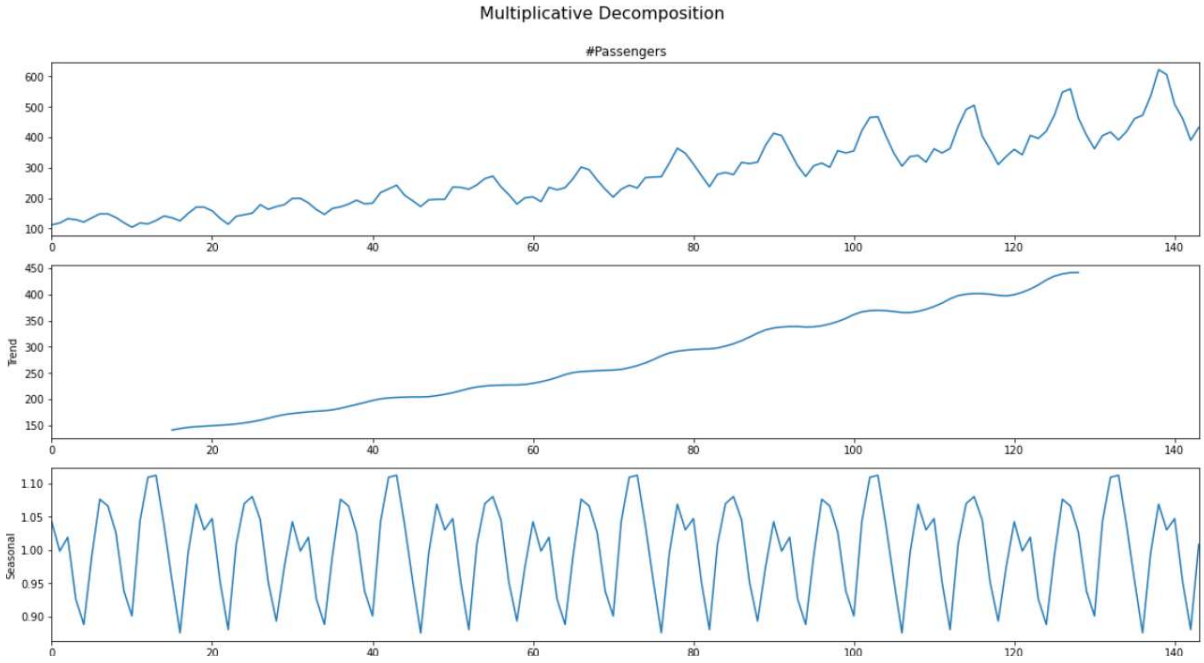
```
plot_df(df, x=df['Month'], y=df['#Passengers'], title='Number of US Airline passengers from 1949 to 1960')
```

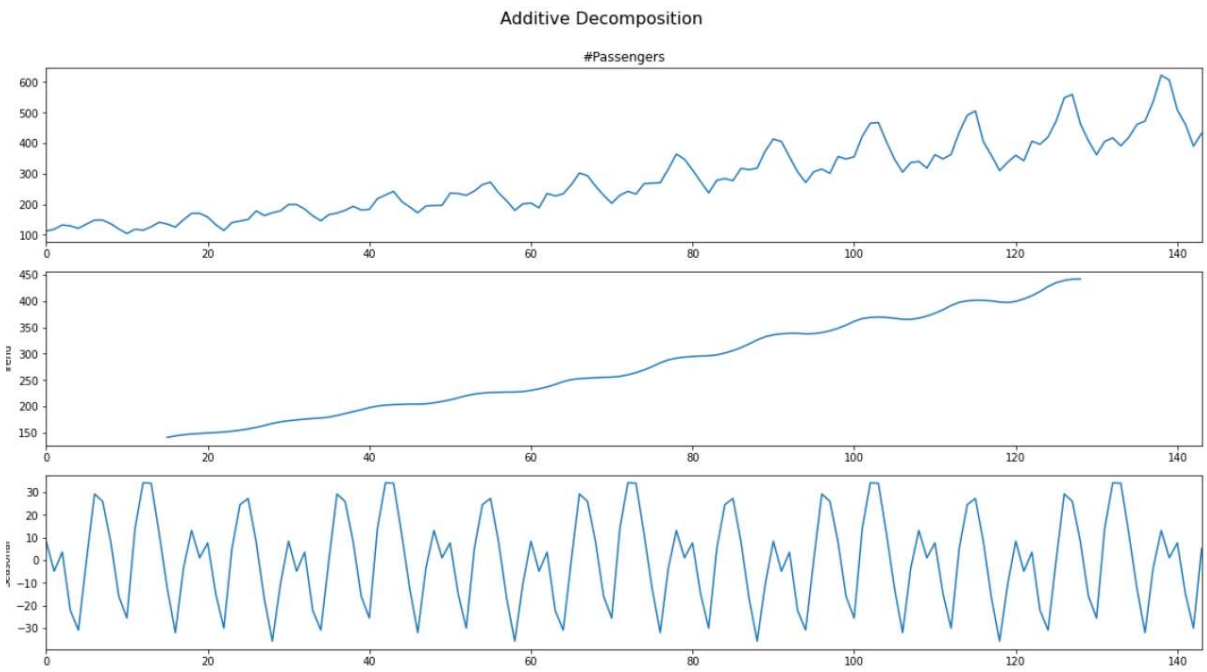



```
from statsmodels.tsa.seasonal import seasonal_decompose
from dateutil.parser import parse
```

```
# Multiplicative Decomposition
multiplicative_decomposition = seasonal_decompose(df['#Passengers'], model='multiplicative', period=30)
```

```
# Additive Decomposition
additive_decomposition = seasonal_decompose(df['#Passengers'],
model='additive', period=30)

# Plot
plt.rcParams.update({'figure.figsize': (16,12)})
multiplicative_decomposition.plot().suptitle('Multiplicative
Decomposition', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])

additive_decomposition.plot().suptitle('Additive Decomposition',
fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])

plt.show()
```






```
#test to check if data is stationary or not using Augmented
Dickey-Fuller test (ADF)
#Null hypothesis (H0): The time series data is non-stationary.
#Alternate hypothesis (H1): The time series is stationary (or
trend-stationary).

import warnings
warnings.filterwarnings("ignore")
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import kpss

result=adfuller (df['#Passengers'])
print("ADF Test result:")
print('Test Statistic: %f' %result[0])
print('p-value: %f' %result[1])
print('Critical values:')
for key, value in result[4].items ():
     print('\t%s: %.3f' %(key, value))

#If the test statistic is greater than the critical value we accept
null hypothesis. This indicates that the data is non-stationary.

#test to check if data is stationary or not using Kwiatkowski Phillips
Schmidt Shin test (KPSS)
#Null hypothesis (H0): The time series is stationary (or
```

```
trend-stationary).
#Alternate hypothesis (H1): The time series data is non-stationary.

result_kpss_ct=kpss(df['#Passengers'],regression="ct")
print("\nKPSS Test results :")
print('Test Statistic: %f' %result_kpss_ct[0])
print('p-value: %f' %result_kpss_ct[1])
print('Critical values:')
for key, value in result_kpss_ct[3].items():
     print('\t%s: %.3f' %(key, value))

#If the test statistics value is greater than the critical value, the
null hypothesis is rejected. This indicates that the data is
non-stationary.
```

```
ADF Test result:
Test Statistic: 0.815369
p-value: 0.991880
Critical values:
        1%: -3.482
        5%: -2.884
        10%: -2.579

KPSS Test results :
Test Statistic: 0.096150
p-value: 0.100000
Critical values:
        10%: 0.119
        5%: 0.146
        2.5%: 0.176
        1%: 0.216
```

```
#Using differencing to transform/stationarize the dataset

def difference(dataset, interval=1):
  diff = list()
  diff.append(dataset[0])

  for i in range(interval, len(dataset)):
    value = dataset[i] - dataset[i - interval]
    diff.append(value)
  return diff
```

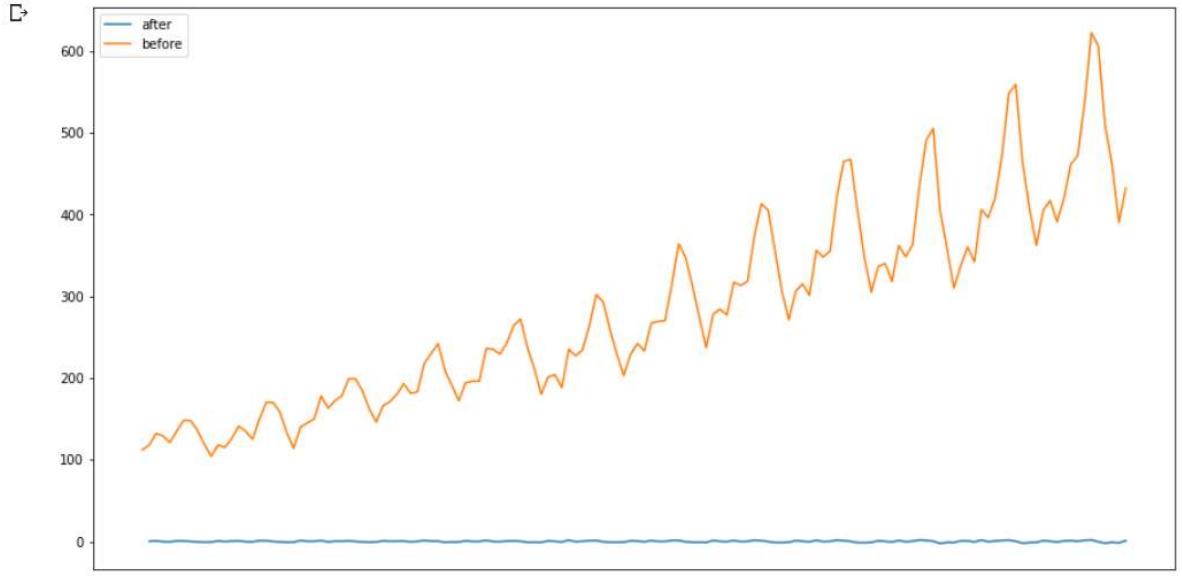
```
new_df = difference(df['#Passengers'])

plt.plot(new_df,label="after")
plt.plot(df['#Passengers'], label="before")
plt.show()

result=adfuller(new_df)
print("ADF Test result:")
print('Test Statistic: %f' %result[0])
print('p-value: %f' %result[1])
print('Critical values:')
for key, value in result[4].items ():
     print('\t%s: %.3f' %(key, value))
```

```
ADF Test result:
Test Statistic: -3.155112
p-value: 0.022734
Critical values:
        1%: -3.481
        5%: -2.884
        10%: -2.579
```

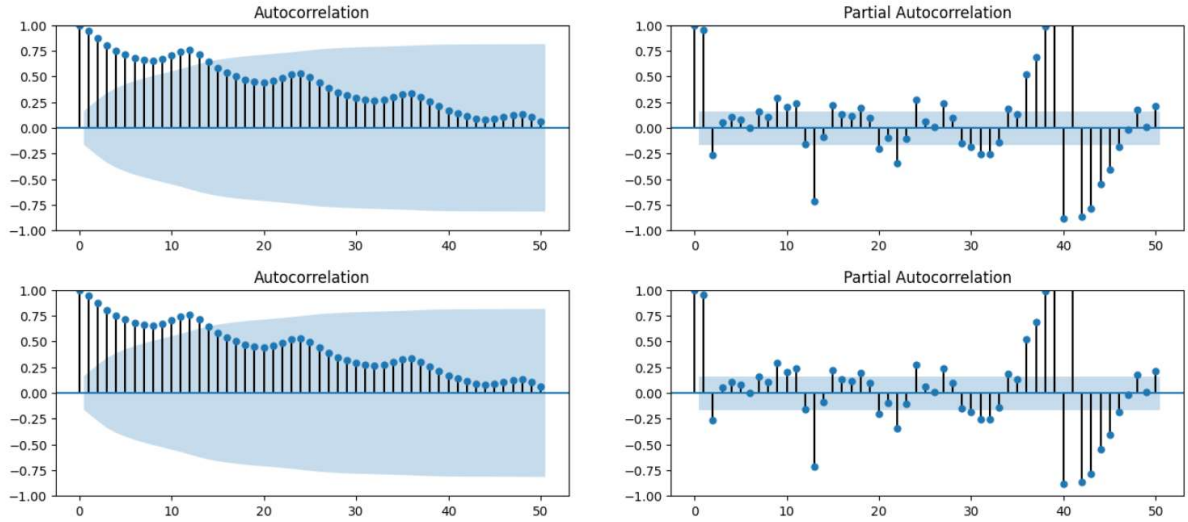
```
#Using log transforming of transform the data

df_log=np.sqrt(df['#Passengers'])
df_diff=df_log.diff().dropna()

result=adfuller(df_diff)
print("ADF Test result:")
print('Test Statistic: %f' %result[0])
print('p-value: %f' %result[1])
print('Critical values:')
for key, value in result[4].items ():
     print('\t%s: %.3f' %(key, value))
```

```
ADF Test result:
Test Statistic: -3.186422
p-value: 0.020784
Critical values:
        1%: -3.482
        5%: -2.884
        10%: -2.579
```

```
plt.figure(figsize=(15,8))
plt.plot(df_diff,label="after")
plt.plot(df['#Passengers'], label="before")
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
plt.legend()
plt.show()
```

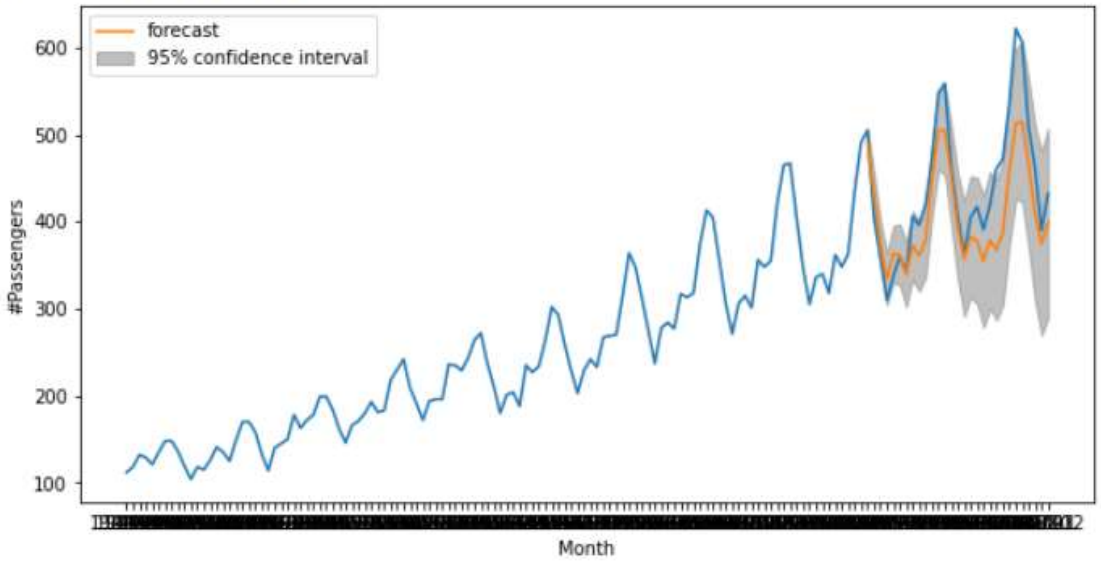



```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf,
plot_predict

fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(df['#Passengers'].tolist(), lags=50, ax=axes[0])
plot_pacf(df['#Passengers'].tolist(), lags=50, ax=axes[1])
```



```
from statsmodels.tsa.arima.model import ARIMA
import pmdarima as pm

test_data = df['#Passengers'][int(len(df)*0.8):]
train_data = df['#Passengers'][:int(len(df)*0.8)]
ARIMA_model = pm.auto_arima(train_data, m=12)
ARIMA_model.summary()
model = ARIMA(train_data, order=(12, 1, 1))
model_fit = model.fit()
model_fit.summary()
fig, axs = plt.subplots(1, 1, figsize=(10, 5))
plt.plot(df['Month'],df['#Passengers'])
plot_predict(model_fit, start=len(train_data), end=(len(df)-1), ax=axs)
plt.xlabel('Month')
plt.ylabel('#Passengers')
```

```
Text(0, 0.5, '#Passengers')
```



DSBDA LAB 7

16019733143
Snigdha Reddy

AIM:- To apply various ML classification algorithms on a dataset and compare its efficiency

DESCRIPTION:-

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.Random decision forests correct for decision trees' habit of overfitting to their training set.

CODE:-

#reading and description the data

```
import pandas as pd
df = pd.read_csv('melb_data.csv')
# print a summary of the data in Melbourne data
df.describe()
#desicion TREE
melbourne_features = ['Rooms','Landsize', 'Lattitude', 'Longtitude','BuildingArea']
#selecting features
X =df[melbourne_features]
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

```
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 0)

melbourne_model = DecisionTreeRegressor()
melbourne_model.fit(train_X, train_y)

# get predicted prices on validation data
val_predictions = melbourne_model.predict(val_X)
print(melbourne_model.score(val_predictions,val_y))
print(mean_absolute_error(val_y, val_predictions))
#random forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
print(forest_model.score(melb_preds,val_y))
print(mean_absolute_error(val_y, melb_preds))
# naive bayes
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train)
pred=y_pred.predict(val_x)
print(gnb.score(pred,val_y))
print(mean_asolute error(val_y,pred))
```

OUTPUT:-

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Propertycount
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000	13518.000000	13580.000000	7130.000000	8205.000000	13580.000000	13580.000000	13580.000000
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242	1.610075	558.416127	151.967650	1964.684217	-37.809203	144.995216	7454.417378
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712	0.962634	3990.669241	541.014538	37.273762	0.079260	0.103916	4378.581772
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1196.000000	-38.182550	144.431810	249.000000
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000	1.000000	177.000000	93.000000	1940.000000	-37.856822	144.929600	4380.000000
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000	2.000000	440.000000	126.000000	1970.000000	-37.802355	145.000100	6555.000000
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000	2.000000	651.000000	174.000000	1999.000000	-37.756400	145.058305	10331.000000
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000	10.000000	433014.000000	44515.000000	2018.000000	-37.408530	145.526350	21650.000000

Description of data used

```
250746.88766946417 0.616398
```

MSE AND F SCORE OF DECISION TREE

185165.77777777777 0.73846

MSE AND F SCORE OF RANDOM FOREST

30479.283408642371 0.214546

MSE AND F SCORE OF NAIVE BAYES

AS WE CAN SEE RANDOM FOREST IS THE MOST EFFICIENT ALGORITHM FOR THE GIVEN DATA SET FOLLOWED BY DECISION TREE AND NAIVE BAYES RESPECTIVELY IN BOTH CATEGORIES i.e MSE(MEAN SQUARE ERROR) AND FSCORE

EXPERIMENT-8

AIM: Installation of Big data technologies and building a Hadoop cluster.

Description:

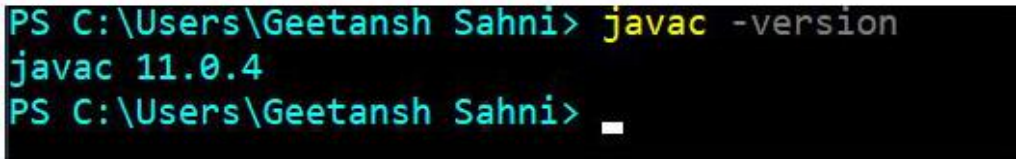
Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

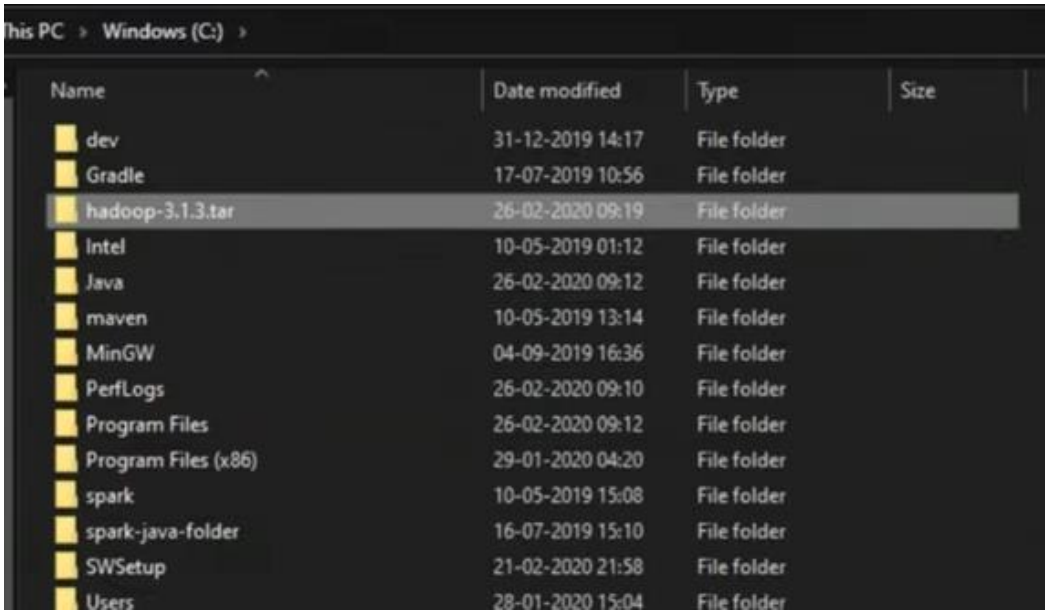
- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
- Hadoop Common – Provides common Java libraries that can be used across all modules.

PROCEDURE:

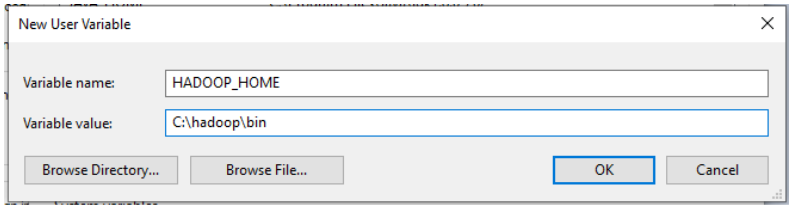
Step 1: Verify the Java installed
javac -version



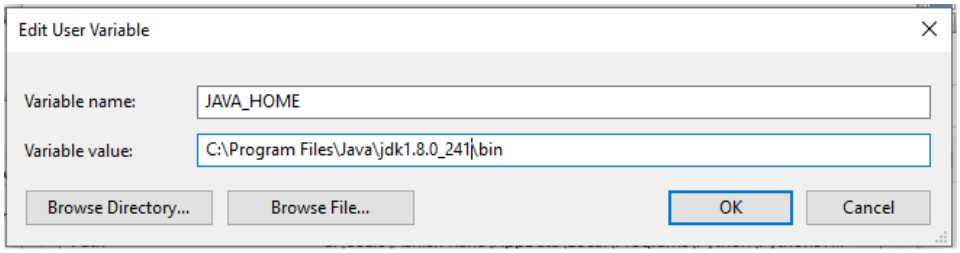
Step 2: Extract Hadoop at C:\Hadoop



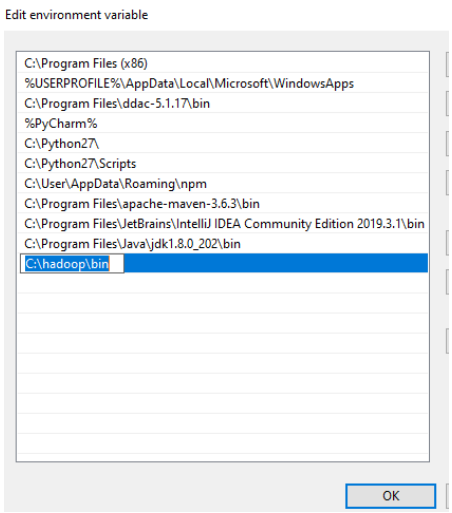
Step 3: Setting up the HADOOP_HOME variable
Use windows environment variable setting for Hadoop Path setting.



Step 4: Set JAVA_HOME variable
Use windows environment variable setting for Hadoop Path setting.



Step 5: Set Hadoop and Java bin directory path



Step 6: Hadoop Configuration :
For Hadoop Configuration we need to modify Six files that are listed below-

1. Core-site.xml

2. Mapred-site.xml

3. Hdfs-site.xml

4. Yarn-site.xml

5. Hadoop-env.cmd

6. Create two folders datanode and namenode

Step 6.1: Core-site.xml configuration

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Step 6.2: Mapred-site.xml configuration

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Step 6.3: Hdfs-site.xml configuration

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>C:\hadoop-2.8.0\data\namenode</value>
  </property>
  <property>
```

```
    <name>dfs.datanode.data.dir</name>
    <value>C:\hadoop-2.8.0\data\datanode</value>
  </property>
</configuration>
```

Step 6.4: Yarn-site.xml configuration

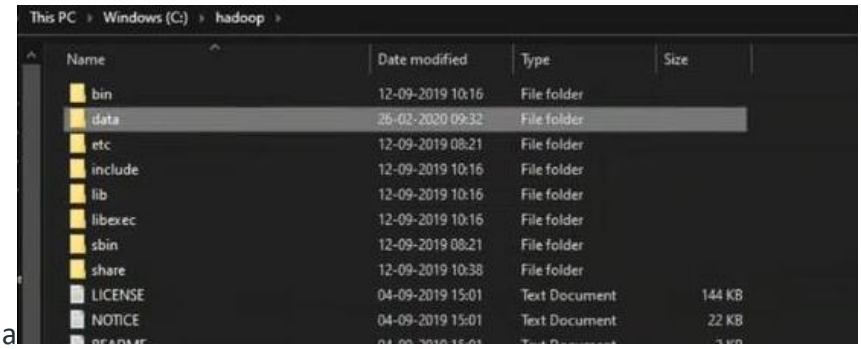
```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property><name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
   <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

Step 6.5: Hadoop-env.cmd configuration
Set "JAVA_HOME=C:\Java" (On C:\java this is path to file jdk.18.0)

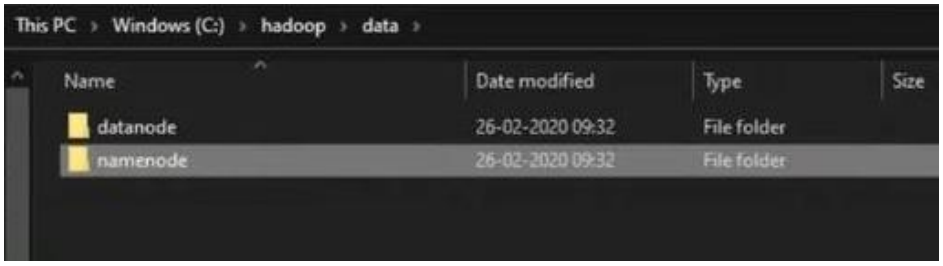
```
@rem Set Hadoop-specific environment variables here.

@rem The only required environment variable is JAVA_HOME.  All others are
@rem optional.  When running a distributed configuration it is best to
@rem set JAVA_HOME in this file, so that it is correctly defined on
@rem remote nodes.

@rem The java implementation to use.  Required.
set JAVA_HOME=%JAVA_HOME%
```

Step 6.6: Create datanode and namenode folders
1. Create folder "data" under "C:\Hadoop-2.8.0"

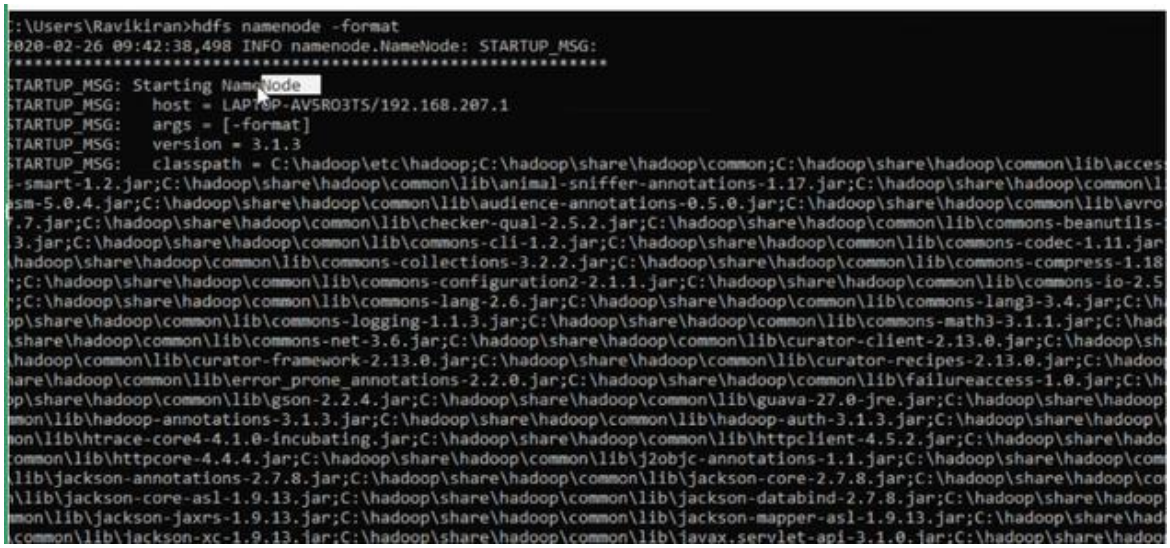
2. Create folder "datanode" under "C:\Hadoop-2.8.0\data"



3. Create folder "namenode" under "C:\Hadoop-2.8.0\data"



Step 7: Format the namenode folder
Open command window (cmd) and typing command “hdfs namenode –format”



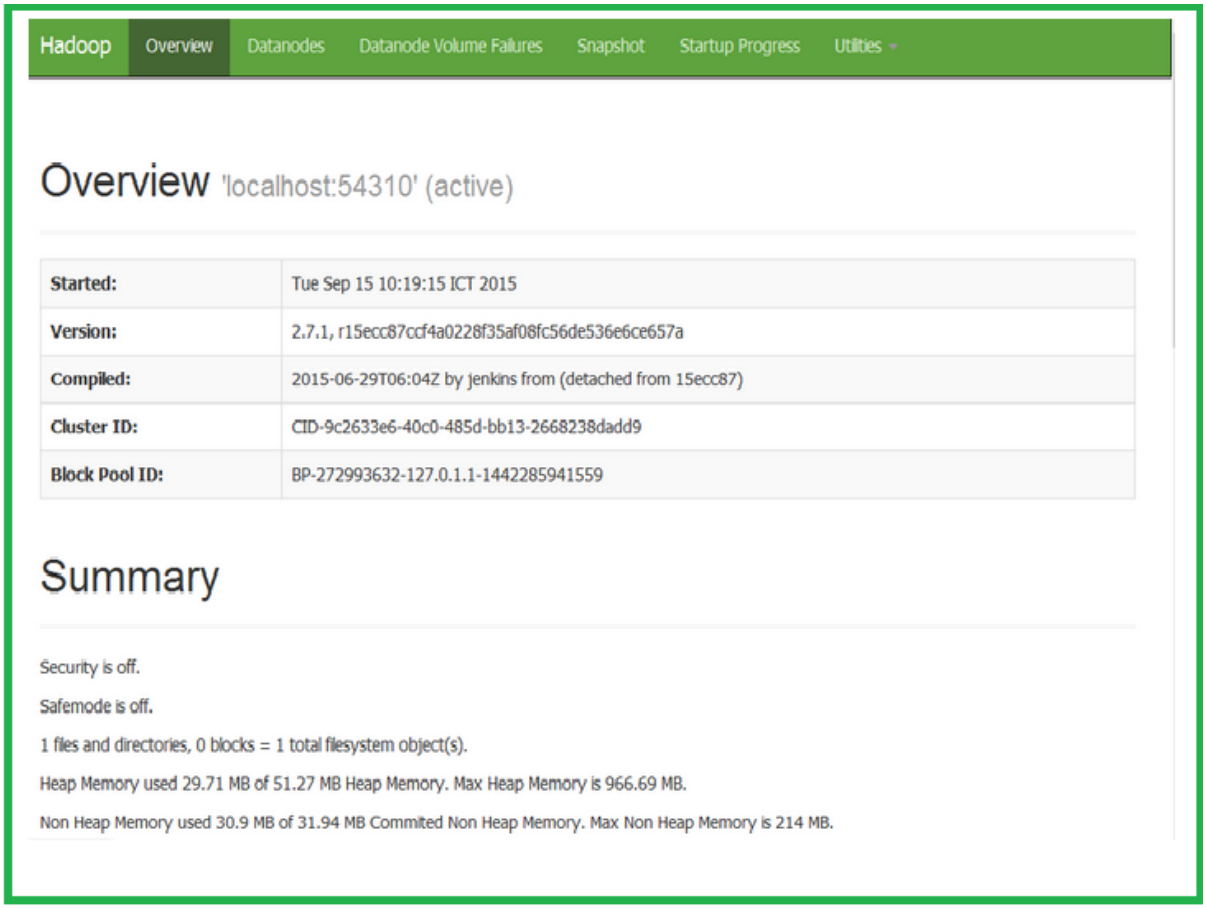
Step 8: Testing the setup
Open command window (cmd) and typing command “start-all.cmd”

Step 8.1: Testing the setup:
Ensure that namenode, datanode, and Resource manager are running

Step 9: Open: http://localhost:8088

Step 10:

Open: http://localhost:50070



ANALYSIS:

Here, we have successfully installed the Hadoop.

160119733143

Snigdha Reddy

DATA SCIENCE AND BIG DATA ANALYTICS

EXPERIMENT-11: Demonstration of Map Reduce concept.

Aim: To demonstrate the concept of map reduce.

Survey:

When the MapReduce framework was not there, parallel and distributed processing used to happen in a traditional way. For example,I have a weather log containing the daily average temperature of the years from 2000 to 2015. Here, I want to calculate the day having the highest temperature in each year.

So, just like in the traditional way, I will split the data into smaller parts or blocks and store them in different machines. Then, I will find the highest temperature in each part stored in the corresponding machine. At last, I will combine the results received from each of the machines to have the final output. Let us look at the challenges associated with this traditional approach:

1. **Critical path problem:** It is the amount of time taken to finish the job without delaying the next milestone or actual completion date. So, if, any of the machines delay the job, the whole work gets delayed.
2. **Reliability problem:** What if, any of the machines which are working with a part of data fails? The management of this failover becomes a challenge.
3. **Equal split issue:** How will I divide the data into smaller chunks so that each machine gets even part of data to work with. In other words, how to equally divide the data such that no individual machine is overloaded or underutilized.
4. **The single split may fail:** If any of the machines fail to provide the output, I will not be able to calculate the result. So, there should be a mechanism to ensure this fault tolerance capability of the system.
5. **Aggregation of the result:** There should be a mechanism to aggregate the result generated by each of the machines to produce the final output.

These are the issues which I will have to take care individually while performing parallel processing of huge data sets when using traditional approaches.

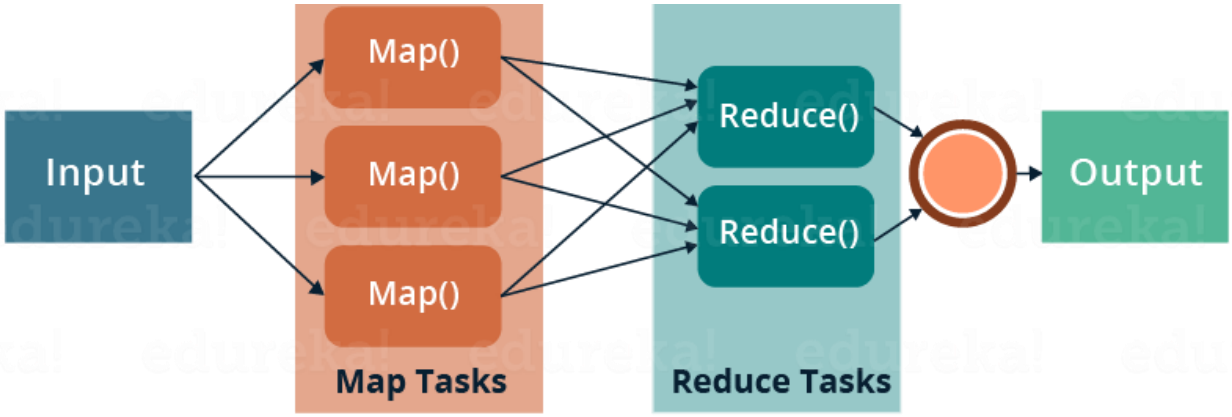
To overcome these issues, we have the MapReduce framework which allows us to perform such parallel computations without bothering about the issues like reliability, fault tolerance etc. Therefore, MapReduce gives you the flexibility to write code logic without caring about the design issues of the system.

MAPREDUCE:

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.

- MapReduce consists of two distinct tasks – Map and Reduce.

- As the name MapReduce suggests, the reducer phase takes place after the mapper phase has been completed.
- So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
- The output of a Mapper or map job (key-value pairs) is input to the Reducer.
- The reducer receives the key-value pair from multiple map jobs.
- Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.



MapReduce majorly has the following three Classes. They are,

Mapper Class

The first stage in Data Processing using MapReduce is the **Mapper Class.** Here, RecordReader processes each Input record and generates the respective key-value pair. Hadoop's Mapper store saves this intermediate data into the local disk.

- **Input Split**

It is the logical representation of data. It represents a block of work that contains a single map task in the MapReduce Program.

- **RecordReader**

It interacts with the Input split and converts the obtained data in the form of **Key-Value** Pairs.

Reducer Class

The Intermediate output generated from the mapper is fed to the reducer which processes it and generates the final output which is then saved in the **HDFS.**

Driver Class

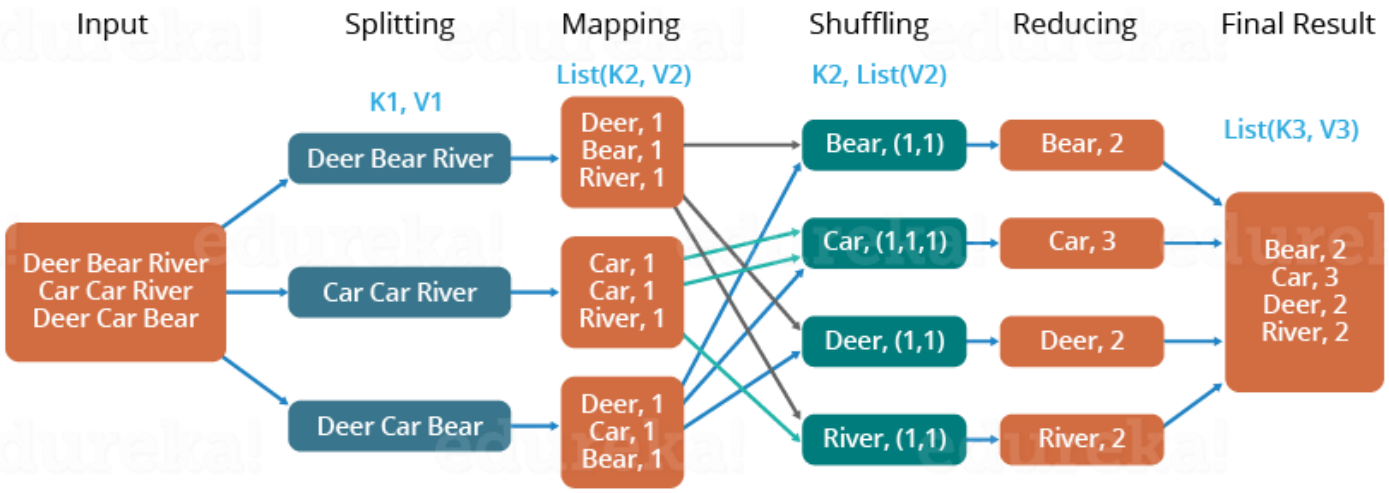
The major component in a MapReduce job is a **Driver Class.** It is responsible for setting up a MapReduce Job to run-in Hadoop. We specify the names of **Mapper** and **Reducer** Classes long with data types and their respective job names.

A Word Count Example of MapReduce

For example,I have a text file called example.txt whose contents are as follows:

Dear, Bear, River, Car, Car, River, Deer, Car and Bear

Now, suppose, we have to perform a word count on the sample.txt using MapReduce. So, we will be finding the unique words and the number of occurrences of those unique words.



- First, we divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, we tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
- Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.
- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.
- Finally, all the output key/value pairs are then collected and written in the output file.

Explanation of MapReduce Program

The entire MapReduce program can be fundamentally divided into three parts:

- Mapper Phase Code
- Reducer Phase Code
- Driver Code

Mapper code:

```
public static class Map extends Mapper&lt;LongWritable,Text,Text,IntWritable&gt; {
    public void map(LongWritable key, Text value, Context context) throws
IOException,InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            value.set(tokenizer.nextToken());
            context.write(value, new IntWritable(1));
        }
```

- We have created a class Map that extends the class Mapper which is already defined in the MapReduce Framework.
- We define the data types of input and output key/value pair after the class declaration using angle brackets.
- Both the input and output of the Mapper is a key/value pair.
- Input:
  - The *key* is nothing but the offset of each line in the text file: *LongWritable*
  - The *value* is each individual line (as shown in the figure at the right): *Text*
- Output:
  - The *key* is the tokenized words: *Text*
  - We have the hardcoded *value* in our case which is 1: *IntWritable*
  - Example – Dear 1, Bear 1, etc.
- We have written a java code where we have tokenized each word and assigned them a hardcoded value equal to *1*.

Reducer Code:

```
public static class Reduce extends Reducer&lt;Text,IntWritable,Text,IntWritable&gt; {
    public void reduce(Text key, Iterable&lt;IntWritable&gt; values,Context context)
        throws IOException,InterruptedException {
            int sum=0;
            for(IntWritable x: values)
            {
                 sum+=x.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
```

- We have created a class Reduce which extends class Reducer like that of Mapper.
- We define the data types of input and output key/value pair after the class declaration using angle brackets as done for Mapper.
- Both the input and the output of the Reducer is a key-value pair.
- Input:
  - The *key* nothing but those unique words which have been generated after the sorting and shuffling phase: *Text*
  - The *value* is a list of integers corresponding to each key: *IntWritable*
  - Example – Bear, [1, 1], etc.
- Output:
  - The *key* is all the unique words present in the input text file: *Text*
  - The *value* is the number of occurrences of each of the unique words: *IntWritable*
  - Example – Bear, 2; Car, 3, etc.
- We have aggregated the values present in each of the list corresponding to each key and produced the final answer.
- In general, a single reducer is created for each of the unique words, but, you can specify the number of reducer in mapred-site.xml.

Driver Code:

```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
```

```
job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

- In the driver class, we set the configuration of our MapReduce job to run in Hadoop.
- We specify the name of the job, the data type of input/output of the mapper and reducer.
- We also specify the names of the mapper and reducer classes.
- The path of the input and output folder is also specified.
- The method setInputFormatClass () is used for specifying how a Mapper will read the input data or what will be the unit of work. Here, we have chosen TextInputFormat so that a single line is read by the mapper at a time from the input text file.
- The main () method is the entry point for the driver. In this method, we instantiate a new Configuration object for the job.

Advantages of MapReduce

The two biggest advantages of MapReduce are:

1. Parallel Processing:

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount as shown.

2. Data Locality:

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework. In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

- Moving huge data to processing is costly and deteriorates the network performance.
- Processing takes time as the data is processed by a single unit which becomes the bottleneck.
- The master node can get over-burdened and may fail.

Now, MapReduce allows us to overcome the above issues by bringing the processing unit to the data. So, as you can see in the above image that the data is distributed among multiple nodes where each node processes the part of the data residing on it. This allows us to have the following advantages:

- It is very cost-effective to move processing unit to the data.
- The processing time is reduced as all the nodes are working with their part of the data in parallel.
- Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

160119733143
Snigdha Reddy

DATA SCIENCE AND BIG DATAANALYTICS LAB

AIM: Experiment for data loading from local machine to Hadoop.

DESCRIPTION:

Initially you have to format the configured HDFS file system, open namenode (HDFS server), and execute the following command.

$ hadoop namenode -format

After formatting the HDFS, start the distributed file system. The following command will start the namenode as well as the data nodes as cluster.

$ start-dfs.sh

Listing Files in HDFS

After loading the information in the server, we can find the list of files in a directory, status of a file, using 'ls'. Given below is the syntax of ls that you can pass to a directory or a filename as an argument.

$ $HADOOP_HOME/bin/hadoop fs -ls <args>

Inserting Data into HDFS

Assume we have data in the file called file.txt in the local system which is ought to be saved in the hdfs file system. Follow the steps given below to insert the required file in the Hadoop file system.

Step 1

You have to create an input directory.

$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input

Step 2

Transfer and store a data file from local systems to the Hadoop file system using the put command.

$ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input

Step 3

You can verify the file using ls command.

$ $HADOOP_HOME/bin/hadoop fs -ls /user/input

Retrieving Data from HDFS

Assume we have a file in HDFS called outfile. Given below is a simple demonstration for retrieving the required file from the Hadoop file system.

Step 1

Initially, view the data from HDFS using cat command.

$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile

Step 2

Get the file from HDFS to the local file system using get command.

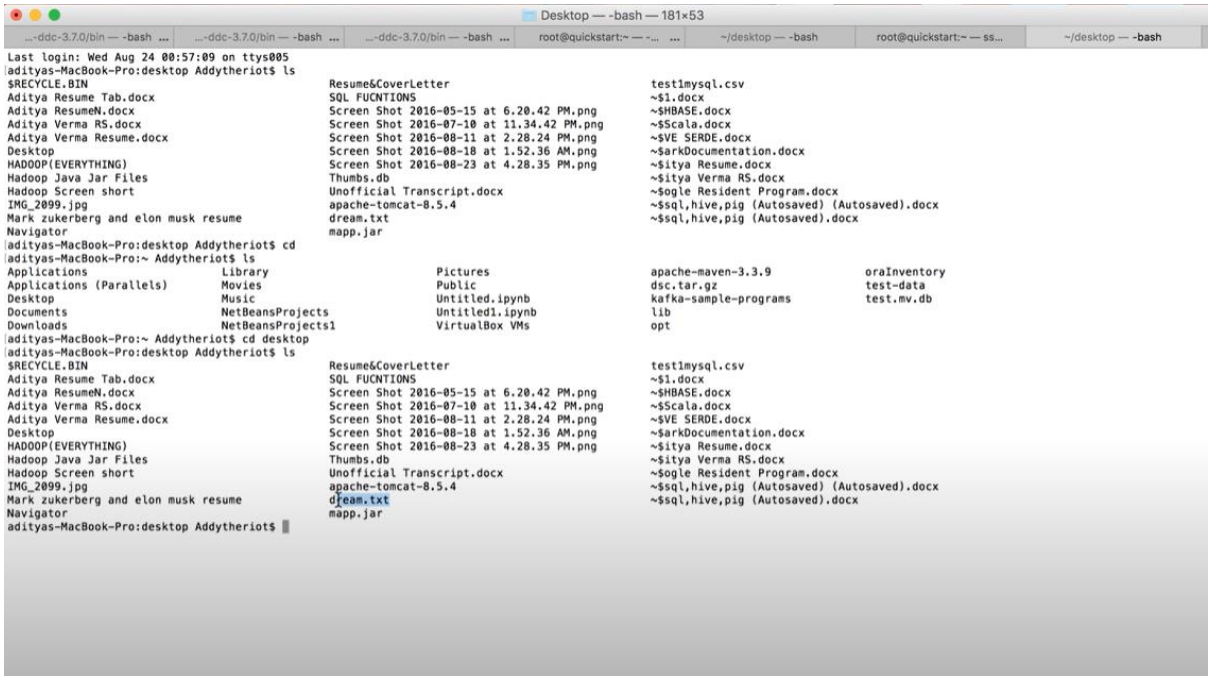
$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/

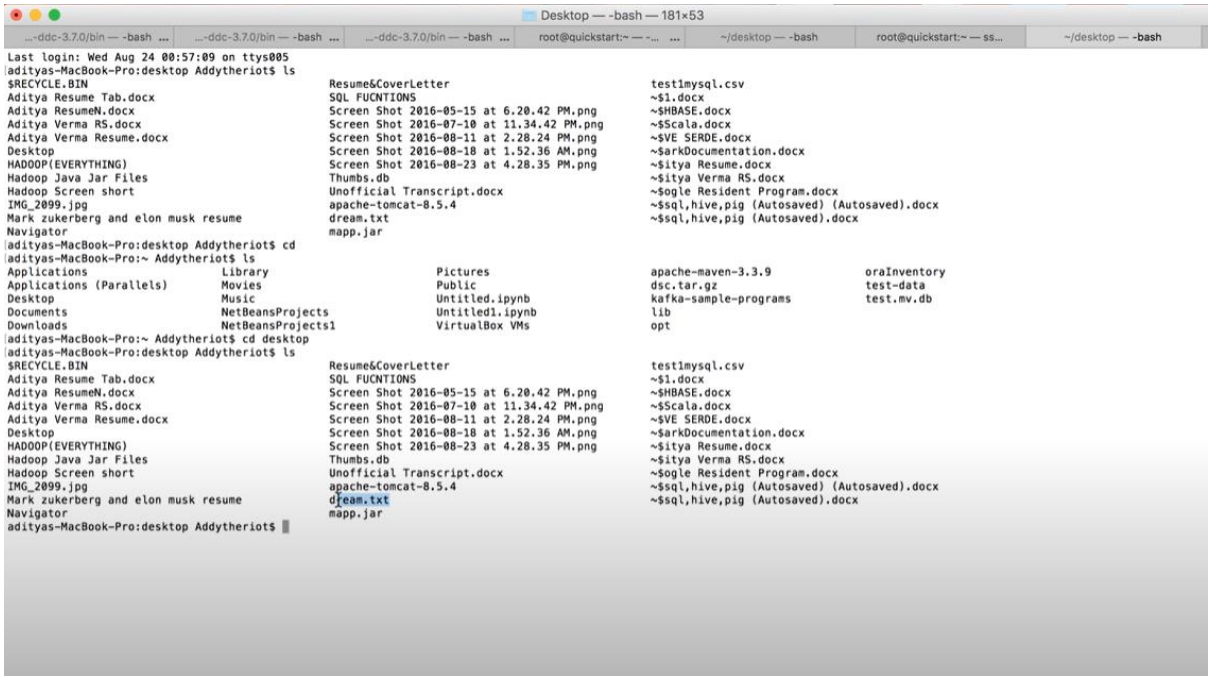
Shutting Down the HDFS

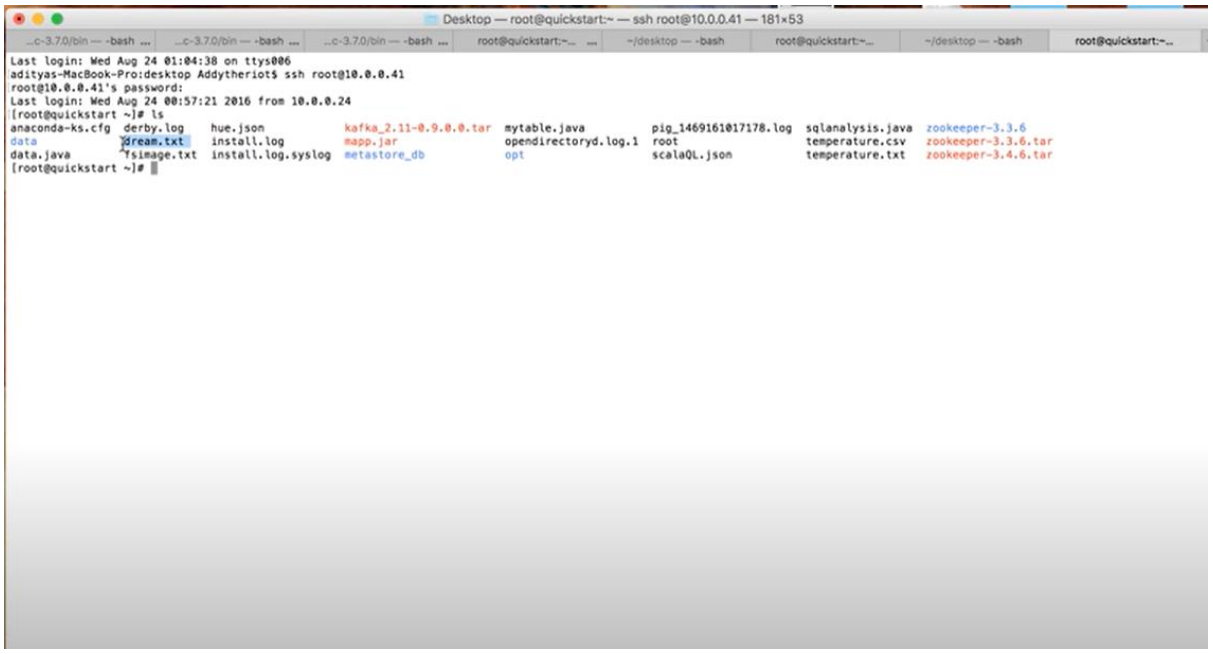
You can shut down the HDFS by using the following command.

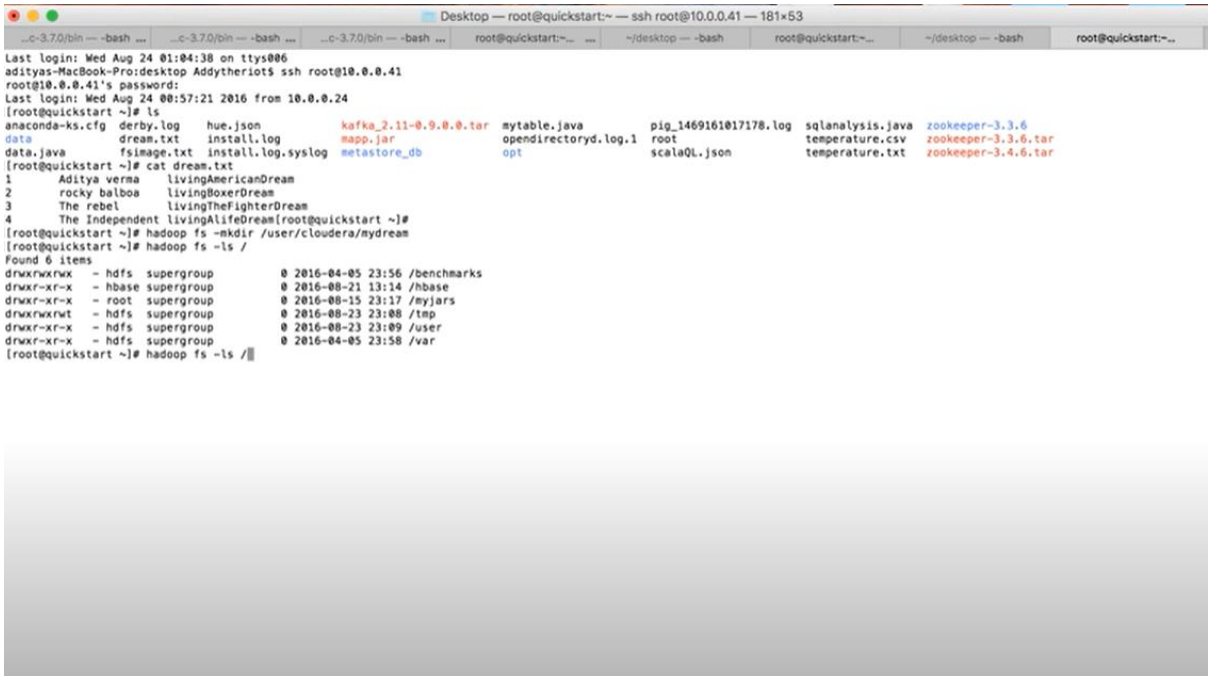
$ stop-dfs.sh

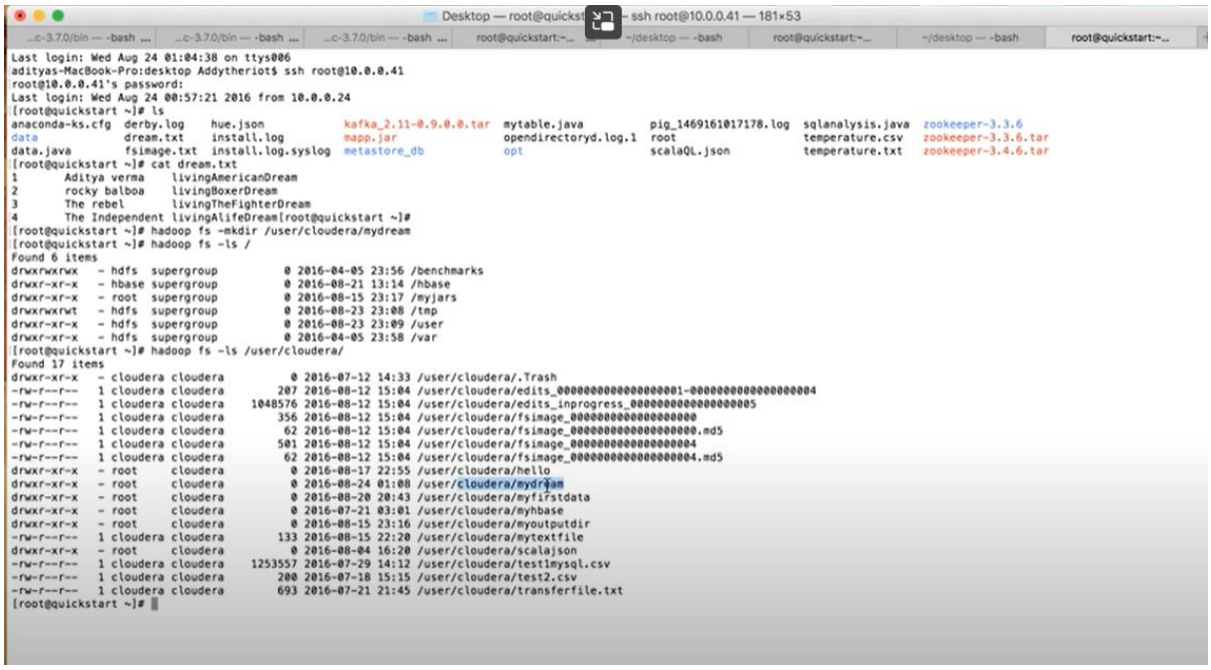
PROCEDURE:

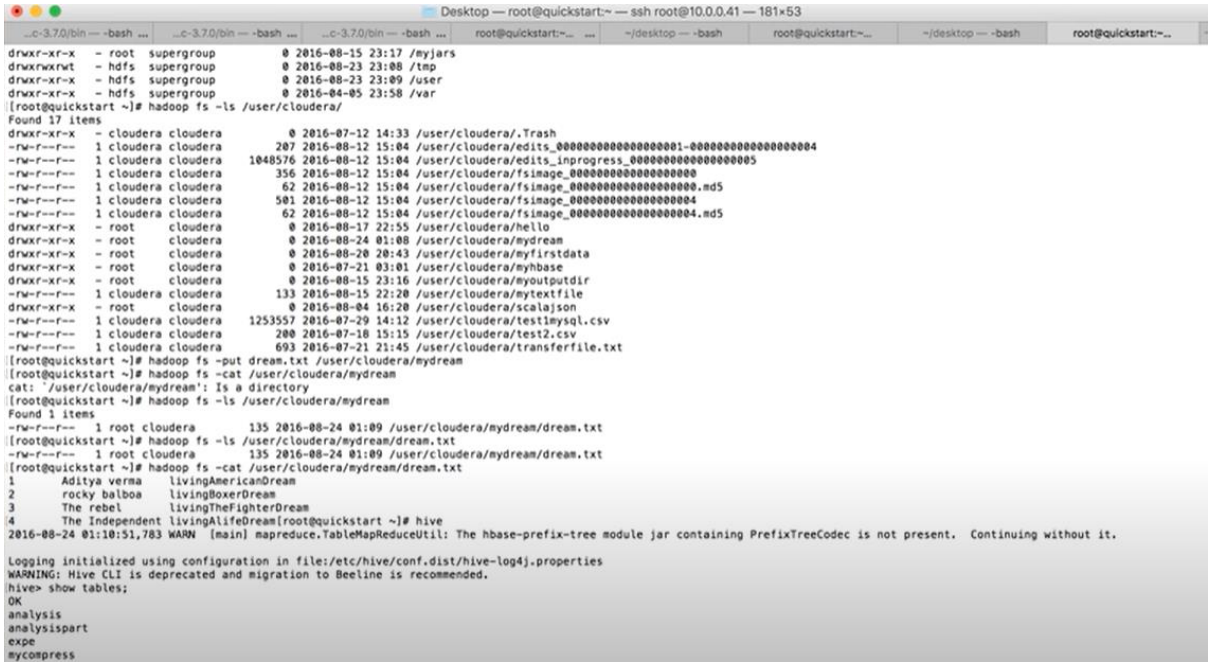


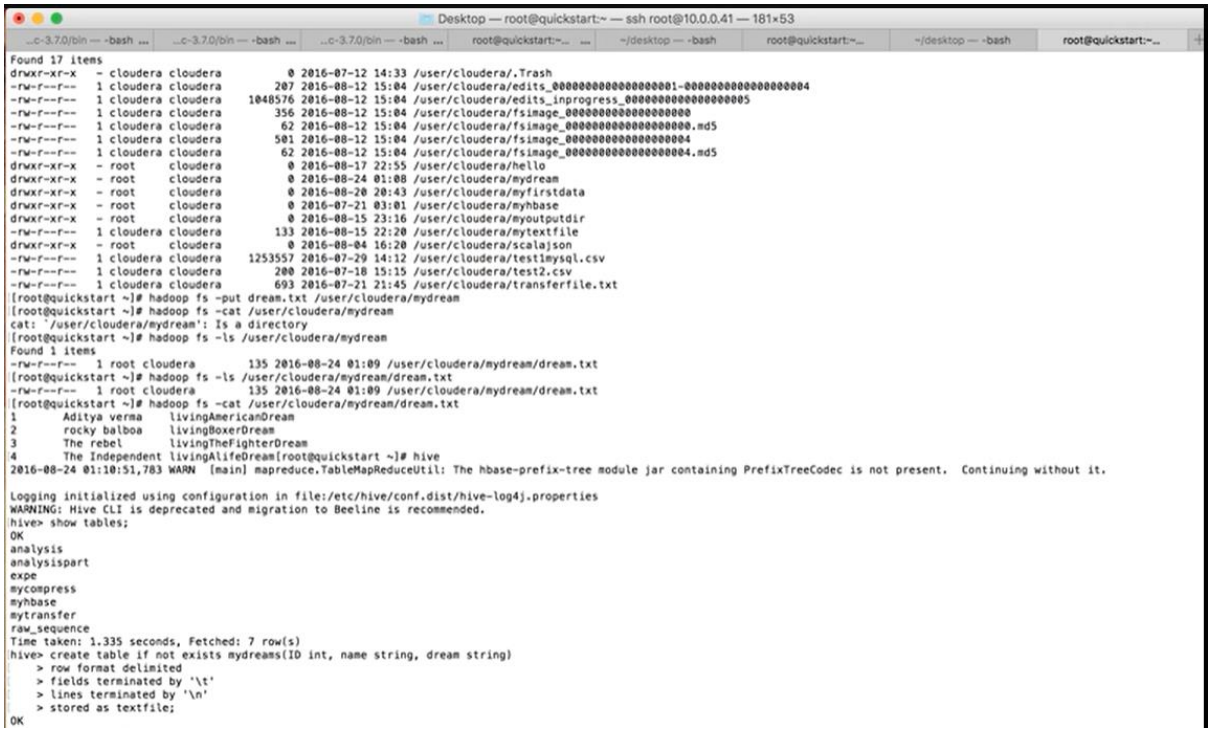

Desktop — -bash — 181×53
Last login: Wed Aug 24 00:57:09 on ttys005
adityas-MacBook-Pro:desktop Addytheriot$ ls
$RECYCLE.BIN
Aditya Resume Tab.docx
Aditya ResumeN.docx
Aditya Verma RS.docx
Aditya Verma Resume.docx
Desktop
HADOOP(EVERYTHING)
Hadoop Java Jar Files
Hadoop Screen short
IMG_2099.jpg
Mark zukerberg and elon musk resume
Navigator
Resume&CoverLetter
SQL FUCNTIONS
Screen Shot 2016-05-15 at 6.20.42 PM.png
Screen Shot 2016-07-10 at 11.34.42 PM.png
Screen Shot 2016-08-11 at 2.28.24 PM.png
Screen Shot 2016-08-18 at 1.52.36 AM.png
Screen Shot 2016-08-23 at 4.28.35 PM.png
Thumbs.db
Unofficial Transcript.docx
apache-tomcat-8.5.4
dream.txt
mapp.jar
test1mysql.csv
~$1.docx
~$HBASE.docx
~$Scala.docx
~$VE SERDE.docx
~$arkDocumentation.docx
~$itya Resume.docx
~$itya Verma RS.docx
~$ogle Resident Program.docx
~$sql,hive,pig (Autosaved) (Autosaved).docx
~$sql,hive,pig (Autosaved).docx
adityas-MacBook-Pro:desktop Addytheriot$ cd
adityas-MacBook-Pro:~ Addytheriot$ ls
Applications
Applications (Parallels)
Desktop
Documents
Downloads
Library
Movies
Music
NetBeansProjects
NetBeansProjects1
Pictures
Public
Untitled.ipynb
Untitled1.ipynb
VirtualBox VMs
apache-maven-3.3.9
dsc.tar.gz
kafka-sample-programs
lib
opt
oraInventory
test-data
test.mv.db
adityas-MacBook-Pro:~ Addytheriot$ cd desktop
adityas-MacBook-Pro:desktop Addytheriot$ ls
adityas-MacBook-Pro:desktop Addytheriot$

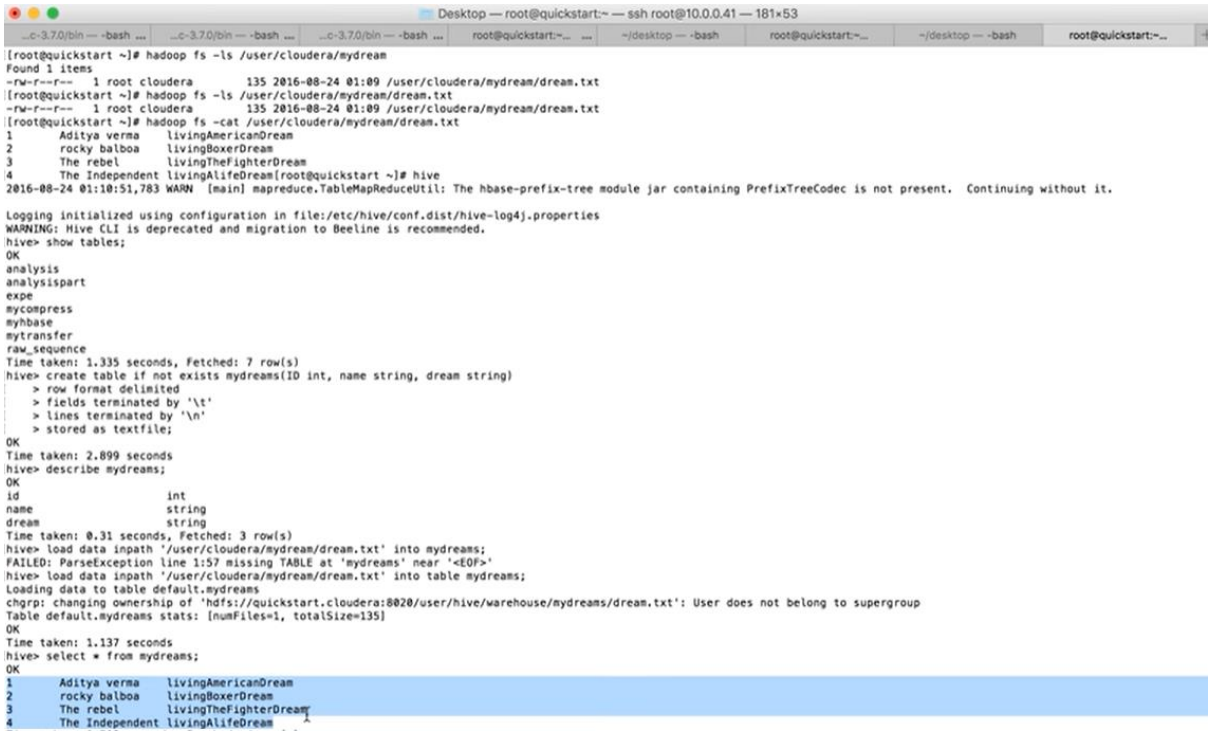

Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
Last login: Wed Aug 24 01:04:38 on ttys006
adityas-MacBook-Pro:desktop Addytheriot$ ssh root@10.0.0.41
root@10.0.0.41's password:
Last login: Wed Aug 24 00:57:21 2016 from 10.0.0.24
[root@quickstart ~]# ls
anaconda-ks.cfg
data
data.java
derby.log
dream.txt
fsimage.txt
hue.json
install.log
install.log.syslog
kafka_2.11-0.9.0.0.tar
mapp.jar
metastore_db
mytable.java
opendirectoryd.log.1
opt
pig_1469161017178.log
root
scalaQL.json
sqlanalysis.java
temperature.csv
temperature.txt
zookeeper-3.3.6
zookeeper-3.3.6.tar
zookeeper-3.4.6.tar
[root@quickstart ~]#

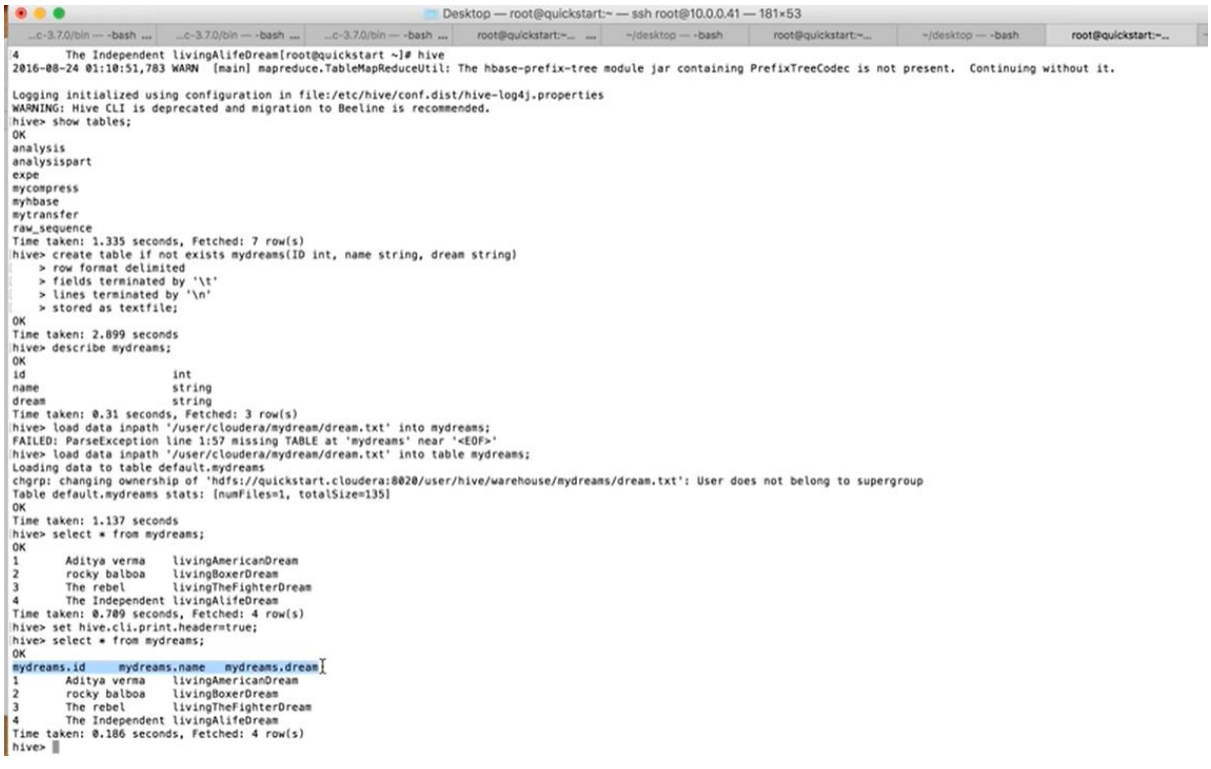

Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
Last login: Wed Aug 24 01:04:38 on ttys006
adityas-MacBook-Pro:desktop Addytheriot$ ssh root@10.0.0.41
root@10.0.0.41's password:
Last login: Wed Aug 24 00:57:21 2016 from 10.0.0.24
[root@quickstart ~]# ls
anaconda-ks.cfg derby.log hue.json kafka_2.11-0.9.0.0.tar mytable.java pig_1469161017178.log sqlanalysis.java zookeeper-3.3.6
data dream.txt install.log mapp.jar opendirectoryd.log.1 root temperature.csv zookeeper-3.3.6.tar
data.java fsimage.txt install.log.syslog metastore_db opt scalaQL.json temperature.txt zookeeper-3.4.6.tar
[root@quickstart ~]# cat dream.txt
1 Aditya verma livingAmericanDream
2 rocky balboa livingBoxerDream
3 The rebel livingTheFighterDream
4 The Independent livingAlifeDream[root@quickstart ~]#
[root@quickstart ~]# hadoop fs -mkdir /user/cloudera/mydream
[root@quickstart ~]# hadoop fs -ls /
Found 6 items
drwxrwxrwx - hdfs supergroup 0 2016-04-05 23:56 /benchmarks
drwxr-xr-x - hbase supergroup 0 2016-08-21 13:14 /hbase
drwxr-xr-x - root supergroup 0 2016-08-15 23:17 /myjars
drwxrwxrwt - hdfs supergroup 0 2016-08-23 23:08 /tmp
drwxr-xr-x - hdfs supergroup 0 2016-08-23 23:09 /user
drwxr-xr-x - hdfs supergroup 0 2016-04-05 23:58 /var
[root@quickstart ~]# hadoop fs -ls /


Last login: Wed Aug 24 01:04:38 on ttys006
adityas-MacBook-Pro:desktop Addytheriot$ ssh root@10.0.0.41
root@10.0.0.41's password:
Last login: Wed Aug 24 00:57:21 2016 from 10.0.0.24
[root@quickstart ~]# ls
anaconda-ks.cfg derby.log hue.json kafka_2.11-0.9.0.0.tar mytable.java pig_1469161017178.log sqlanalysis.java zookeeper-3.3.6
data dream.txt install.log mapp.jar opendirectoryd.log.1 root temperature.csv zookeeper-3.3.6.tar
data.java fsimage.txt install.log.syslog metastore_db opt scalaQL.json temperature.txt zookeeper-3.4.6.tar
[root@quickstart ~]# cat dream.txt
1 Aditya verma livingAmericanDream
2 rocky balboa livingBoxerDream
3 The rebel livingTheFighterDream
4 The Independent livingAlifeDream[root@quickstart ~]#
[root@quickstart ~]# hadoop fs -mkdir /user/cloudera/mydream
[root@quickstart ~]# hadoop fs -ls /
Found 6 items
drwxrwxrwx - hdfs supergroup 0 2016-04-05 23:56 /benchmarks
drwxr-xr-x - hbase supergroup 0 2016-08-21 13:14 /hbase
drwxr-xr-x - root supergroup 0 2016-08-15 23:17 /myjars
drwxrwxrwt - hdfs supergroup 0 2016-08-23 23:08 /tmp
drwxr-xr-x - hdfs supergroup 0 2016-08-23 23:09 /user
drwxr-xr-x - hdfs supergroup 0 2016-04-05 23:58 /var
[root@quickstart ~]# hadoop fs -ls /user/cloudera/
Found 17 items
drwxr-xr-x - cloudera cloudera 0 2016-07-12 14:33 /user/cloudera/.Trash
-rw-r--r-- 1 cloudera cloudera 207 2016-08-12 15:04 /user/cloudera/edits_0000000000000000001-0000000000000000004
-rw-r--r-- 1 cloudera cloudera 1048576 2016-08-12 15:04 /user/cloudera/edits_inprogress_0000000000000000005
-rw-r--r-- 1 cloudera cloudera 356 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000
-rw-r--r-- 1 cloudera cloudera 62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000.md5
-rw-r--r-- 1 cloudera cloudera 501 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004
-rw-r--r-- 1 cloudera cloudera 62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004.md5
drwxr-xr-x - root cloudera 0 2016-08-17 22:55 /user/cloudera/hello
drwxr-xr-x - root cloudera 0 2016-08-24 01:08 /user/cloudera/mydream
drwxr-xr-x - root cloudera 0 2016-08-20 20:43 /user/cloudera/myfirstdata
drwxr-xr-x - root cloudera 0 2016-07-21 03:01 /user/cloudera/myhbase
drwxr-xr-x - root cloudera 0 2016-08-15 23:16 /user/cloudera/myoutputdir
-rw-r--r-- 1 cloudera cloudera 133 2016-08-15 22:20 /user/cloudera/mytextfile
drwxr-xr-x - root cloudera 0 2016-08-04 16:20 /user/cloudera/scalajson
-rw-r--r-- 1 cloudera cloudera 1253557 2016-07-29 14:12 /user/cloudera/test1mysql.csv
-rw-r--r-- 1 cloudera cloudera 200 2016-07-18 15:15 /user/cloudera/test2.csv
-rw-r--r-- 1 cloudera cloudera 693 2016-07-21 21:45 /user/cloudera/transferfile.txt
[root@quickstart ~]#


Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
drwxr-xr-x   - root supergroup   0 2016-08-15 23:17 /myjars
drwxrwxrwt   - hdfs supergroup   0 2016-08-23 23:08 /tmp
drwxr-xr-x   - hdfs supergroup   0 2016-08-23 23:09 /user
drwxr-xr-x   - hdfs supergroup   0 2016-04-05 23:58 /var
[root@quickstart ~]# hadoop fs -ls /user/cloudera/
Found 17 items
drwxr-xr-x   - cloudera cloudera   0 2016-07-12 14:33 /user/cloudera/.Trash
-rw-r--r--   1 cloudera cloudera   207 2016-08-12 15:04 /user/cloudera/edits_0000000000000000001-0000000000000000004
-rw-r--r--   1 cloudera cloudera   1048576 2016-08-12 15:04 /user/cloudera/edits_inprogress_0000000000000000005
-rw-r--r--   1 cloudera cloudera   356 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000
-rw-r--r--   1 cloudera cloudera   62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000.md5
-rw-r--r--   1 cloudera cloudera   501 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004
-rw-r--r--   1 cloudera cloudera   62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004.md5
drwxr-xr-x   - root   cloudera   0 2016-08-17 22:55 /user/cloudera/hello
drwxr-xr-x   - root   cloudera   0 2016-08-24 01:08 /user/cloudera/mydream
drwxr-xr-x   - root   cloudera   0 2016-08-20 20:43 /user/cloudera/myfirstdata
drwxr-xr-x   - root   cloudera   0 2016-07-21 03:01 /user/cloudera/myhbase
drwxr-xr-x   - root   cloudera   0 2016-08-15 23:16 /user/cloudera/myoutputdir
-rw-r--r--   1 cloudera cloudera   133 2016-08-15 22:20 /user/cloudera/mytextfile
drwxr-xr-x   - root   cloudera   0 2016-08-04 16:20 /user/cloudera/scalajson
-rw-r--r--   1 cloudera cloudera   1253557 2016-07-29 14:12 /user/cloudera/test1mysql.csv
-rw-r--r--   1 cloudera cloudera   200 2016-07-18 15:15 /user/cloudera/test2.csv
-rw-r--r--   1 cloudera cloudera   693 2016-07-21 21:45 /user/cloudera/transferfile.txt
[root@quickstart ~]# hadoop fs -put dream.txt /user/cloudera/mydream
[root@quickstart ~]# hadoop fs -cat /user/cloudera/mydream
cat: `/user/cloudera/mydream': Is a directory
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream
Found 1 items
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream/dream.txt
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -cat /user/cloudera/mydream/dream.txt
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream[root@quickstart ~]# hive
2016-08-24 01:10:51,783 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show tables;
OK
analysis
analysispart
expe
mycompress


Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
Found 17 items
drwxr-xr-x   - cloudera cloudera   0 2016-07-12 14:33 /user/cloudera/.Trash
-rw-r--r--   1 cloudera cloudera   207 2016-08-12 15:04 /user/cloudera/edits_0000000000000000001-0000000000000000004
-rw-r--r--   1 cloudera cloudera   1048576 2016-08-12 15:04 /user/cloudera/edits_inprogress_0000000000000000005
-rw-r--r--   1 cloudera cloudera   356 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000
-rw-r--r--   1 cloudera cloudera   62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000000.md5
-rw-r--r--   1 cloudera cloudera   501 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004
-rw-r--r--   1 cloudera cloudera   62 2016-08-12 15:04 /user/cloudera/fsimage_0000000000000000004.md5
drwxr-xr-x   - root   cloudera   0 2016-08-17 22:55 /user/cloudera/hello
drwxr-xr-x   - root   cloudera   0 2016-08-24 01:08 /user/cloudera/mydream
drwxr-xr-x   - root   cloudera   0 2016-08-20 20:43 /user/cloudera/myfirstdata
drwxr-xr-x   - root   cloudera   0 2016-07-21 03:01 /user/cloudera/myhbase
drwxr-xr-x   - root   cloudera   0 2016-08-15 23:16 /user/cloudera/myoutputdir
-rw-r--r--   1 cloudera cloudera   133 2016-08-15 22:20 /user/cloudera/mytextfile
drwxr-xr-x   - root   cloudera   0 2016-08-04 16:20 /user/cloudera/scalajson
-rw-r--r--   1 cloudera cloudera   1253557 2016-07-29 14:12 /user/cloudera/test1mysql.csv
-rw-r--r--   1 cloudera cloudera   200 2016-07-18 15:15 /user/cloudera/test2.csv
-rw-r--r--   1 cloudera cloudera   693 2016-07-21 21:45 /user/cloudera/transferfile.txt
[root@quickstart ~]# hadoop fs -put dream.txt /user/cloudera/mydream
[root@quickstart ~]# hadoop fs -cat /user/cloudera/mydream
cat: `/user/cloudera/mydream': Is a directory
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream
Found 1 items
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream/dream.txt
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -cat /user/cloudera/mydream/dream.txt
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream[root@quickstart ~]# hive
2016-08-24 01:10:51,783 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show tables;
OK
analysis
analysispart
expe
mycompress
myhbase
mytransfer
raw_sequence
Time taken: 1.335 seconds, Fetched: 7 row(s)
hive> create table if not exists mydreams(ID int, name string, dream string)
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK


Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream
Found 1 items
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -ls /user/cloudera/mydream/dream.txt
-rw-r--r--   1 root cloudera   135 2016-08-24 01:09 /user/cloudera/mydream/dream.txt
[root@quickstart ~]# hadoop fs -cat /user/cloudera/mydream/dream.txt
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream[root@quickstart ~]# hive
2016-08-24 01:10:51,783 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show tables;
OK
analysis
analysispart
expe
mycompress
myhbase
mytransfer
raw_sequence
Time taken: 1.335 seconds, Fetched: 7 row(s)
hive> create table if not exists mydreams(ID int, name string, dream string)
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 2.899 seconds
hive> describe mydreams;
OK
id   int
name   string
dream   string
Time taken: 0.31 seconds, Fetched: 3 row(s)
hive> load data inpath '/user/cloudera/mydream/dream.txt' into mydreams;
FAILED: ParseException line 1:57 missing TABLE at 'mydreams' near '<EOF>'
hive> load data inpath '/user/cloudera/mydream/dream.txt' into table mydreams;
Loading data to table default.mydreams
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/mydreams/dream.txt': User does not belong to supergroup
Table default.mydreams stats: [numFiles=1, totalSize=135]
OK
Time taken: 1.137 seconds
hive> select * from mydreams;
OK
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream


Desktop — root@quickstart:~ — ssh root@10.0.0.41 — 181×53
4   The Independent livingAlifeDream[root@quickstart ~]# hive
2016-08-24 01:10:51,783 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show tables;
OK
analysis
analysispart
expe
mycompress
myhbase
mytransfer
raw_sequence
Time taken: 1.335 seconds, Fetched: 7 row(s)
hive> create table if not exists mydreams(ID int, name string, dream string)
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 2.899 seconds
hive> describe mydreams;
OK
id   int
name   string
dream   string
Time taken: 0.31 seconds, Fetched: 3 row(s)
hive> load data inpath '/user/cloudera/mydream/dream.txt' into mydreams;
FAILED: ParseException line 1:57 missing TABLE at 'mydreams' near '<EOF>'
hive> load data inpath '/user/cloudera/mydream/dream.txt' into table mydreams;
Loading data to table default.mydreams
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/mydreams/dream.txt': User does not belong to supergroup
Table default.mydreams stats: [numFiles=1, totalSize=135]
OK
Time taken: 1.137 seconds
hive> select * from mydreams;
OK
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream
Time taken: 0.709 seconds, Fetched: 4 row(s)
hive> set hive.cli.print.header=true;
hive> select * from mydreams;
OK
mydreams.id   mydreams.name   mydreams.dream
1   Aditya verma   livingAmericanDream
2   rocky balboa   livingBoxerDream
3   The rebel   livingTheFighterDream
4   The Independent livingAlifeDream
Time taken: 0.186 seconds, Fetched: 4 row(s)
hive>

INTRODUCTION TO HADOOP

Prepared By : Manoj Kumar Joshi & Vikas Sawhney

General Agenda

Introduction to Hadoop Architecture

Acknowledgement

- Thanks to all the authors who left their self-explanatory images on the internet.
- Thanks to bradhedlund.com and Cloudera Inc for their blogs on Hadoop
- We own the errors of course

History Of Hadoop

- Hadoop was started by Doug Cutting to support two of his other well known projects, Lucene and Nutch
- Hadoop has been inspired by Google's File System (GFS) which was detailed in a paper by released by Google in 2003
- Hadoop, originally called Nutch Distributed File System (NDFS) split from Nutch in 2006 to become a sub-project of Lucene. At this point it was renamed to Hadoop.

History Of Hadoop (Contd.)

Hadoop Creation History

Doug Cutting adds DFS & MapReduce support to Nutch

Doug Cutting & Mike Cafarella started working on Nutch

Fastest sort of a TB, 3.5mins over 910 nodes

NY Times converts 4TB of image archives over 100 EC2s

Fastest sort of a TB, 62secs over 1,460 nodes
Sorted a PB in 16.25hours over 3,658 nodes

2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009

Google publishes GFS & MapReduce papers

Yahoo! hires Cutting, Hadoop spins out of Nutch

Facebooks launches Hive: SQL Support for Hadoop

cloudera Founded

Doug Cutting joins Cloudera

Hadoop Summit 2009, 750 attendees

Source: Cloudera, Inc.

Distributed File System (DFS)

- A **Distributed File System** (DFS) is simply a classical model of a file system distributed across multiple machines. The purpose is to promote sharing of dispersed files.
- This is an area of active research interest today.
- Clients should view a DFS the same way they would a centralized FS; the distribution is hidden at a lower level.
- A DFS provides high throughput data access and fault tolerance.

Why Hadoop ?

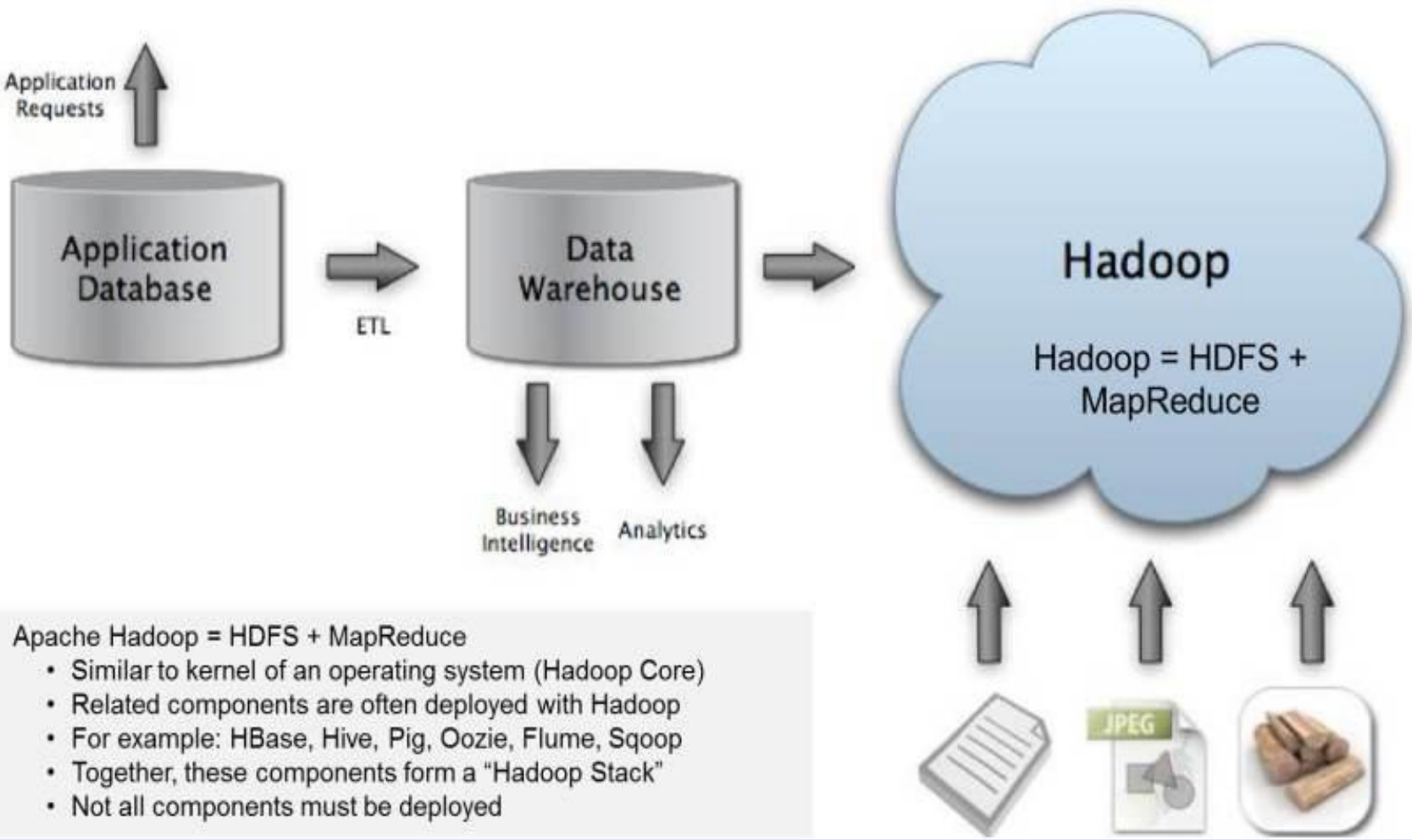
Hadoop infrastructure provides these capabilities

- Scalability
  - -Thousands of Compute Nodes
  - -Petabytes of data
- Cost effective
  - -Runs On Low Cost Commodity Hardware
- Efficient
  - -By distributing the data, Hadoop can process it in parallel on the nodes where the data is located.

What Is Hadoop ?

- Open source software platform for scalable, distributed computing
- Hadoop provides fast and reliable analysis of both structured data and unstructured data
- Apache Hadoop software library is essentially a framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model.
- Hadoop can scale up from single servers to thousands of machines, each offering local computation and storage.

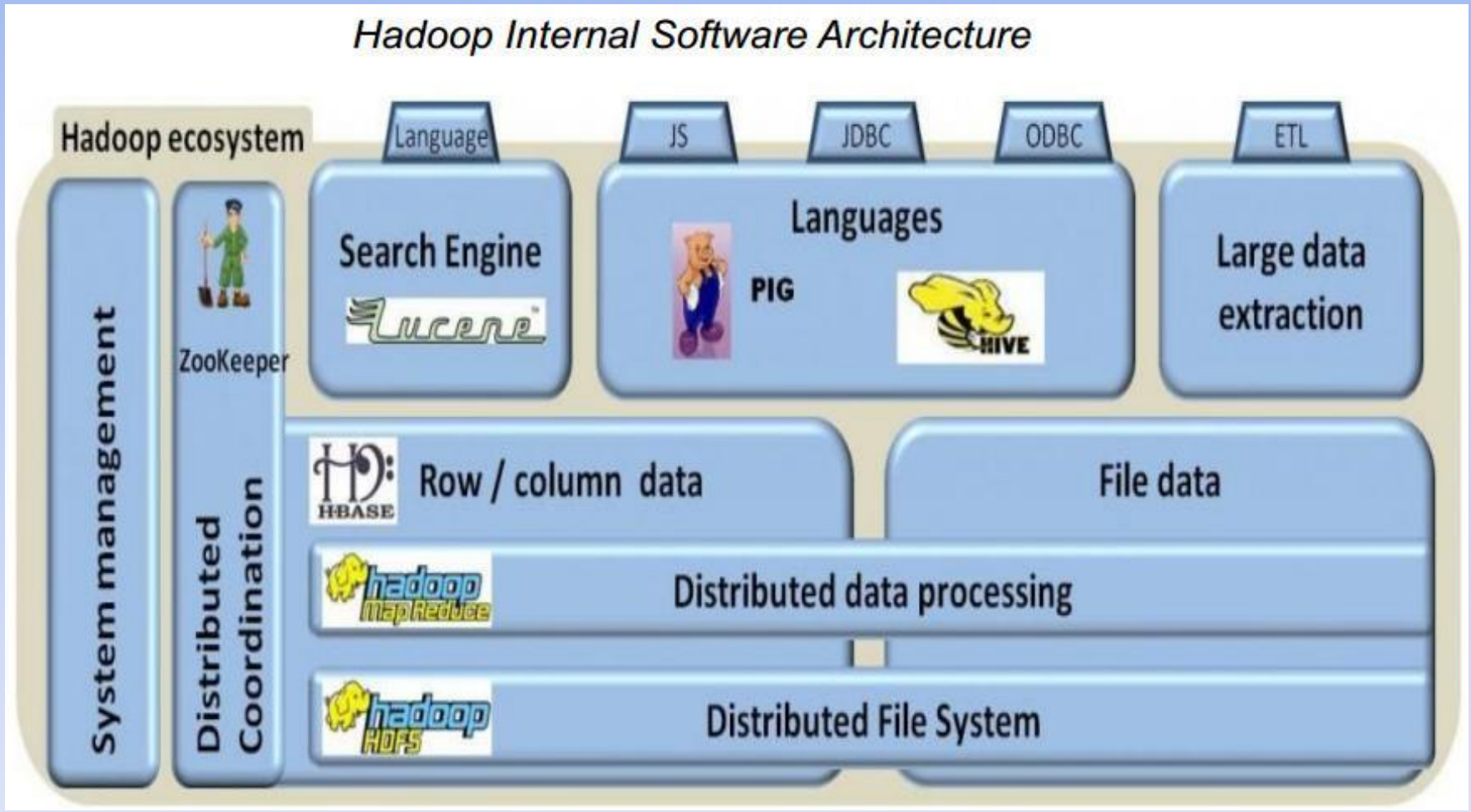
Hadoop Architecture



Apache Hadoop = HDFS + MapReduce

- Similar to kernel of an operating system (Hadoop Core)
- Related components are often deployed with Hadoop
- For example: HBase, Hive, Pig, Oozie, Flume, Sqoop
- Together, these components form a "Hadoop Stack"
- Not all components must be deployed

Hadoop Stack



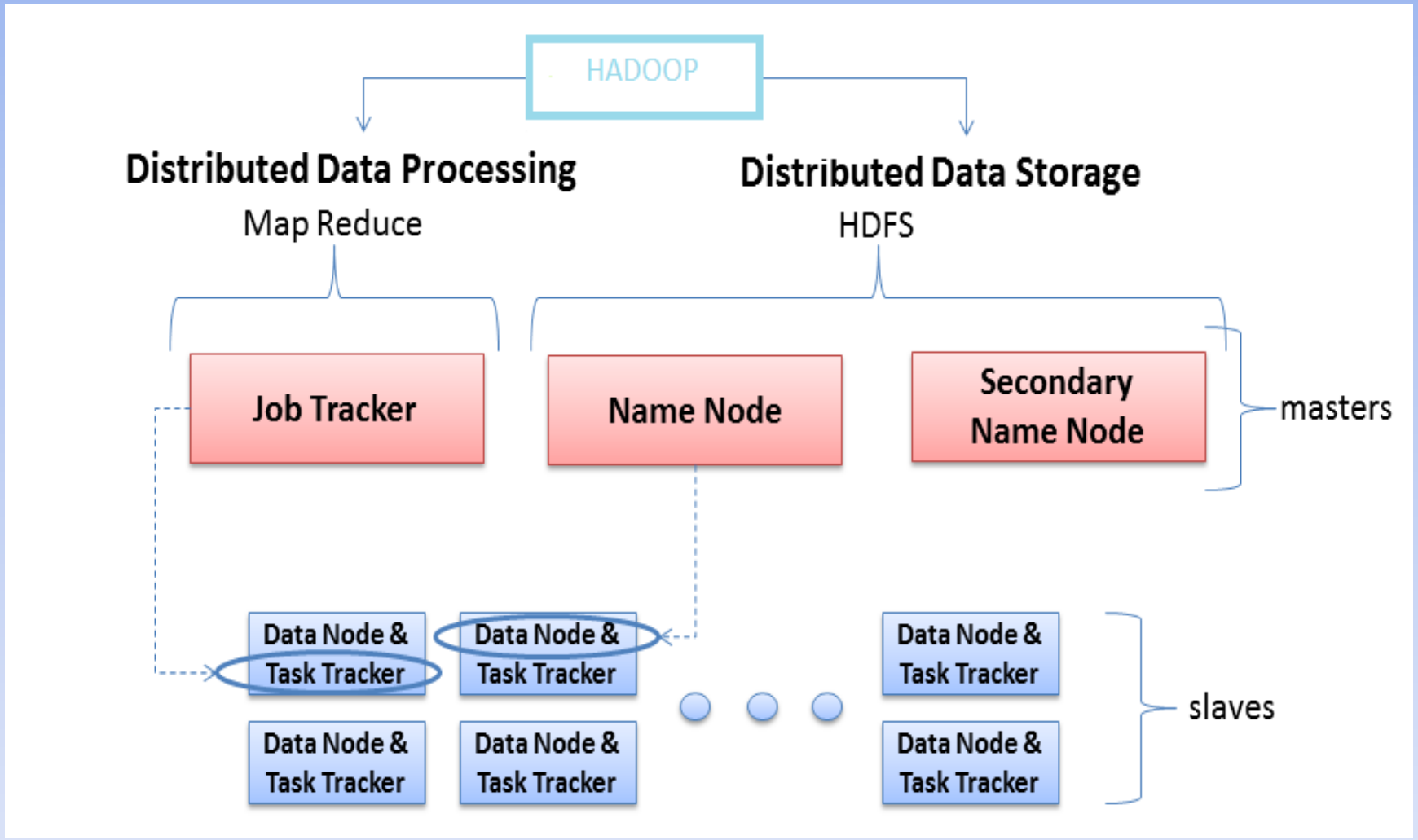
Hadoop Ecosystems Projects

- Hadoop Ecosystem Projects includes:
  - Hadoop Common utilities
  - Avro: A data serialization system with scripting languages.
  - Chukwa: managing large distributed systems.
  - HBase: A scalable, distributed database for large tables.
  - HDFS: A distributed file system.
  - Hive: data summarization and ad hoc querying.
  - MapReduce: distributed processing on compute clusters.
  - Pig: A high-level data-flow language for parallel computation.
  - ZooKeeper: coordination service for distributed applications.

Use Cases for Hadoop

- To aggregate “data exhaust” — messages, posts, blog entries, photos, video clips, maps, web graph
- To give data context — friends networks, social graphs, recommendations, collaborative filtering
- To keep apps running — web logs, system logs, system metrics, database query logs
- To deliver novel mashup services – mobile location data, clickstream data, SKUs, pricing

Hadoop Server Roles



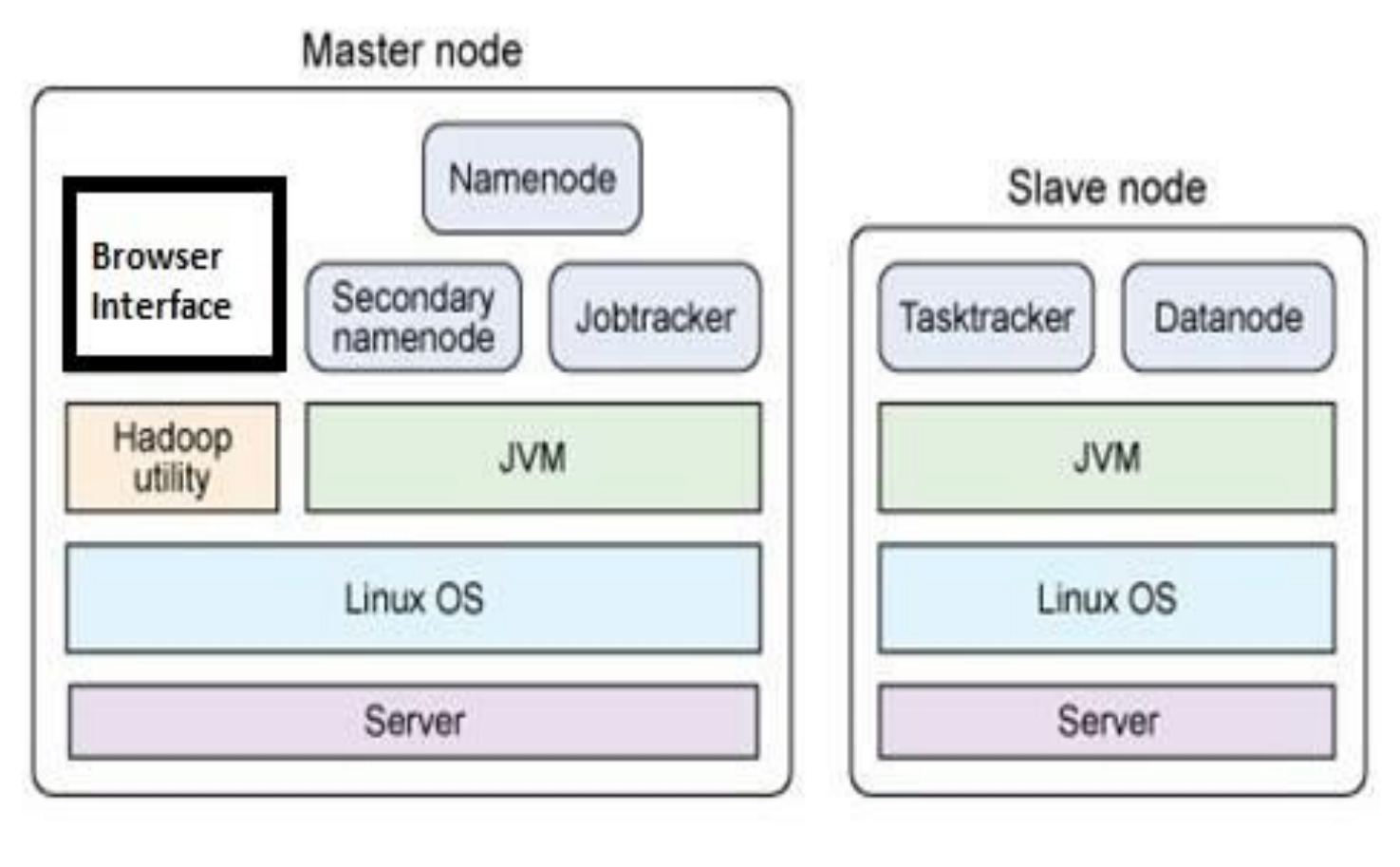
Assumptions and Goals of HDFS

- Hardware Failure
- Streaming Data Access (Best for batch processing)
- Large Data Sets
- Simple Coherency Model (write-once-read-many access model)
- Portability Across Heterogeneous Hardware and Software Platforms

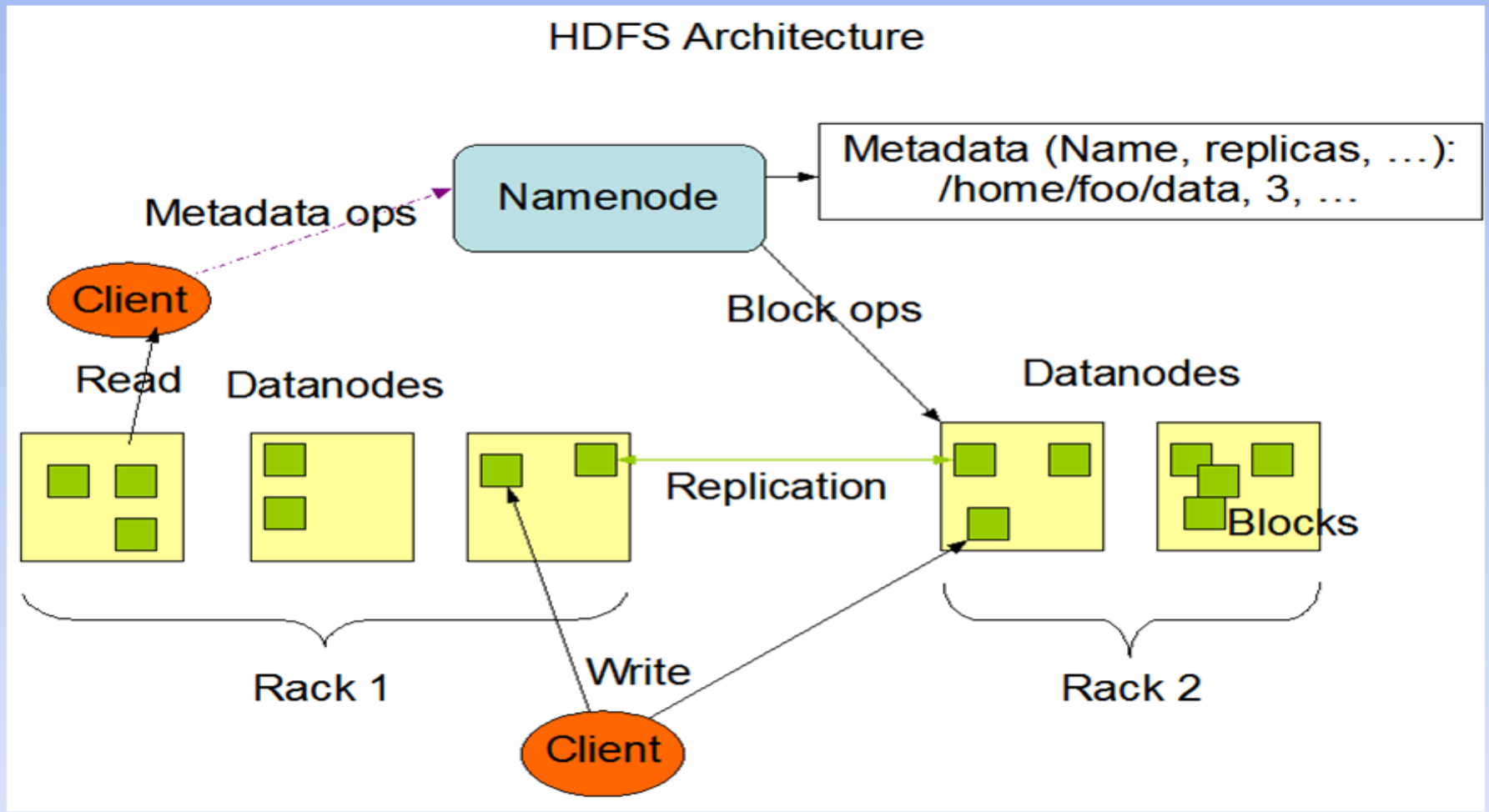
HDFS (Hadoop Distributed File System)

- A distributed file system that provides high-throughput access to application data
- HDFS uses a master/slave architecture in which one device (master) termed as NameNode controls one or more other devices (slaves) termed as DataNode.
- It breaks Data/Files into small blocks (128 MB each block) and stores on DataNode and each block replicates on other nodes to accomplish fault tolerance.
- NameNode keeps the track of blocks written to the DataNode.

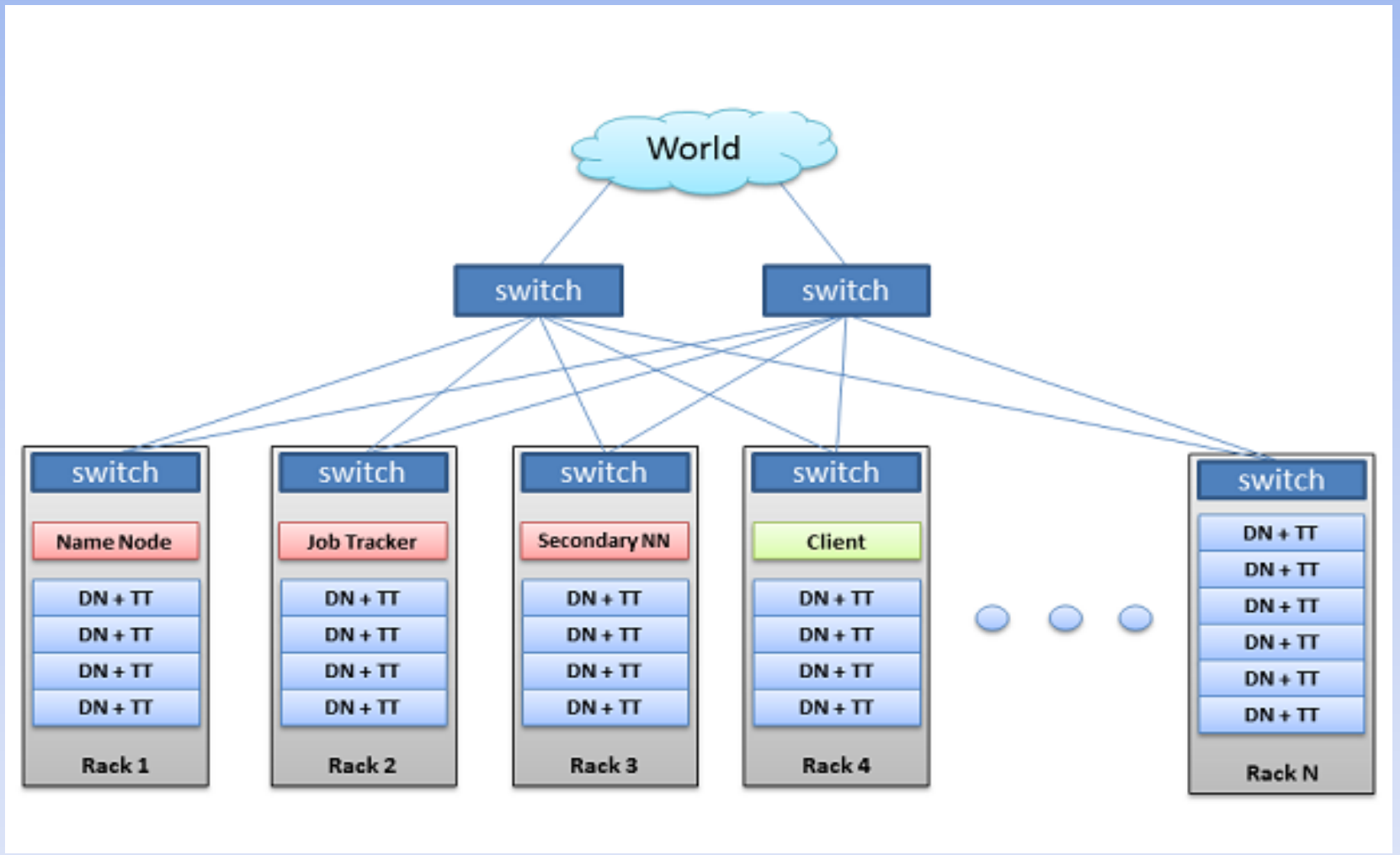
HDFS Cluster Architecture



HDFS Architecture



HDFS Cluster Architecture



HDFS Daemons

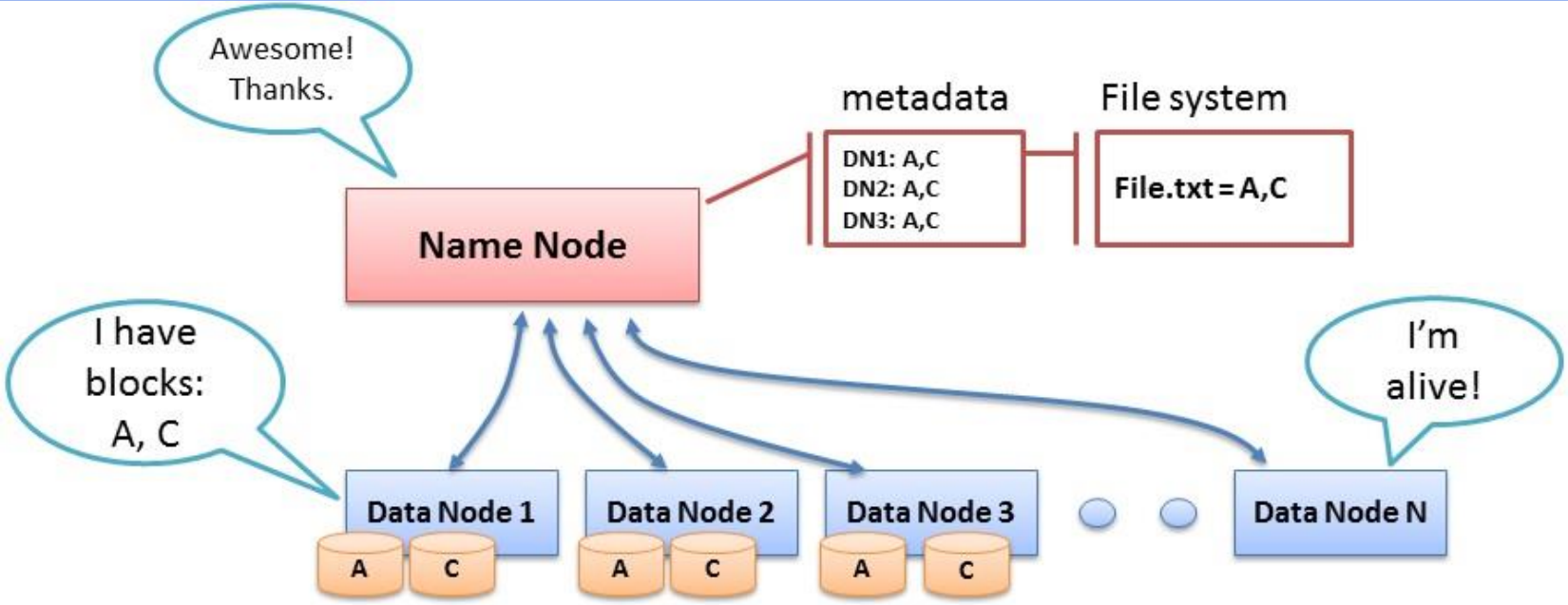
- NAME NODE
- DATA NODE
- SECONDRY NAME NODE

Name Node

- Keeps the metadata of all files/blocks in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Kind of block lookup dictionary(index or address book of blocks).

- Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives

Name Node (Contd.)

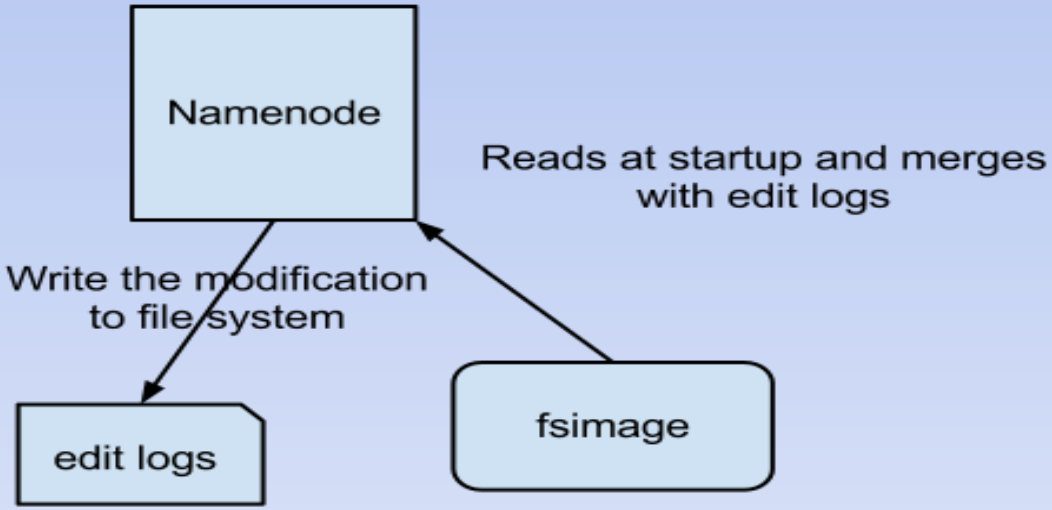


- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds
- If Name Node is down, HDFS is down

Name Node (Contd.)

fsimage - Its the snapshot of the filesystem when NameNode started

Edit logs - Its the sequence of changes made to the filesystem after NameNode started



Data Node

- DataNode stores data in the Hadoop Filesystem
- A functional filesystem has more than one DataNode, with data replicated across them
- On startup, a DataNode connects to the NameNode; spinning until that service comes up. It then responds to requests from the NameNode for filesystem operations.
- Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data

Data Replication

- **Why need data replication ?**

- HDFS is designed to handle large scale data in distributed environment
- Hardware or software failure, or network partition exist
- Therefore need replications for those fault tolerance

Replication (Contd.)

- **Replication factor**
  - Decided by users, and can be dynamically tuned.

- **How to Create replications efficiently ?**
  - Replication pipeline: Instead of single machine create replications, a pipe line is applied
  - Machine 1 make replication to machine 2, at the same time machine 2 make the replication to machine 3, etc.

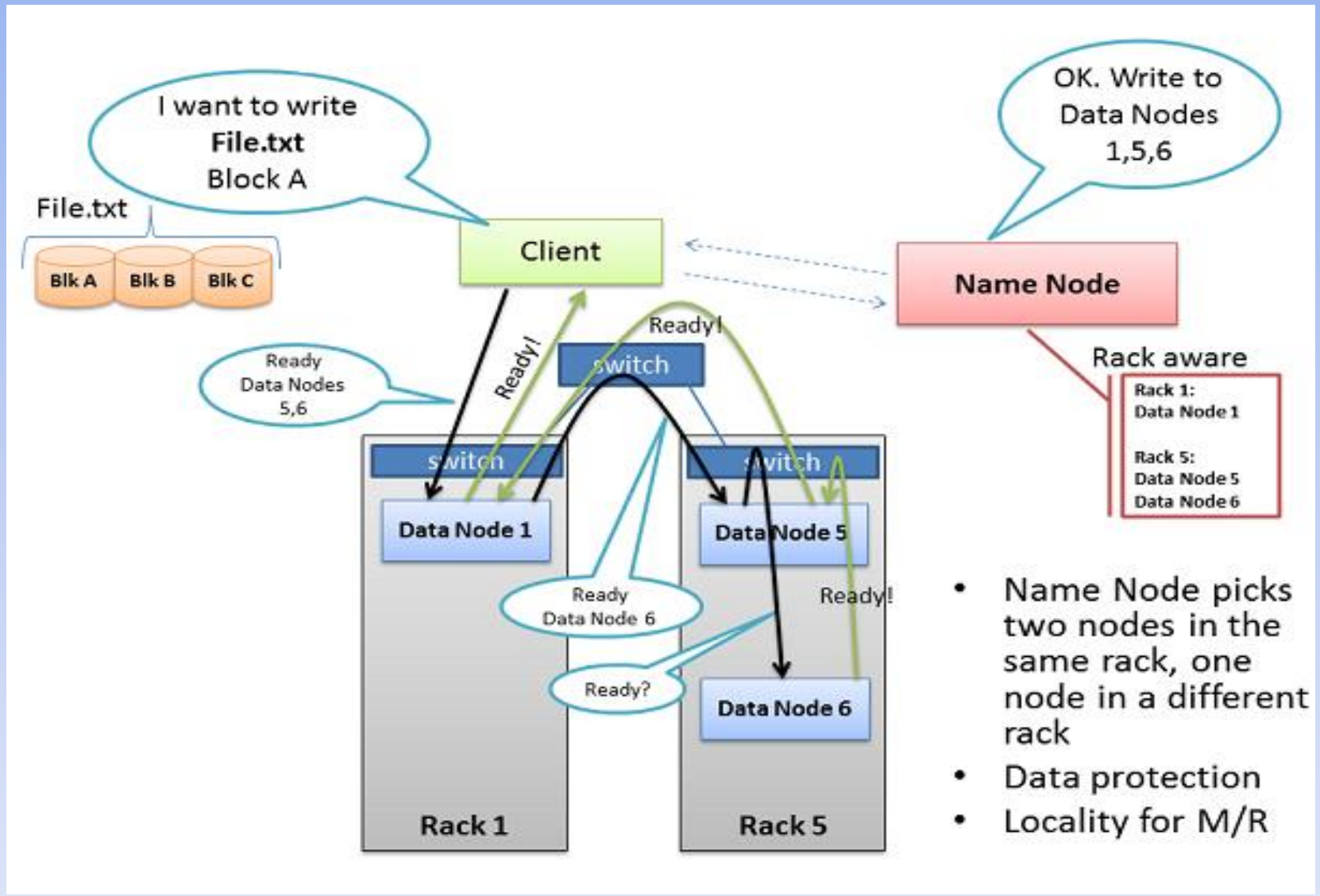
- **Replication placement**
  - High initialization time to create replication to all machines
  - An approximate solution: Only 3 replications

  One replication resides in current node

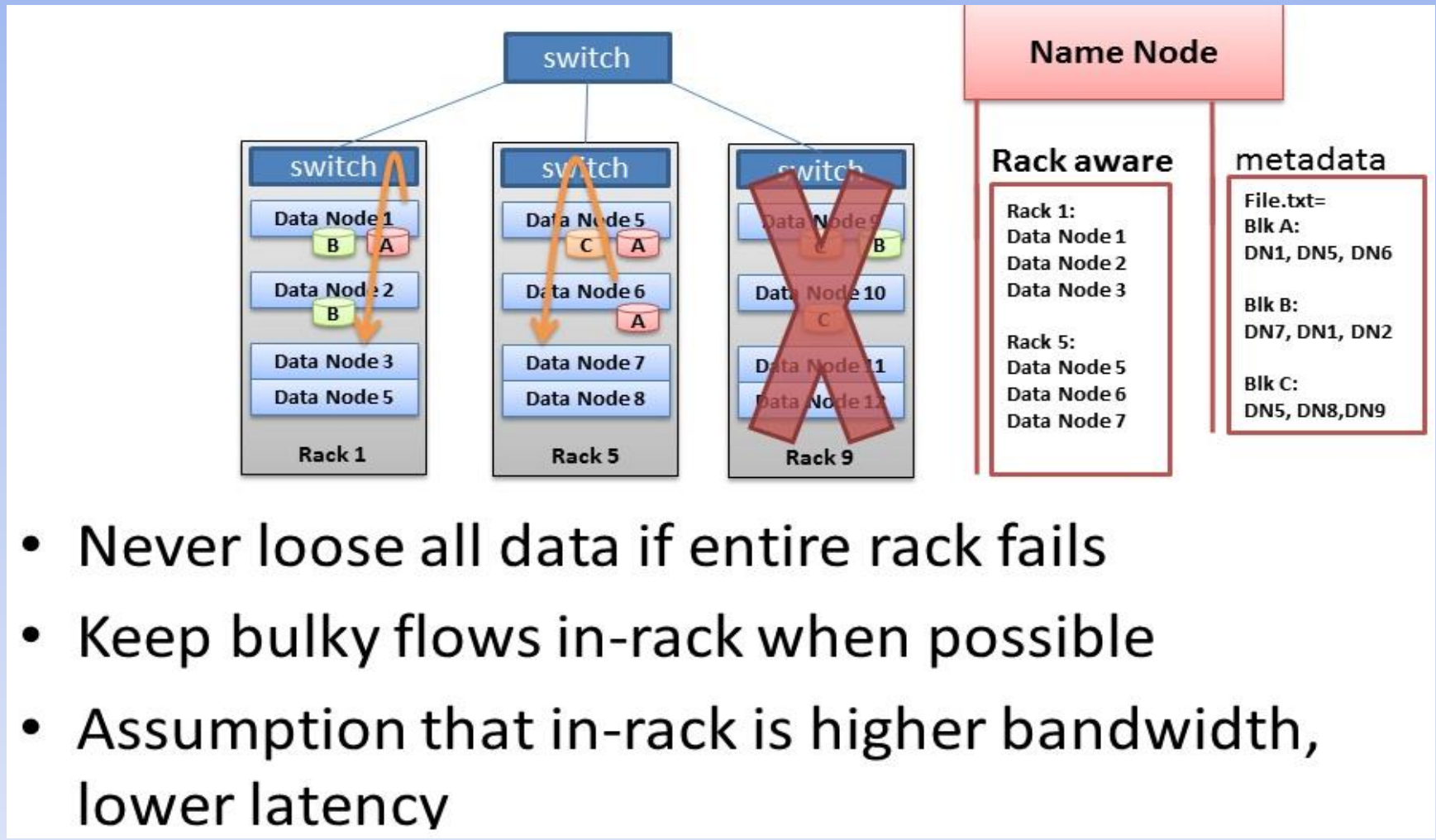
  One replication resides in current rack

  One replication resides in another rack

Replication Pipeline



Rack Awareness



- Never loose all data if entire rack fails
- Keep bulky flows in-rack when possible
- Assumption that in-rack is higher bandwidth, lower latency

Data Node Failure

Data Node Failure Condition

If a data node failed, Name Node could know the blocks it contains, create same replications to other alive nodes, and unregister this dead node

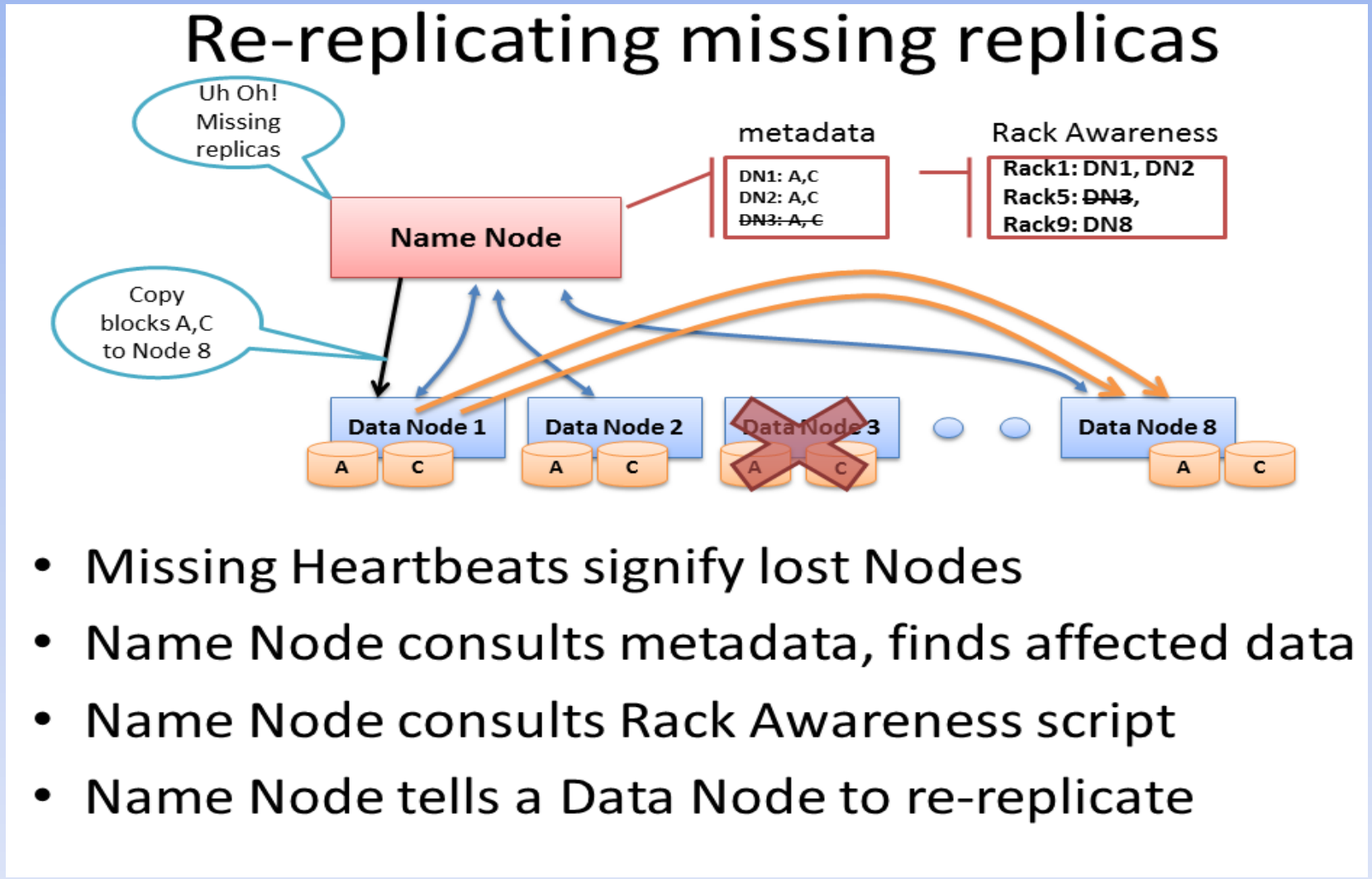
Data Integrity

Corruption may occur in network transfer, Hardware failure etc.

Apply checksum checking on the contents of files on HDFS, and store the checksum in HDFS namespace

If checksum is not correct after fetching, drop it and fetch another replication from other machines.

Heartbeats and Re-Replication

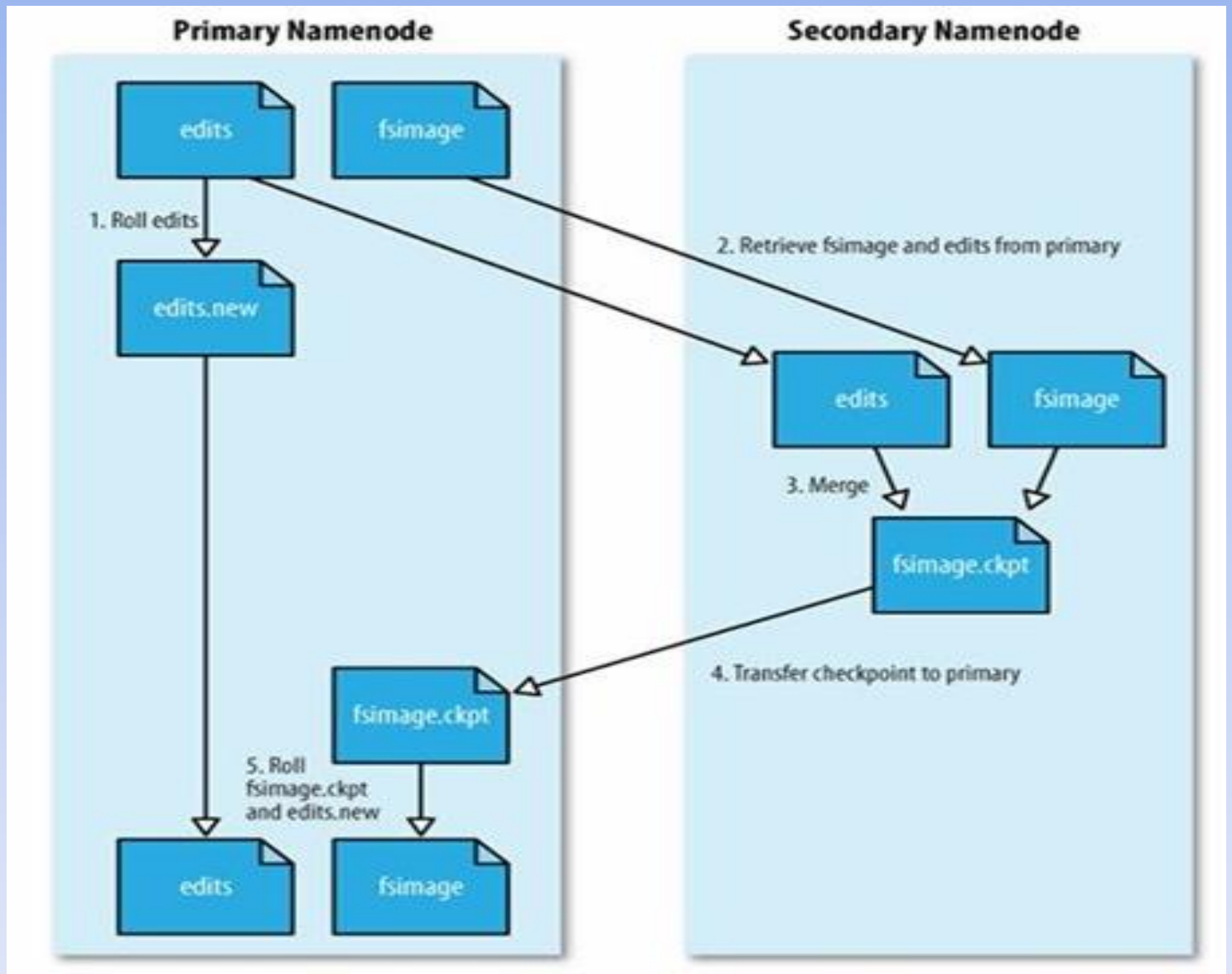


- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

Secondary Name Node

- Not a failover NameNode
- The only purpose of the secondary name-node is to perform periodic checkpoints. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the (primary and the only) name-node
- Default checkpoint time is one hour. It can be set to one minute on highly busy clusters where lots of write operations are being performed.

Secondary Name Node (Contd.)



Name Node Failure

- NameNode is the single point of failure in the cluster
- If NameNode is down due to software glitch, restart the machine
- If original NameNode can be restored, secondary can re-establish the most current metadata snapshot
- If machine don't come up, metadata for the cluster is irretrievable. In this situation create a new NameNode, use secondary to copy metadata to new primary, restart whole cluster
- Trick : Bring new NameNode up, but use DNS to make cluster believe it's the original one

Apache MapReduce

- A software framework for distributed processing of large data sets
- The framework takes care of scheduling tasks, monitoring them and re-executing any failed tasks.
- It splits the input data set into independent chunks that are processed in a completely parallel manner.
- MapReduce framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file system.

MapReduce Architecture

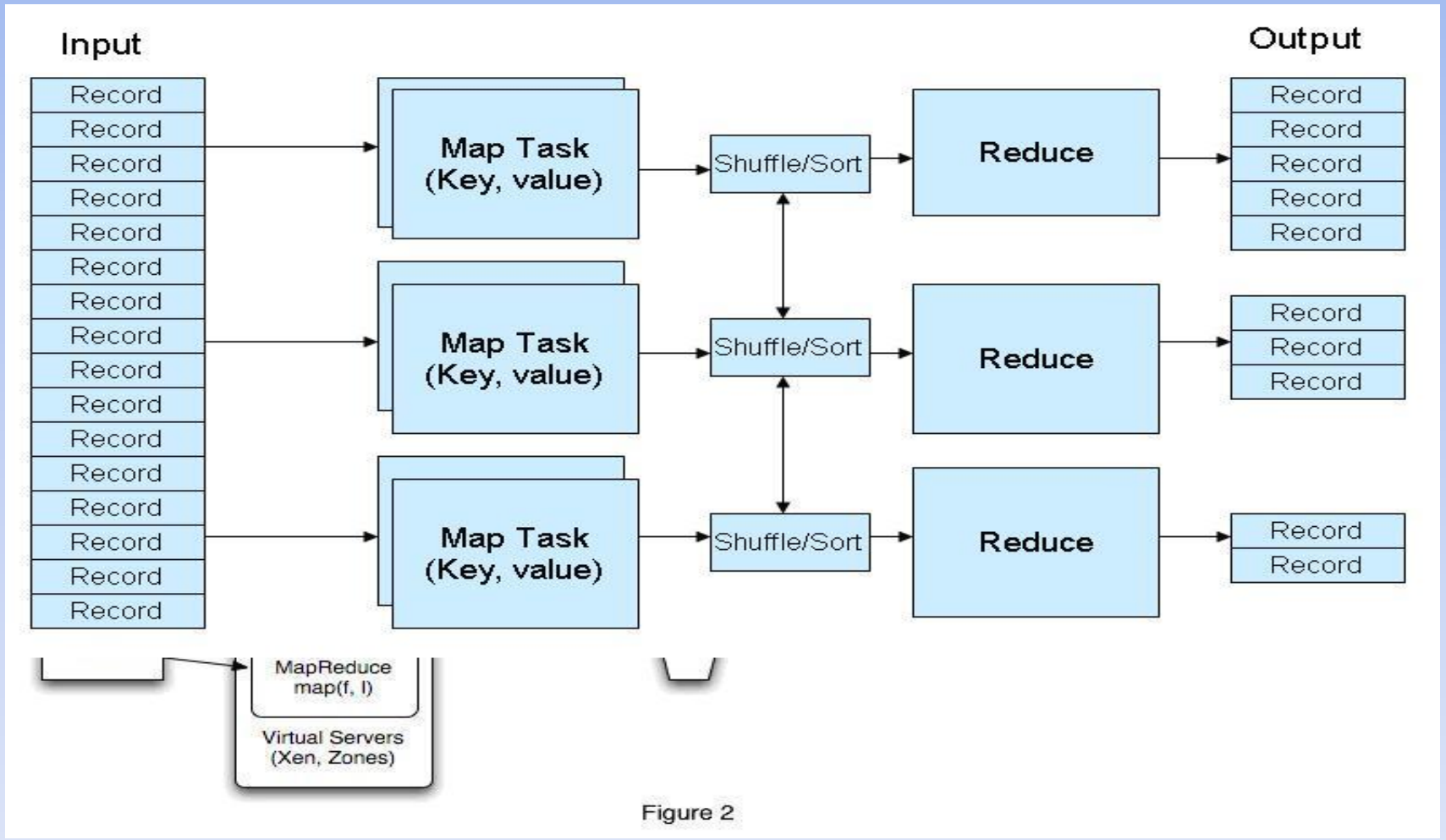
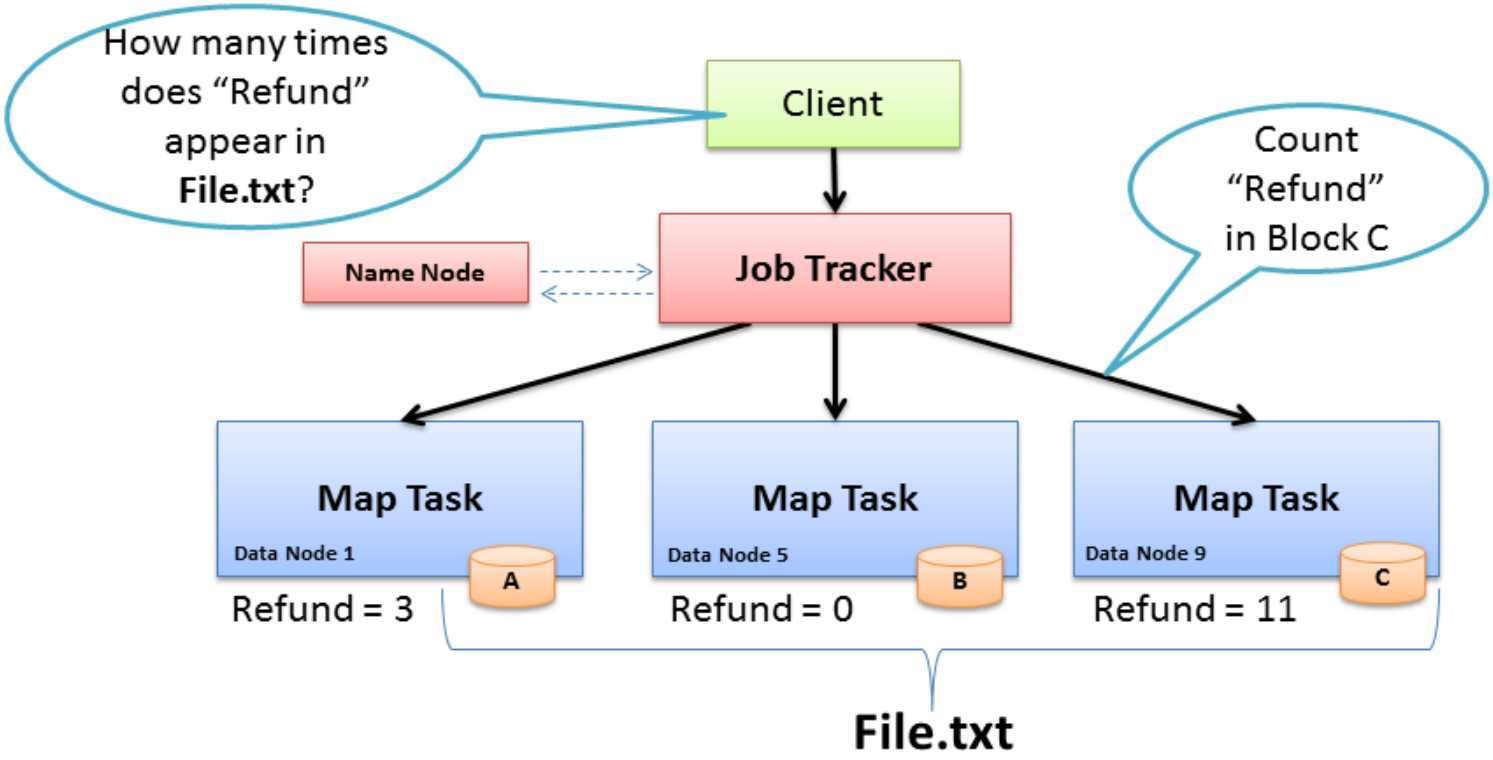


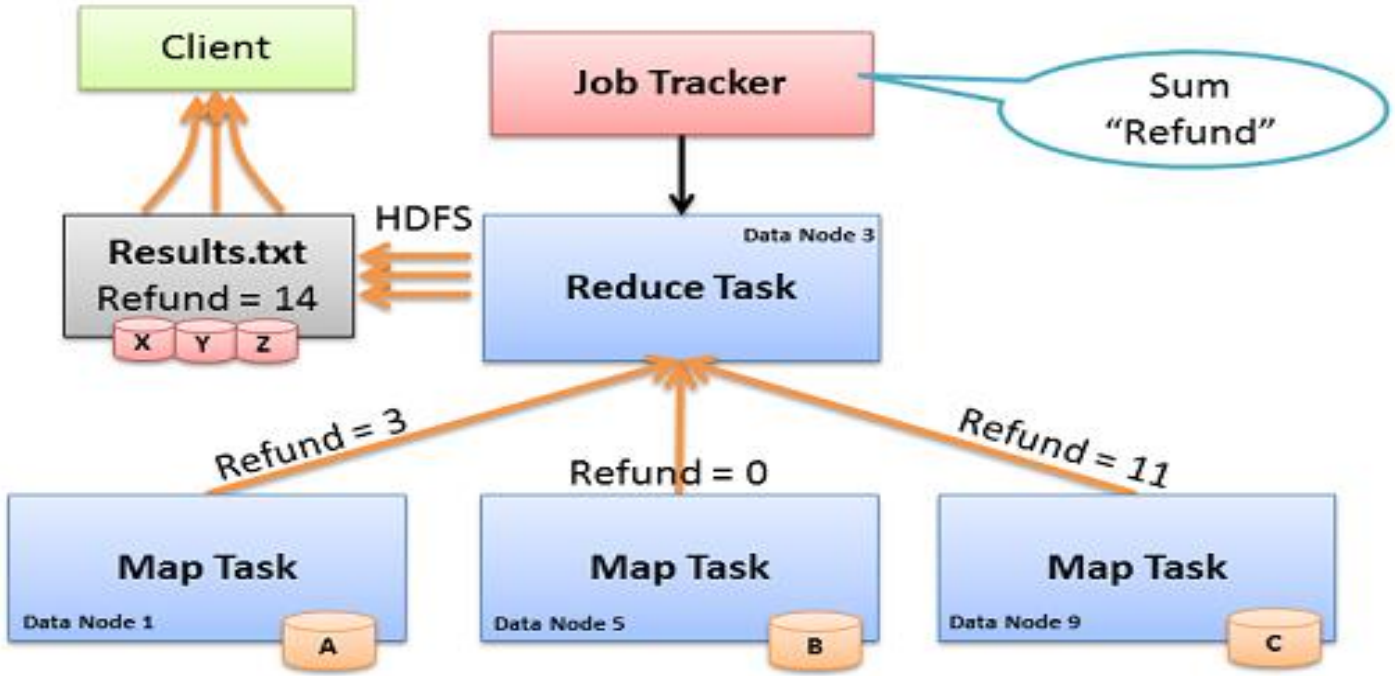
Figure 2

Map Task



- **Map:** "Run this computation on your local data"
- Job Tracker delivers Java code to Nodes with local data

Reduce Task



- **Reduce:** "Run this computation across Map results"
- Map Tasks send output data to Reducer over the network
- Reduce Task data output written to and read from HDFS

MapReduce Dataflow

- An input reader
- A Map function
- A partition function
- A compare function
- A Reduce function
- An output writer

MapReduce Daemons

- JOB TRACKER
- TASK TRACKER

JobTracker

- JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop

- JobTracker performs following actions in Hadoop :

  - It accepts the MapReduce Jobs from client applications
  - Talks to NameNode to determine data location
  - Locates available TaskTracker Node
  - Submits the work to the chosen TaskTracker Node

TaskTracker

- A TaskTracker node accepts map, reduce or shuffle operations from a JobTracker
- Its configured with a set of *slots*, these indicate the number of tasks that it can accept
- JobTracker seeks for the free slot to assign a job
- TaskTracker notifies the JobTracker about job success status.
- TaskTracker also sends the heartbeat signals to the job tracker to ensure its availability, it also reports the no. of available free slots with it.

Hadoop Configuration

- Untar hadoop-*.**.*.tar.gz to your user path

About Version:
The latest stable version 1.0.4 is recommended.

- edit the file conf/hadoop-env.sh to define at least JAVA_HOME to be the root of your Java installation

- edit the files to configure properties:

conf/core-site.xml:

```
<configuration>
  <property>
    <name>
      fs.default.name
    </name>
    <value>
      hdfs://localhost:9000
    </value>
  </property>
</configuration>
```

conf/hdfs-site.xml:

```
<configuration>
  <property>
    <name>
      dfs.replication
    </name>
    <value>
      1
    </value>
  </property>
</configuration>
```

conf/mapred-site.xml:

```
<configuration>
  <property>
    <name>
      mapred.job.tracker
    </name>
    <value>
      localhost:9001
    </value>
  </property>
</configuration>
```

Hadoop Releases

- **1.0.X** - current stable version, 1.0 release
- **1.1.X** - current beta version, 1.1 release
- **2.X.X** - current alpha version
- **0.23.X** - simmilar to 2.X.X but missing NN HA.
- **0.22.X** - does not include security
- **0.20.203.X** - old legacy stable version
- **0.20.X** - old legacy version

HDFS Access Methods

- Java API (For applications)
- Browser Interface (Next Slide)
- Hadoop FS Shell
  - Formatting filesystem with HDFS
    - bin/hadoop namenode -format
  - To add a directory
    - bin/hadoop dfs –mkdir abc
  - To list a directory
    - bin/hadoop dfs -ls /
  - To display content of a file
    - bin/hadoop dfs -cat filename

Hadoop Browser Interfaces

- MapReduce Job Tracker Web Interface

  http://localhost:50030/
- Task Tracker Web Interface

  http://localhost:50060/
- HDFS Name Node Web Interface

  http://localhost:50070/

Name Node Interface

NameNode vm1

Started:	Tue Mar 05 09:17:54 EST 2013
Version:	1.0.4, r1393290
Compiled:	Wed Oct 3 05:13:58 UTC 2012 by hortonfo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

51 files and directories, 56 blocks = 107 total. Heap Size is 27.83 MB / 966.69 MB (2%)

Configured Capacity	:	112.75 GB
DFS Used	:	4.11 GB
Non DFS Used	:	11.08 GB
DFS Remaining	:	97.56 GB
DFS Used%	:	3.64 %
DFS Remaining%	:	86.53 %
Live Nodes	:	2
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	7

References

- http://hadoop.apache.org/
- http://wiki.apache.org/hadoop/
- http://hadoop.apache.org/core/docs/current/hdfs_design.html
- http://wiki.apache.org/hadoop/FAQ

Questions?

- Write your queries to mkjoshi@expresskcs.com