



# University of Chittagong

## Computer Science and Engineering

Lab report on Cloud Computing lab

Course: Cloud Computing Lab (CSE 814)

Name: Lab report

**Submitted to:**

**Dr. Atiqur Rahman**

Associate Professor,

Dept. of Computer Science & Engineering

University of Chittagong, Bangladesh.

**Submitted by:**

**Ishra Naznin**

**ID: 18701069**

Session: 2017-18

Dept. of Computer Science & Engineering

University of Chittagong, Bangladesh

19<sup>th</sup> June, 2023

## **Programs on SAAS:**

**Question1. Create an word document of your class time table and store locally and on cloud with doc and pdf format.**

**Ans:** To create a Word document of my class timetable and store it locally and on the cloud in doc and pdf format, we can use the following steps involved:

1. Open Microsoft Word.
2. Click on the "Insert" tab.
3. In the Tables group, click on the "Table" icon.
4. In the Table dialog box, we enter the number of rows and columns we want for our table.
5. Click on the "OK" button.
6. Fill in the details of our class schedule, including the course name, instructor name, time, and location.
7. Then, save the document as a Word document (.docx) file. To store the document locally, click on the "File" tab. Then, in the Save As dialog box, select the "Save As Type" drop-down menu and choose "Word Document (.docx)" and click on the "Save" button.
8. To store the document on the cloud, click on the "File" tab. In the Save As dialog box, select the "Save As Type" drop-down menu and choose "PDF" then, click on the "Save" button. Our class timetable will now be stored in both doc and pdf format, locally.
9. To store the document on the cloud, we can use cloud storage services like Google Drive, Dropbox, or OneDrive. Sign up for an account if we don't have one already. Upload the document to your cloud storage account. To do this, first open the cloud storage service we're using and click on "Upload" or "New". Select the file we want to upload and click on "Open". Once the document is uploaded, we can access it from any device with an internet connection. To ensure that our document is safe, we have to make sure to back it up regularly. For this, we can set up automatic backups on our cloud storage service or manually save a copy of the document on an external hard drive.

By following these steps, we can create a class timetable document and store it both locally and on the cloud for easy access and backup. An example creating of class routine can be the following screenshot:

Day	Time	Class	Location
Monday	8:00am - 9:00am	English	Room 101
Monday	9:00am-10:00am	Math	Room 202
Monday	10:00am-10:15am	Break	-
Monday	10:15am - 11:15am	Science	Room 303
Monday	11:15am - 12:15pm	Lunch	-
Monday	12:15pm - 1:15pm	History	Room 404
Monday	1:15pm - 2:15pm	Spanish	Room 505
Monday	2:15pm - 2:45pm	Break	-
Monday	2:45pm - 3:45pm	Computer Science	Room 606
Tuesday	8:00am - 9:00am	English	Room 101
Tuesday	9:00am - 10:00am	Math	Room 202
Tuesday	10:00am - 10:15am	Break	-

**Figure 1: Screenshot of the spread sheet of some records.**

**Question2. Create a spread sheet which contains employee salary information and calculate gross and total salary using given formula.**

**Ans:** To create a spreadsheet containing employee salary information and calculate gross and total salary using the given formula, we can follow these steps:

1. Open Microsoft Excel and create a new worksheet.
2. In the first column, enter the names of the employees. In the second column, enter their basic salary.
3. In the third column, enter the formula for DA (Dearness Allowance) which is 10% of the basic salary. The formula for DA will be " $=B2*10\%$ ".
4. In the fourth column, enter the formula for HRA (House Rent Allowance) which is 30% of the basic salary. The formula for HRA will be " $=B2*30\%$ ".
5. In the fifth column, enter the formula for PF (Provident Fund) which is 10% of the basic salary if the basic salary is less than or equal to 3000, and 12% of the basic salary if the basic salary is greater than 3000. The formula for PF will be " $=IF(B2<=3000,B2*10\%,B2*12\%)$ ".
6. In the sixth column, enter the formula for TAX which is 10% of the basic salary if the basic salary is less than or equal to 1500, 11% of the basic salary if the basic salary is greater than 1500 and less than or equal to 2500, and 12% of the basic salary if the basic salary is greater than 2500. The formula for TAX will be " $=IF(B2<=1500,B2*10\%,IF(B2>1500,B2*11\%,IF(B2>2500,B2*12\%)))$ ".
7. In the seventh column, enter the formula for NET\_SALARY which is the sum of basic salary, DA, HRA, minus PF and TAX. The formula for NET\_SALARY will be " $=B2+C2+D2-E2-F2$ ".
8. Copy the formulas for DA, HRA, PF, TAX, and NET\_SALARY for all the employees in the respective columns.
9. In the eighth column, enter the formula for GROSS\_SALARY which is the sum of basic salary, DA, and HRA. The formula for GROSS\_SALARY will be " $=B2+C2+D2$ ".
10. Copy the formula for GROSS\_SALARY for all the employees in the respective column.
11. Save the spreadsheet on your computer in .xlsx format.

By following these steps, you can create a spreadsheet containing employee salary information and calculate gross and total salary using the given formula. Here a screenshot of some employee record is given:

	A	B	C	D	E	F	G
	Employee Name	Basic Salary	DA	HRA	PF	Tax	Net Salary
1							
2	Rahun	1000	100	300	100	100	1600
3	sadia	1200	120	360	120	120	1920
4	saima	1500	150	450	150	150	2400
5	sharmin	1000	100	300	100	100	1600
6	aman	1000	100	300	100	100	1600
7	riyad	1200	120	360	120	120	1920

**Figure 2: Screenshot of the spread sheet of some records.**

**Question 3. Prepare a ppt on cloud computing- introduction, models, services and architecture(use [www.zoho.com](http://www.zoho.com) and [docs.google.com](http://docs.google.com))-ppt should contain explanations, images and at least 20 pages.**

**Ans:** To prepare a PowerPoint presentation on cloud computing, we can follow these steps:

1. Open 'Zoho' Show or Google Slides and create a new presentation.
2. Add a title slide with the title "Introduction to Cloud Computing" and your name and date.
3. Add a slide with an overview of cloud computing, including a definition and key features.
4. Add a slide with the different cloud computing models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).
5. Add a slide with the different cloud computing services, including public cloud, private cloud, and hybrid cloud.
6. Add a slide with the cloud computing architecture, including the different layers such as infrastructure, platform, and application.
7. Add a slide with the benefits of cloud computing, including cost savings, scalability, and flexibility.
8. Add a slide with the challenges of cloud computing, including security, privacy, and compliance.
9. Add a slide with the future of cloud computing, including emerging technologies and trends.

10. Add images and diagrams to illustrate the concepts and models of cloud computing.
11. Use bullet points and short sentences to explain the concepts and models of cloud computing.
12. Use a consistent color scheme and font style throughout the presentation.
13. Add at least 20 slides to the presentation.
14. Use transitions and animations to make the presentation more engaging.
15. Add a conclusion slide with a summary of the key points and your contact information.

By following these steps, we can create a PowerPoint presentation on cloud computing that includes an introduction, models, services, and architecture, with explanations, images, and at least 20 pages.

**Question 4. Create your resume in a neat format using Google and Zoho cloud Cloud computing**

Ans: To create a neat resume using Google and Zoho cloud, you can follow these steps:

1. Open Google Docs or Zoho Recruit and create a new document.
2. Choose a resume template from the available options. In Google Docs, we can click on "Template Gallery" and select a resume template. In Zoho Recruit, we can go to the details page of the candidate and click on "More Actions" and then "Generate Formatted Resume".
3. Customize the template with our personal information, including our name, contact information, and professional summary.
4. Add our work experience, education, skills, and achievements in separate sections using lists and paragraphs.
5. Use bullet points to highlight your accomplishments and responsibilities in each job.
6. Use a consistent font style and size throughout the resume.
7. Add relevant images or graphics to make the resume more visually appealing.
8. Use keywords and phrases that match the job description to make your resume more relevant to the job.
9. Proofread the resume for any errors or typos.
10. Save the resume on your computer in .doc or .pdf format.

By following these steps, we can create a neat resume using Google and Zoho cloud that includes our personal information, work experience, education, skills, and achievements in separate sections using lists and paragraphs.

## **Programs on PAAS**

### **Question 1. Write a Google app engine program to generate n even numbers and deploy it to Google cloud**

Ans: To write a Google App Engine program to generate n even numbers and deploy it to Google Cloud, we follow these steps:

1. Open the Google Cloud Console and create a new project.
2. Enable the App Engine API for the project.
3. Install the Google Cloud SDK on our local machine.
4. Create a new directory for your App Engine program.
5. Create a new Python file in the directory and name it "main.py".
6. Write the Python code to generate n even numbers. For example, we can use a for loop to iterate through the range of n and print the even numbers.
7. Define a route for the app that calls the function and returns the list of even numbers as a string response.
8. Test the program locally by running the command "python evenGenerator.py" in the terminal.
9. Create a requirements.txt file in the directory to list the dependencies for the app.
10. Create an app.yaml file in the directory to define the settings for the App Engine.
11. Edit the app.yaml file to specify the runtime environment and other settings.
12. Deploy the program to the App Engine using the command "gcloud app deploy".
13. Access the program on the App Engine by visiting the URL provided by the deployment process.

By following these steps, we can write a Python program to generate n even numbers and deploy it to Google Cloud using App Engine. We can use Python and Flask to write the code, and the Google Cloud SDK and the App Engine API to create and deploy the program.

```

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    n = int(input("Enter the value of n: "))
    even_numbers = []
    count = 0
    number = 2 # Start with the first even number
    while count < n:
        even_numbers.append(str(number))
        count += 1
        number += 2 # Increment by 2 to get the next even number

    even_numbers_string = ", ".join(map(str, even_numbers))
    return even_numbers_string

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)

```

Figure 3.1: Code screenshot of the Google app engine program to generate n even numbers.

```

^Cbash-5.1python3 evenGenerator.py
* Serving Flask app 'evenGenerator'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production dep
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 112-977-223
Enter the value of n: 5
127.0.0.1 - - [12/Jun/2023 18:46:04] "GET /?authuser=0 HTTP/1.1" 200 -
^[[3~

```

Figure 3.2: Programming running in Google app engine console

```

Enter the value of n: 5
2, 4, 6, 8, 10

```

Figure 3.4: Output of the Google app engine program to generate n even numbers.

## Question 2. Google app engine program multiply two matrices

**Ans:** To write a Google App Engine program to multiply two matrices, we can follow these steps:

1. Open a text editor and create a new Python file named " matrix.py".
2. Write the Python code to multiply two matrices. we use nested for loops to iterate through the rows and columns of the matrices and perform the multiplication.
3. Import the Flask module and create a new Flask app instance.
4. Define a route for the app that takes two matrix parameters.
5. Write a function that multiplies the two matrices using nested for loops and returns the result as a matrix.
6. Define a route for the app that calls the function and returns the result as a string response.
7. Test the program locally by running the command "python matrix.py" in the terminal.
8. Create a requirements.txt file in the directory to list the dependencies for the app.
9. Create an app.yaml file in the directory to define the settings for the App Engine.
10. Edit the app.yaml file to specify the runtime environment and other settings.
11. Deploy the program to the App Engine using the command "gcloud app deploy".
12. Access the program on the App Engine by visiting the URL provided by the deployment process.

By following these steps, we write a Google App Engine program to multiply two matrices. We use Python and Flask to write the code, and the Google Cloud SDK and the App Engine API to create and deploy the program. Here the screenshot of code, running in google app engine console and the output is given below:



```

from flask import Flask(__name__)
@app.route('/')
def hello():
    matrix1 = [[1, 2, 3], [4, 5, 6]]
    matrix2 = [[7, 8], [9, 10], [11, 12]]
    rows1 = len(matrix1)
    cols1 = len(matrix1[0])
    rows2 = len(matrix2)
    cols2 = len(matrix2[0])
    # Check if matrices can be multiplied
    if cols1 != rows2:
        return "Matrices cannot be multiplied."
    # Create a result matrix filled with zeros
    result = [[0] * cols2 for _ in range(rows1)]
    # Multiply matrices
    for i in range(rows1):
        for j in range(cols2):
            for k in range(cols1):
                result[i][j] += matrix1[i][k] * matrix2[k][j]
    # Convert result to string
    result_string = ""
    for row in result:
        result_string += " ".join(str(element) for element in row)
        result_string += "\n"
    return result_string
if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)

```

Figure 4.1: Code screenshot of the Google app engine program to multiply two matrices

```

bash-5.1# python3 matrix.py
* Serving Flask app 'matrix'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 112-977-223
127.0.0.1 - - [12/Jun/2023 19:14:12] "GET /?authuser=0 HTTP/1.1" 200 -

```

Figure 4.2: Programming running in Google app engine console

```

The multiplication result is:
[58, 64]
[139, 154]

```

Figure 4.3: Output of the program to multiply two matrices.

**3. Google app engine program to validate user; create a database login (username, password) in mysql and deploy to cloud.**

**Ans:** To create a Google App Engine program to validate user and create a database login (username, password) in MySQL and deploy it to the cloud, we use the following these steps:

1. Open your Google Cloud account and click SQL in the left menu.
2. Click Create Instance and choose MySQL as your database engine.
3. Fill in the instance information, including the Instance ID, root password, and region.
4. Record the username and password for the database login.
5. Create a database using either the Google Cloud Console or a data visualization tool like MySQL Workbench.
6. Write the Python code to validate user and create a database login. We use the Flask module to create a web app that takes user input and interacts with the MySQL database.
7. Import the Flask module and create a new Flask app instance.
8. Define a route for the app that takes user input for the username and password.
9. Write a function that validates the user input and creates a new database login in MySQL using the recorded username and password.
10. Define a route for the app that calls the function and returns a message indicating whether the login was successful.
11. Test the program locally by running the command "databae.py" in the terminal.
12. Create a 'requirements.txt' file in the directory to list the dependencies for the app.
13. Create an app.yaml file in the directory to define the settings for the App Engine.
14. Edit the app.yaml file to specify the runtime environment and other settings.
15. Deploy the program to the App Engine using the command "gcloud app deploy".
16. Access the program on the App Engine by visiting the URL provided by the deployment process.

By following these steps, we create a Google App Engine program to validate user and create a database login in MySQL and deploy it to the cloud. We use Python and Flask to write the code, and the Google Cloud SDK and the App Engine API to create and deploy the program.

Here the screenshot of code, running in google app engine console and the output is given below:

```

import os
import MySQLdb
from flask import Flask, request
app = Flask(__name__)
@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    password = request.form.get('password')
    if not username or not password:
        return 'Invalid username or password', 400
    try:
        db = MySQLdb.connect(
            unix_socket=f'/cloudsql/{os.environ["MYSQL_INSTANCE_CONNECTION_NAME"]}',
            user=os.environ['MYSQL_USER'],
            password=os.environ['MYSQL_PASSWORD'],
            database=os.environ['MYSQL_DATABASE']
        )
        cursor = db.cursor()
        cursor.execute("SELECT * FROM users WHERE username=%s AND password=%s", (username, password))
        user = cursor.fetchone()
        db.close()

        if user:
            return 'Login successful'
        else:
            return 'Invalid username or password', 401
    except Exception as e:
        return str(e), 500

if __name__ == '__main__':
    app.run()

```

**Figure 5: Code screenshot of the Google app engine program to validate user; create a database login (username, password) in mysql.**

#### **4. Write a Google app engine program to display nth largest no from the given list of numbers and deploy it in Google cloud.**

**Ans:** To write a Google App Engine program to display the nth largest number from a given list of numbers and deploy it in Google Cloud, we follow these steps:

1. Write a Python program that takes a list of numbers as input and returns the nth largest number in the list.
2. Import the Flask module and create a new Flask app instance.
3. Define a route for the app that takes the list of numbers and the value of n as parameters.
4. Write a function that sorts the list of numbers in descending order and returns the nth largest number.
5. Define a route for the app that calls the function and returns the nth largest number.
6. Test the program locally by running the command "python nthLargest.py" in the terminal.
7. Create a 'requirements.txt' file in the directory to list the dependencies for the app.
8. Create an 'app.yaml' file in the directory to define the settings for the App Engine.
9. Edit the 'app.yaml' file to specify the runtime environment and other settings.
10. Deploy the program to the App Engine using the command "gcloud app deploy".

11. Access the program on the App Engine by visiting the URL provided by the deployment process.

By following these steps, we write a Google App Engine program to display the nth largest number from a given list of numbers and deploy it in Google Cloud. We use Python and Flask to write the code, and the Google Cloud SDK and the App Engine API to create and deploy the program. To store the list of numbers, we use a database service like Cloud SQL or Google Cloud Storage.

Here the screenshot of code, running in google app engine console and the output is given below:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    # Take user input for the list of numbers
    number_list = input("Enter a list of numbers, separated by spaces: ").split()
    number_list = [int(num) for num in number_list]

    # Take user input for the nth value
    nth = int(input("Enter the value of n: "))
    numbers.sort(reverse=True) # Sort the numbers in descending order
    if n > len(numbers):
        return "Invalid value of n"

    return "The {nth} largest number is: ",str(numbers[n - 1])

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

**Figure 6.1:** Code screenshot of the Google app engine program to display nth largest no from the given list of numbers and deploy it in Google cloud.

```
* Debug mode: on
WARNING: This is a development server. Do not use it in a prod
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 112-977-223
Enter a list of numbers, separated by spaces: 6 7 3 9 24 74
Enter the value of n: 2
```

**Figure 6.2:** Programming running in Google app engine console

```
C:\Users\Acer\OneDrive\Downloads>python 1largenumber.py
Enter a list of numbers, separated by spaces: 6 7 3 9 24 76
Enter the value of n: 2
The 2 largest number is: 24
```

**Figure 6.3: Output of the program to display nth largest no from the given list of numbers**

### **5. Google app engine program to validate the user use mysql to store user info and deploy on to cloud**

**Ans:** To write a Google App Engine program to validate the user using MySQL to store user info and deploy it on the cloud, we follow these steps:

1. Create a MySQL database in Google Cloud SQL to store user information.
2. Create a table in the database to store the user's login credentials.
3. Write a Python program using Flask to validate user login information.
4. Import the Flask module and create a new Flask app instance.
5. Define a route for the app that takes the username and password as parameters.
6. Write a function that queries the database to check if the username and password match.
7. Define a route for the app that calls the function and returns a response indicating whether the login was successful or not.
8. Test the program locally by running the command "python user.py" in the terminal.
9. Create a requirements.txt file in the directory to list the dependencies for the app.
10. Create an app.yaml file in the directory to define the settings for the App Engine.
11. Edit the app.yaml file to specify the runtime environment and other settings.
12. Deploy the program to the App Engine using the command "gcloud app deploy".
13. Access the program on the App Engine by visiting the URL provided by the deployment process.

By following these steps, we write a Google App Engine program to validate the user using MySQL to store user info and deploy it on the cloud. We use Python and Flask to write the code, and the Google Cloud SDK and the App Engine API to create and deploy the program. To connect to the MySQL database from the App Engine, we use the Cloud SQL service provided by Google Cloud.

Here the screenshot of code, running in google app engine console and the output is given below:

```

import os
import MySQLdb
from flask import Flask, request
app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    password = request.form.get('password')

    if not username or not password:
        return 'Invalid username or password', 400

    try:
        db = MySQLdb.connect(
            unix_socket=f'/cloudsql/{os.environ["MYSQL_CONNECTION_NAME"]}',
            user=os.environ['MYSQL_USER'],
            passwd=os.environ['MYSQL_PASSWORD'],
            db=os.environ['MYSQL_DATABASE']
        )
        cursor = db.cursor()
        cursor.execute("SELECT * FROM users WHERE username=%s AND password=%s", (username, password))
        user = cursor.fetchone()
        db.close()

        if user:
            return 'Login successful'
        else:
            return 'Invalid username or password', 401
    except Exception as e:
        return str(e), 500

if __name__ == '__main__':
    app.run()

```

Figure 7.1: Code screenshot of the Google app engine program ‘user.py’ to validate the user use mysql to store user info.

```

runtime: python39

instance_class: F2

automatic_scaling:
  target_cpu_utilization: 0.65
  min_instances: 1
  max_instances: 10

env_variables:
  MYSQL_USER: 'your_mysql_user'
  MYSQL_PASSWORD: 'your_mysql_password'
  MYSQL_DATABASE: 'your_mysql_database'
  MYSQL_CONNECTION_NAME: 'your_mysql_connection_name'

```

Figure 7.2: Code screenshot of the Google app engine program of ‘app.yaml’ to validate the user use mysql to store user info

## MapReduce

MapReduce is a programming model used to process large amounts of data in parallel across a distributed cluster of computers. MapReduce is a framework developed by Google. The word count problem is a classic example of a MapReduce program. In this problem, the goal is to count the number of occurrences of each word in a given text file. The input is typically a collection of text documents, and the output is a list of words along with their respective counts.

The MapReduce solution to the word count problem consists of two steps:

Map phase: In this phase, the input data is split into key-value pairs, where the key is the word and the value is 1. Each mapper processes a portion of the input data and outputs intermediate key-value pairs.

Shuffle phase: In this phase, the intermediate key-value pairs are sorted and grouped by key. All the values associated with a particular key are sent to the same reducer.

Reduce phase: In this phase, the values associated with each key are aggregated to produce the final output. Each reducer processes a subset of the intermediate key-value pairs and outputs the final count for each word.

Final Output: The output of the reduce phase is a list of words along with their respective counts.

The MapReduce framework handles the distribution of the map and reduces tasks across multiple machines, making it suitable for processing large amounts of data in a parallel and scalable manner. This approach efficiently solves the word count problem by leveraging the parallel processing capabilities of the MapReduce framework.

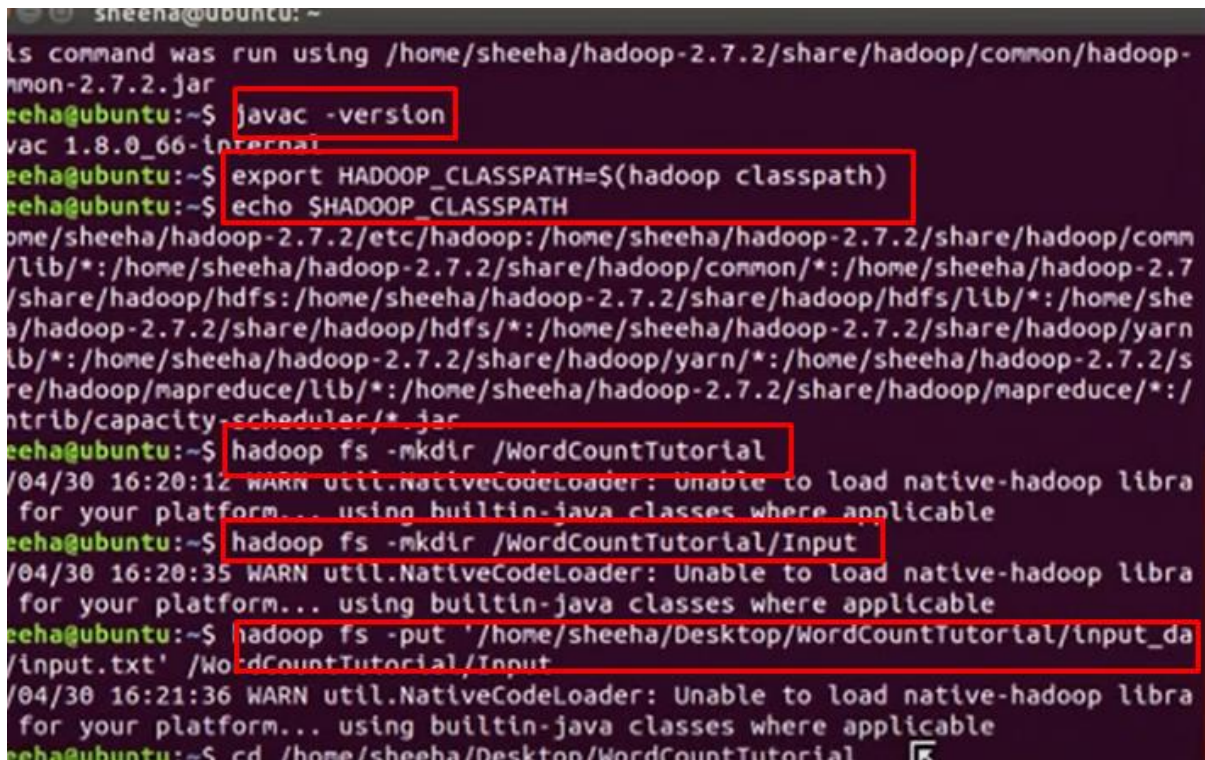
To implement MapReduce word count in Linux, you can follow these general steps:

- Install Hadoop on my Linux system.
- Create a text file containing the input data.
- Copy the text file to the Hadoop Distributed File System (HDFS) using the command "hadoop fs -put input\_file.txt /input".
- Write a MapReduce program in a programming language like Java that counts the number of occurrences of each word in the input data.
- Compile the MapReduce program using the Hadoop compiler.
- Run the MapReduce program on the Hadoop cluster using the command "hadoop jar wordcount.jar input output".
- View the output of the MapReduce program using the command "hadoop fs -cat /output/part-r-000000".

These steps assume that you have a Hadoop cluster set up and running on your Linux system. The MapReduce program should be designed to take input from the HDFS and output the results to the HDFS. The output of the program can be viewed using the "hadoop fs -cat"

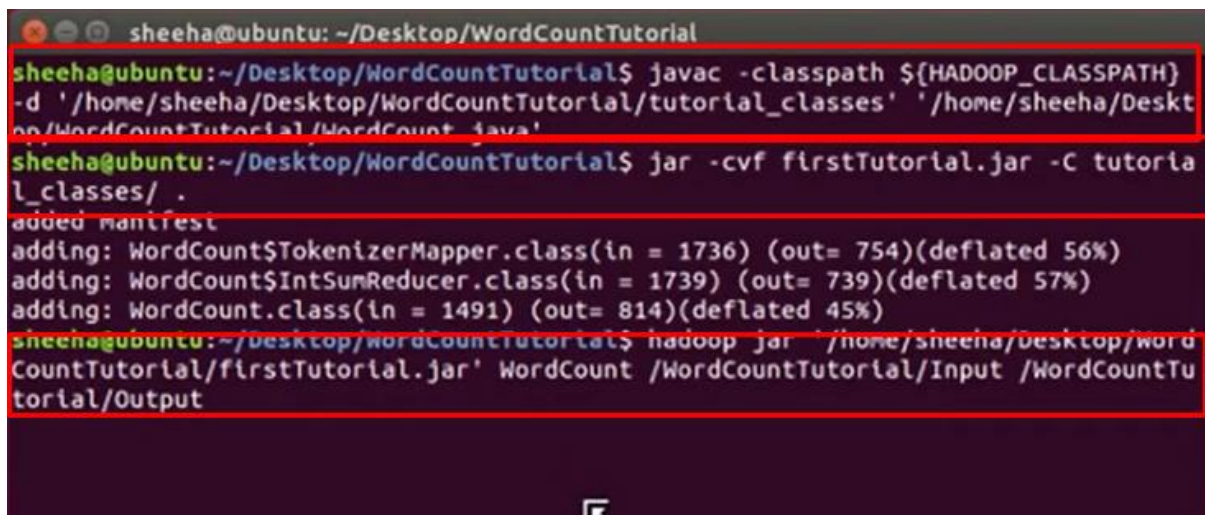


command. Here are the screenshot of my word count program implementation in linux platform:

A terminal window showing a series of commands to set up Hadoop. The user is at the prompt 'sheeha@ubuntu: ~'. The commands are: 'javac -version', 'export HADOOP\_CLASSPATH=\$(hadoop classpath)', 'echo \$HADOOP\_CLASSPATH', 'hadoop fs -mkdir /WordCountTutorial', 'hadoop fs -mkdir /WordCountTutorial/Input', 'hadoop fs -put '/home/sheeha/Desktop/WordCountTutorial/input\_data/input.txt' /WordCountTutorial/Input', and 'cd /home/sheeha/Desktop/WordCountTutorial'. Several warning messages from 'util.NativeCodeLoader' are visible in the background.

```
sheeha@ubuntu: ~  
is command was run using /home/sheeha/hadoop-2.7.2/share/hadoop/common/hadoop-  
mon-2.7.2.jar  
sheeha@ubuntu:~$ javac -version  
javac 1.8.0_66-internal  
sheeha@ubuntu:~$ export HADOOP_CLASSPATH=$(hadoop classpath)  
sheeha@ubuntu:~$ echo $HADOOP_CLASSPATH  
/home/sheeha/hadoop-2.7.2/etc/hadoop:/home/sheeha/hadoop-2.7.2/share/hadoop/comm  
/lib/*:/home/sheeha/hadoop-2.7.2/share/hadoop/common/*:/home/sheeha/hadoop-2.7  
/share/hadoop/hdfs:/home/sheeha/hadoop-2.7.2/share/hadoop/hdfs/lib/*:/home/she  
a/hadoop-2.7.2/share/hadoop/hdfs/*:/home/sheeha/hadoop-2.7.2/share/hadoop/yarn  
lib/*:/home/sheeha/hadoop-2.7.2/share/hadoop/yarn/*:/home/sheeha/hadoop-2.7.2/s  
re/hadoop/mapreduce/lib/*:/home/sheeha/hadoop-2.7.2/share/hadoop/mapreduce/*:/  
ntrib/capacity-scheduler/*_jar  
sheeha@ubuntu:~$ hadoop fs -mkdir /WordCountTutorial  
/04/30 16:20:12 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
for your platform... using builtin-java classes where applicable  
sheeha@ubuntu:~$ hadoop fs -mkdir /WordCountTutorial/Input  
/04/30 16:20:35 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
for your platform... using builtin-java classes where applicable  
sheeha@ubuntu:~$ hadoop fs -put '/home/sheeha/Desktop/WordCountTutorial/input_da  
/input.txt' /WordCountTutorial/Input  
/04/30 16:21:36 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
for your platform... using builtin-java classes where applicable  
sheeha@ubuntu:~$ cd /home/sheeha/Desktop/WordCountTutorial
```

Figure 8.1: Code screenshot of the commands to run Hadoop MapReduce in Linux terminal

A terminal window showing the execution of Hadoop MapReduce. The user is at the prompt 'sheeha@ubuntu: ~/Desktop/WordCountTutorial'. The commands are: 'javac -classpath \${HADOOP\_CLASSPATH} -d '/home/sheeha/Desktop/WordCountTutorial/tutorial\_classes' '/home/sheeha/Desktop/WordCountTutorial/WordCount.java'', 'jar -cvf firstTutorial.jar -C tutorial\_classes/ .', and 'hadoop jar '/home/sheeha/Desktop/WordCountTutorial/firstTutorial.jar' WordCount /WordCountTutorial/Input /WordCountTutorial/Output'. The output shows the addition of manifest and the classes being added to the jar.

```
sheeha@ubuntu: ~/Desktop/WordCountTutorial  
sheeha@ubuntu:~/Desktop/WordCountTutorial$ javac -classpath ${HADOOP_CLASSPATH}  
-d '/home/sheeha/Desktop/WordCountTutorial/tutorial_classes' '/home/sheeha/Desk  
op/WordCountTutorial/WordCount.java'  
sheeha@ubuntu:~/Desktop/WordCountTutorial$ jar -cvf firstTutorial.jar -C tutoria  
l_classes/ .  
added manifest  
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 754)(deflated 56%)  
adding: WordCount$IntSumReducer.class(in = 1739) (out= 739)(deflated 57%)  
adding: WordCount.class(in = 1491) (out= 814)(deflated 45%)  
sheeha@ubuntu:~/Desktop/WordCountTutorial$ hadoop jar '/home/sheeha/Desktop/Word  
CountTutorial/firstTutorial.jar' WordCount /WordCountTutorial/Input /WordCountTu  
torial/Output
```

Figure 8.2: Code screenshot of the commands to run Hadoop MapReduce in Linux terminal



```
sheeha@ubuntu: ~/Desktop/WordCountTutorial
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=150
File Output Format Counters
  Bytes Written=83
sheeha@ubuntu:~/Desktop/WordCountTutorial$ hadoop dfs -cat /WordCountTutorial/Output/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/30 16:29:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Bahcesehir      1
Istanbul        2
Jerusalem       2
Mohammed        4
Omar            1
Palestine       4
Sheeha          2
Shiha          1
sheeha@ubuntu:~/Desktop/WordCountTutorial$
```

Figure 8.3: Screenshot of the output after completing Hadoop MapReduce in Linux terminal

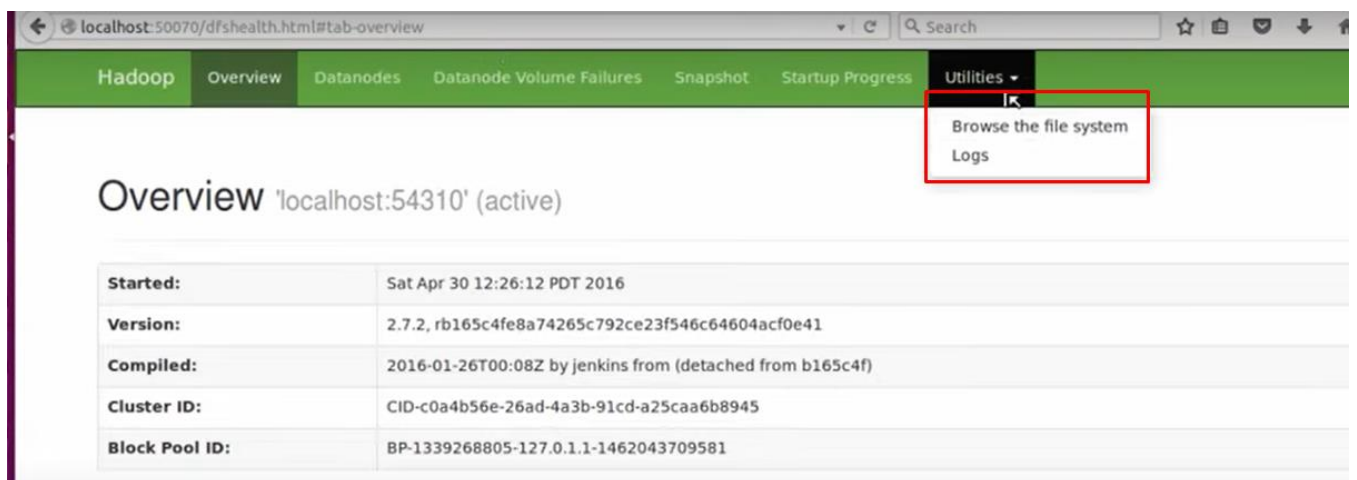


Figure 9.1: Screenshot of the localhost in browser

```

WordCountReducer.java
WordCount ▸ src ▸ com.wordcount.wc ▸ WordCountReducer ▸ reduce(Text, Iterable<IntWritable>, Context): void
1 package com.wordcount.wc;
2 import java.io.IOException;
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Reducer;
6 public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
7 {
8     private IntWritable count = new IntWritable();
9
10    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
11        InterruptedException
12    {
13
14        int valueSum = 0;
15        for (IntWritable val : values)
16        {
17            valueSum += val.get();
18        }
19        count.set(valueSum);
20        context.write(key, count);
21    }
22 }

```

Figure 10.1: Screenshot of the WordCountReducer.java program

```

package com.wordcount.wc;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.LongWritable;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
    private Text wordToken = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
    {
        StringTokenizer tokens = new StringTokenizer(value.toString()); //Dividing String into tokens
        while (tokens.hasMoreTokens())
        {
            wordToken.set(tokens.nextToken());
            context.write(wordToken, new IntWritable(1));
        }
    }
}

```

Figure 10.2: Screenshot of the WordCountMapper.java program

```

WordCount.java
WordCount ▸ src ▸ com.wordcount.wc ▸ WordCount ▸
1 import org.apache.hadoop.conf.Configuration;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Job;
6 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
7 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
8 import org.apache.hadoop.util.GenericOptionsParser;
9 public class WordCount
10 {
11     public static void main(String[] args) throws Exception
12     {
13         Configuration conf = new Configuration();
14         String[] pathArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
15         if (pathArgs.length < 2)
16         {
17             System.err.println("MR Project Usage: wordcount <input-path> [...] <output-path>");
18             System.exit(2);
19         }
20         Job wcJob = Job.getInstance(conf, "MapReduce WordCount");
21         wcJob.setJarByClass(WordCount.class);
22         wcJob.setMapperClass(WordCountMapper.class);
23         wcJob.setCombinerClass(WordCountReducer.class);
24         wcJob.setReducerClass(WordCountReducer.class);
25         wcJob.setOutputKeyClass(Text.class);
26         wcJob.setOutputValueClass(IntWritable.class);
27         for (int i = 0; i < pathArgs.length - 1; ++i)
28         {
29             FileInputFormat.addInputPath(wcJob, new Path(pathArgs[i]));
30         }
31         FileOutputFormat.setOutputPath(wcJob, new Path(pathArgs[pathArgs.length - 1]));
32         System.exit(wcJob.waitForCompletion(true) ? 0 : 1);
33     }
34 }

```

Figure 10.3: Screenshot of the WordCountDiver.java program

## **CASE STUDY ON CLOUD COMPUTING**

### **1. CASE STUDY ON AMAZON (AIM: To Understand The Services of Amazon Elastic Cloud.)**

**Ans:** This case study aims to explore the services provided by Amazon Elastic Compute Cloud (EC2) and understand its significance in the context of cloud computing. EC2 is a fundamental component of Amazon Web Services (AWS) and offers scalable compute resources to meet various business needs. By examining the features, benefits, and real-world applications of EC2, we can gain insights into how this service revolutionizes the way organizations build and deploy their applications. Here are some of the services provided by Amazon Elastic Cloud:

Resizable compute capacity: EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers.

Scalable deployment of applications: EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired.

Broadest and deepest compute platform: Amazon EC2 offers the broadest and deepest compute platform, with over 600 instances and choice of the latest processor, storage, networking, operating system, and purchase model to help you best match the needs of your workload.

Virtual computing environments: EC2 provides virtual computing environments known as instances, which can be launched as many or as few as needed, configured for security and networking, and managed for storage.

Integration with other AWS services: EC2 is designed for Amazon Web Services and works fine with Amazon services like Amazon S3, Amazon RDS, Amazon DynamoDB, and Amazon SQS.

#### Case Study: Company X's Migration to EC2:

- Company X's infrastructure challenges and requirements
- Evaluation of EC2's suitability for Company X's needs
- Migration process and implementation
- Benefits and outcomes achieved by Company X after adopting EC2.

Overall, Amazon Elastic Compute Cloud provides a scalable, secure, and cost-effective way to deploy applications in the cloud. It eliminates the need to invest in hardware up front, allowing developers to develop and deploy applications faster.

## 2. CASE STUDY ON AZURE (AIM: To Understand The Services of Microsoft AZURE)

**Ans:** Microsoft Azure is a public cloud computing platform that offers a wide range of cloud services, including compute, analytics, storage, and networking. Azure comprises more than 200 cloud services and supports varied operating systems, databases, and developer tools. Azure has a wide range of features that make it a powerful and versatile platform. These features include:

- Compute: Azure offers a variety of compute options, including virtual machines, containers, and serverless computing. This allows businesses to choose the right compute solution for their needs.
- Data storage: Azure offers a variety of data storage options, including relational databases, NoSQL databases, and blob storage. This allows businesses to store their data in the most efficient way possible.
- Data analytics: Azure offers a variety of data analytics services, including machine learning, big data analytics, and business intelligence. This allows businesses to gain insights from their data.
- Artificial intelligence: Azure offers a variety of artificial intelligence services, including machine learning, natural language processing, and computer vision. This allows businesses to build applications that can understand and interact with the world around them.

Benefits of Azure:

- Scalability and Elasticity: Ability to scale resources on-demand and handle varying workloads
- Global Reach: Azure's extensive network of data centers worldwide
- Security and Compliance: Robust security measures and compliance certifications
- Hybrid Capabilities: Seamless integration between on-premises and cloud environments
- Cost Optimization: Flexible pricing models, cost management tools, and cost-saving opportunities

Case Study: Organization X's Migration to Azure:

- Organization X's IT challenges and requirements
- Evaluation of Azure's suitability for Organization X's needs
- Migration process and implementation
- Benefits and outcomes achieved by Organization X after adopting Azure

By examining Microsoft Azure's services and capabilities through this case study, we can gain insights into its role as a comprehensive cloud computing platform. This analysis will showcase how Azure empowers organizations to innovate, scale, and achieve their digital transformation goals effectively.

### 3. CASE STUDY ON HADOOP (AIM: To Understand The Services of HADOOP.)

**Ans:** Hadoop is a software framework developed by the Apache Software Foundation for distributed storage and processing of huge amounts of datasets. Hadoop is an open-source software framework for distributed storage and processing of large data sets. It is designed to scale up to thousands of nodes, each of which can have terabytes or petabytes of storage. Hadoop is used by a wide variety of organizations, including financial institutions, telecommunications companies, and government agencies.

#### Hadoop Architecture and Components:

- Hadoop Distributed File System (HDFS): Overview of the distributed file system for storing data across a cluster
- MapReduce: Understanding the parallel processing framework for distributed data processing
- YARN: Resource management and job scheduling in Hadoop
- Hadoop EcoSystem: Introduction to additional components like Hive, Pig, HBase, and Spark.

#### Key Features and Benefits of Hadoop:

- Scalability and Fault Tolerance: Ability to handle large volumes of data and recover from failures
- Data Locality: Optimized data processing by moving computation close to the data
- Cost-Effectiveness: Utilizing commodity hardware for distributed processing
- Flexibility and Extensibility: Integration with various tools and frameworks for diverse use cases
- Data Processing Efficiency: Parallel execution of tasks on distributed nodes

#### Case Study: Company X's Implementation of Hadoop:

- Company X's data challenges and requirements
- Evaluation of Hadoop's suitability for Company X's needs
- Implementation process and integration with existing infrastructure
- Benefits and outcomes achieved by Company X after adopting Hadoop

Through this case study, we can gain a comprehensive understanding of Apache Hadoop and its services in distributed data processing. Hadoop is commonly used by data analysts to handle big data, and its market size continues to grow. It is a beneficial technology for data analysts as it can store and process enormous amounts of data at an extremely fast rate. Hadoop enhances operational decision-making and batch workloads for historical analysis by supporting real-time analytics.

#### **4. CASE STUDY ON ANEKA (AIM: To Understand The Services of Aneka Elastic Cloud.)**

**Ans:** This case study aims to explore the services provided by Aneka Elastic Cloud, a distributed cloud computing platform designed to enable the efficient utilization of resources across a network of computers. Aneka is an open-source cloud computing platform that supports the development and deployment of high-performance applications. It provides a unified programming model for different types of cloud resources, including virtual machines, containers, and Hadoop clusters. Aneka is used by a wide variety of organizations, including universities, research institutions, and businesses.

Aneka Architecture and Components:

- Resource Manager: Managing and allocating computing resources across a network
- Task Manager: Scheduling and executing application tasks on distributed resources
- Data Manager: Handling data distribution and storage across the network
- Programming Model: Support for developing distributed applications using familiar programming paradigms

Key Features and Benefits of Aneka:

- Elasticity and Scalability: Dynamic allocation and de-allocation of resources based on workload demands
- Resource Management: Efficient utilization of distributed resources for optimal performance
- Fault Tolerance: Redundancy and fault recovery mechanisms to ensure high availability
- Data Management: Efficient data distribution and storage across the distributed network
- Developer-Friendly: Support for various programming languages and frameworks

Case Study: Organization X's Implementation of Aneka Elastic Cloud:

- Organization X's computing challenges and requirements
- Evaluation of Aneka's suitability for Organization X's needs
- Implementation process and integration with existing infrastructure
- Benefits and outcomes achieved by Organization X after adopting Aneka

Through this case study, we can gain a comprehensive understanding of Aneka Elastic Cloud and its services in distributed cloud computing. The analysis will showcase how Aneka facilitates resource utilization, application deployment, and distributed computing in various domains, leading to improved efficiency and performance.

## 5. CASE STUDY ON GOOGLE APPS (AIM: To Understand The Services of Google App Engine.)

**Ans:** Google App Engine (GAE) is a cloud computing platform-as-a-service (PaaS) for developing and hosting web applications in Google-managed data centers. This case study aims to explore the services provided by Google App Engine, a fully managed platform that enables developers to build and deploy scalable web applications on Google Cloud. By examining the features, benefits, and real-world applications of Google App Engine, we can gain insights into its services and understand how it simplifies the process of developing and managing cloud-based applications.

### Google App Engine Architecture and Components:

- Application Environment: The runtime environment for deploying applications
- Datastore: A NoSQL database for storing application data
- Services and APIs: Integration with various Google Cloud services and third-party APIs
- Security and Identity: Authentication and access control mechanisms
- Deployment and Scaling: Automatic scaling and versioning of application instances

### Key Features and Benefits of Google App Engine:

- Serverless Infrastructure: Automatic provisioning and scaling of resources
- Easy Application Deployment: Simplified deployment and management of applications
- Built-in Services: Integration with various Google Cloud services for enhanced functionality
- Data Management: Scalable and reliable storage and querying capabilities with Datastore
- Security and Compliance: Robust security features and compliance certifications

### Case Study: Company X's Implementation of Google App Engine:

- Company X's application development challenges and requirements
- Evaluation of Google App Engine's suitability for Company X's needs
- Implementation process and integration with existing infrastructure
- Benefits and outcomes achieved by Company X after adopting Google App Engine

Through this case study, we can gain a comprehensive understanding of Google App Engine and its services in cloud application development. The analysis will showcase how Google App Engine simplifies the process of building and managing scalable web applications, enabling organizations to focus on their application logic while leveraging the benefits of Google Cloud infrastructure.

## **6. CASE STUDY ON GOOGLE APPS BUSINESS SOLUTION FOR DATA ACCESS & DATA UPLOAD (AIM: To Understand The business Solution Application of Google apps.)**

**Ans:** Google Apps is a suite of cloud-based productivity and collaboration tools that can be used by businesses of all sizes. Google Apps offers a number of features that make it easy to access and upload data.

### Google Apps Business Solutions:

- G Suite: Introduction to the suite of productivity and collaboration tools including Gmail, Google Drive, Google Docs, Google Sheets, and Google Slides
- Google Drive: File storage and synchronization, document management, and sharing capabilities
- Google Docs: Collaborative document editing and real-time collaboration features
- Google Sheets: Spreadsheet application for data analysis, manipulation, and sharing
- Google Forms: Creating and managing online forms for data collection and surveys

### Data Access with Google Apps:

- Seamless Data Accessibility: Anytime, anywhere access to files and documents through Google Drive
- Real-time Collaboration: Simultaneous editing and collaboration on documents with multiple stakeholders
- Access Control and Permissions: Granular control over sharing settings and access permissions to ensure data security

### Data Upload with Google Apps:

- File Upload and Synchronization: Uploading files to Google Drive and syncing across devices
- Bulk Data Import: Efficient upload of large volumes of data using Google Sheets or Google Forms
- Automation and Integration: Integration with third-party applications and APIs for automated data upload processes

### Case Study: Organization X's Implementation of Google Apps:

- Organization X's data access and upload challenges and requirements
- Evaluation of Google Apps' suitability for Organization X's needs
- Implementation process and integration with existing infrastructure
- Benefits and outcomes achieved by Organization X after adopting Google Apps

Through this case study, we can gain a comprehensive understanding of how Google Apps provides a business solution for efficient data access and streamlined data upload processes. The analysis will showcase how organizations leverage Google Apps' suite of tools to



enhance productivity, collaboration, and data management, ultimately improving business efficiency and effectiveness.

## **7. CASE STUDY ON CONTROL PANEL SOFTWARE MANAGER APPLICATION OF HYPERVISOR (AIM: To Understand The Applications of hypervisor.)**

**Ans:** A hypervisor is a piece of software that allows multiple operating systems to run on the same physical hardware. This is done by creating virtual machines (VMs), which are isolated from each other and from the host operating system. Hypervisors can be used to improve server utilization, reduce costs, and improve security. Control panel software managers are used to manage hypervisors. They provide a graphical user interface (GUI) that allows users to create and manage VMs, as well as perform other tasks such as assigning resources, monitoring performance, and troubleshooting problems.

### Hypervisor Technology and Virtualization:

- Understanding the concept of hypervisor and its role in virtual machine management
- Types of hypervisors: Type 1 (bare-metal) and Type 2 (hosted)
- Benefits of virtualization: Consolidation, resource optimization, and flexibility

### Control Panel Software Manager Application of Hypervisor:

- Architecture and Components: Overview of the control panel software manager built on top of the hypervisor
- Virtual Machine Provisioning: Creating and deploying virtual machines with predefined configurations
- Resource Allocation and Monitoring: Managing virtual machine resources (CPU, memory, storage) and monitoring performance
- Network Management: Configuring and managing virtual networks for seamless communication
- Security and Access Control: Implementing security measures and controlling user access to virtual machines
- Backup and Disaster Recovery: Enabling backup and recovery mechanisms for virtual machine data

### Key Features and Benefits:

- Centralized Management: A unified interface to manage multiple virtual machines and resources
- Resource Optimization: Efficient allocation and utilization of computing resources across virtual machines
- Scalability and Flexibility: Easy scaling of resources and adding/removing virtual machines as needed
- Automation and Orchestration: Automating routine tasks and streamlining operations
- Enhanced Security: Isolation and security measures to protect virtual machines and data.

### Case Study: Organization X's Implementation of Hypervisor-based Control Panel Software Manager:

- Organization X's virtualization challenges and requirements
- Evaluation of the hypervisor-based control panel software manager for Organization X's needs
- Implementation process and integration with existing infrastructure
- Benefits and outcomes achieved by Organization X after adopting the control panel software manager.

Through this case study, we can gain a comprehensive understanding of how a hypervisor-based control panel software manager enables efficient management and monitoring of virtualized environments. The analysis will showcase how organizations leverage this technology to streamline operations, optimize resource utilization, and enhance overall efficiency in their virtualized infrastructure.