R.M. K. COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

R.S.M. NAGAR, PUDUVOYAL – 601 206.
GUMMIDIPOONDI TK. THIRUVALLUR DIST.

Department **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

Laboratory **22CS401 – DISTRIBUTED AND CLOUD COMPUTING LABORATORY MANUAL**

Semester **IV**

Certified that this is a bonafide record work done by………................................... with

Roll / Reg. Number …………………..……………………….. He / She is a student of

…………………………………………………………………….………in the

R.M.K. COLLEGE OF ENGINEERING AND TECHNOLOGY, Puduvoyal.

Faculty-in-charge **Head of the Department**

External Examiner **Date:** **Internal Examiner**

RMK College of Engineering and Technology
(An Autonomous Institution)
RSM Nagar, Puduvoyal-601206

INSTITUTION VISION

To be knowledge hub of providing quality technical education and promoting research for building up of our nation and its contribution for the betterment of humanity.

INSTITUTION MISSION

- To make the best use of state-of-the-art infrastructure to ensure quality technical education.
- To develop industrial collaborations to promote innovation and research capabilities.
- To inculcate values and ethics to serve humanity

RMK College of Engineering and Technology

(An Autonomous Institution)

RSM Nagar, Puduvoyal-601206

Department of Artificial Intelligence and Data Science

VISION

To accomplish excellence in the field of Artificial Intelligence and Data Science through innovative research ideas to meet the societal needs.

MISSION

- To develop industry-ready graduates through state-of-the art infrastructure facilities.
- To inculcate high personal and professional values that benefit the society.
- To promote interest in higher studies, research and entrepreneurship to meet global challenges.

PROGRAM EDUCATIONAL OBJECTIVES

Graduates of Artificial Intelligence and Data Science Program will

PEO I: Work effectively in inter-disciplinary field with the knowledge of Artificial Intelligence and Data Science to develop appropriate solutions to the real-world problems.

PEO II: Excel in professional career and pursue higher education in the field of Artificial Intelligence and Data Science.

PEO III: Apply their knowledge to the technological revolution through life-long learning.

PEO IV: Excel as socially committed engineers or entrepreneurs with high ethical and moral values.

PROGRAM SPECIFIC OUTCOMES

Graduates of Artificial Intelligence and Data Science Program will be able to:

- Apply fundamental concepts of Artificial Intelligence and Data Science to solve technical problems.
- Utilize Artificial Intelligence and Data Science tools to provide innovative business solutions.
- Implement the domain knowledge to achieve successful career as an employee, entrepreneur and an engineering professional.

RMK College of Engineering and Technology
(An Autonomous Institution)
RSM Nagar, Puduvoyal-601206

Department of Artificial Intelligence and Data Science

PROGRAM OUTCOME

- **Engineering Knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
- **Problem Analysis:** Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
- **Design/ Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
- **Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.
- **Modern Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **The Engineer and Society:** Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.
- **Environment and Sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.
- **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.
- **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
- **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
- **Project Management and Finance:** Demonstrate knowledge and understanding of engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **Life-long Learning:** Recognize the need for and have the preparation and ability to Engage in independent and life- long learning in the broadest context of technological Change.

RMK College of Engineering and Technology

(An Autonomous Institution)

RSM Nagar, Puduvoyal-601206

Department of Artificial Intelligence and Data Science

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab observation notebook, and get certified by the concerned faculty.

6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.

8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department

SYLLABUS

22CS401	DISTRIBUTED AND CLOUD COMPUTING LABORATORY	L	T	P	C
		0	0	4	2

OBJECTIVES:

- Articulate the concepts and models underlying distributed computing.
- Maintain consistency and perform efficient coordination in distributed systems through the use of logica global states, and snapshot recording algorithms.
- Learn different distributed mutual exclusion algorithms.
- Develop the ability to understand the cloud infrastructure and virtualization that help in the development cloud.
- Explain the high-level automation and orchestration systems that manage the virtualized infrastructure.

LIST OF EXERCISES:

1. Implement a simple distributed program that communicates between two nodes using Java's RMI (Remote Method Invocation) API.
2. Develop a distributed program that uses Java's messaging API (JMS) to communicate between nodes. Explore the different messaging paradigms (pub/sub, point-to-point) and evaluate their performance and scalability.
3. Develop a model of a distributed program using Java's concurrency and synchronization primitives.
4. Develop a program in Java that implements vector clocks to synchronize the order of events between nodes in a distributed system.
5. Implement a snapshot algorithm for recording the global state of the distributed system using vector clocks, for both FIFO and non-FIFO channels. Test the algorithm by recording snapshots at various points in the system's execution and analyzing the resulting global state.
6. Implement Lamport's algorithm for mutual exclusion in a distributed system using Java's RMI API.
7. Develop a program in Java that implements Maekawa's algorithm for mutual exclusion in a distributed system.
8. Set up a virtualized data center using a hypervisor like VMware or VirtualBox and create multiple virtual machines (VMs) on it. Configure the VMs with different operating systems, resources, and network configurations, and test their connectivity and performance.
9. Deploy a containerized application on a virtual machine using Docker or Kubernetes.
10. Set up and configure a single-node Hadoop cluster.
11. Run the word count program in Hadoop.
12. Deploy a microservices architecture using a container orchestration tool like Kubernetes or Docker Swarm.

TOTAL: 60 PERIODS

INDEX

Ex. No: 1 (a) **RMI FOR SENDING MESSAGE**

Date: 10.01.2024

AIM:

To implement a simple distributed program that communicates between two nodes by sending a message using Java's RMI (Remote Method Invocation) API.

STEPS:

1. Create a folder Hello in either D:\ drive or E:\ drive
2. Write all the 3 programs
3. Set path=C:\Program Files\Java\JDK1.2\bin
4. Compile all programs javac*.java
5. Start rmi registry (in a separate command Prompt)
6. Run the server java HelloServer (in a separate command prompt)
7. Run the client java Hello Client (in a separate command prompt)

   Note: for each prompt get inside the folder and set path= C:\Program Files\Java\jdk-21\bin

PROGRAMS:

<u>HelloClient.java (Client Program)</u>

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.Naming;
public class HelloClient {
   public static void main(String[] args) {
         try {
                 Hello stub = (Hello) Naming.lookup("rmi://localhost:5000/hello");
                 String response = stub.sayHello();
                 System.out.println("Response: " + response);
               } catch (Exception e) {
                  System.err.println("Client exception: " + e.toString());
                  e.printStackTrace();
               }
            }
         }
```

HelloServer.java (Server Program)

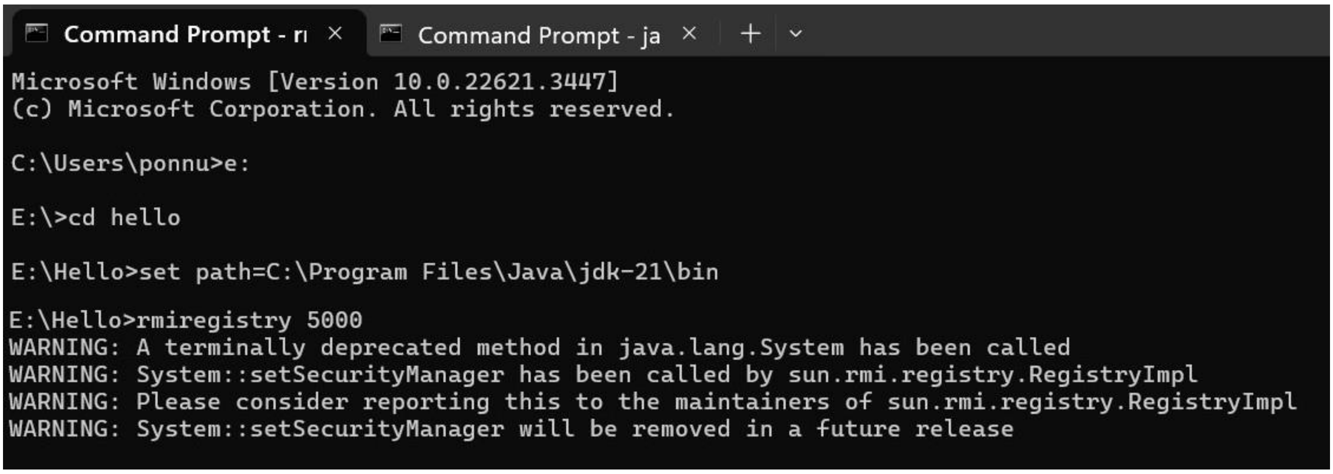
```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class HelloServer extends UnicastRemoteObject implements Hello {
    protected HelloServer() throws RemoteException {
        super();
    }
    public String sayHello() {
        return "Hello, world!";
    }
    public static void main(String args[]) {
        try {
            Hello stub = new HelloServer();
            Naming.rebind("rmi://localhost:5000/hello", stub);
            System.out.println("Server ready");
        } catch (Exception e) {
            System.err.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Hello.java (Interface)

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Hello extends Remote {
    String sayHello() throws RemoteException;
}
```

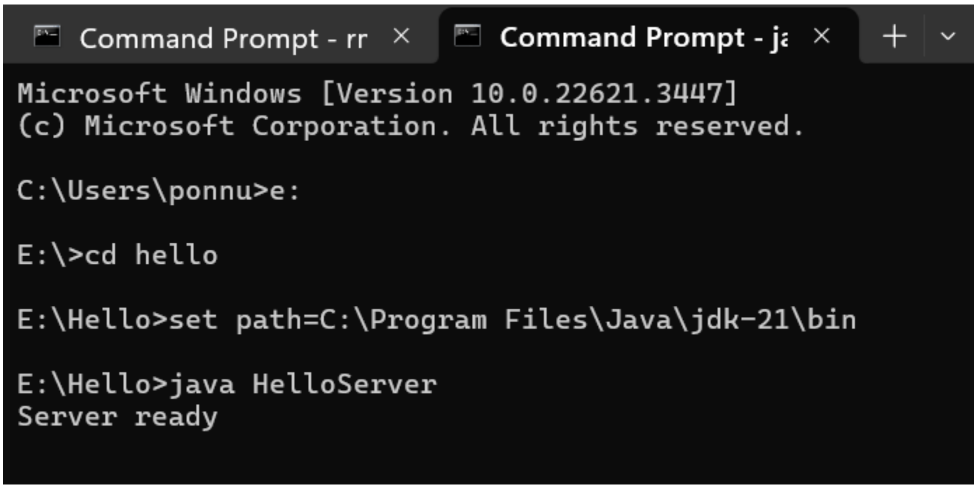
OUTPUTS:

Starting RMI registry at localhost 5000



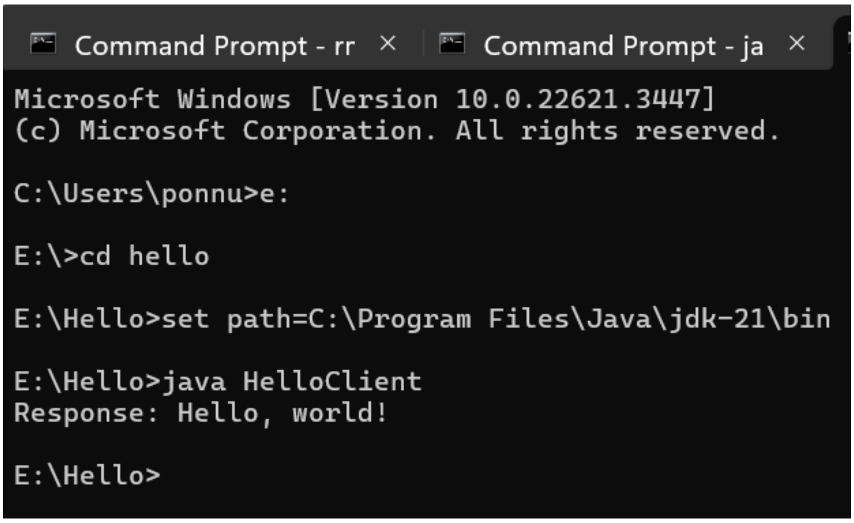
```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>cd hello

E:\Hello>set path=C:\Program Files\Java\jdk-21\bin

E:\Hello>rmiregistry 5000
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

Server-side



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>cd hello

E:\Hello>set path=C:\Program Files\Java\jdk-21\bin

E:\Hello>java HelloServer
Server ready
```

Client-side



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>cd hello

E:\Hello>set path=C:\Program Files\Java\jdk-21\bin

E:\Hello>java HelloClient
Response: Hello, world!

E:\Hello>
```

RESULT:

Thus the experiment has been executed successfully.

Ex. No: 1 (b) **RMI FOR ADDING TWO NUMBERS**

Date: 10.01.2024

AIM:

To implement a simple distributed program that communicates between two nodes by adding two numbers using Java's RMI (Remote Method Invocation) API.

STEPS:

1. Create a folder Adder in either D:\ drive or E:\ drive
2. Write all the 3 programs
3. Set path=C:\Program Files\Java\JDK1.2\bin
4. Compile all programs javac*.java
5. Start rmi registry (in a separate command Prompt)
6. Run the server java AdderServer (in a separate command prompt)
7. Run the client java AdderClient (in a separate command prompt)

   Note: for each prompt get inside the folder and set path= C:\Program Files\Java\jdk-21\bin

PROGRAMS:

AdderClient.java (Client)

```
import java.rmi.*;
public class AdderClient {
    public static void main(String args[]) {
        try {
            Adder stub = (Adder) Naming.lookup("rmi://localhost:2000/Adderservice");
            System.out.println(stub.adder(34, 4));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

AdderRemote.java (Server)

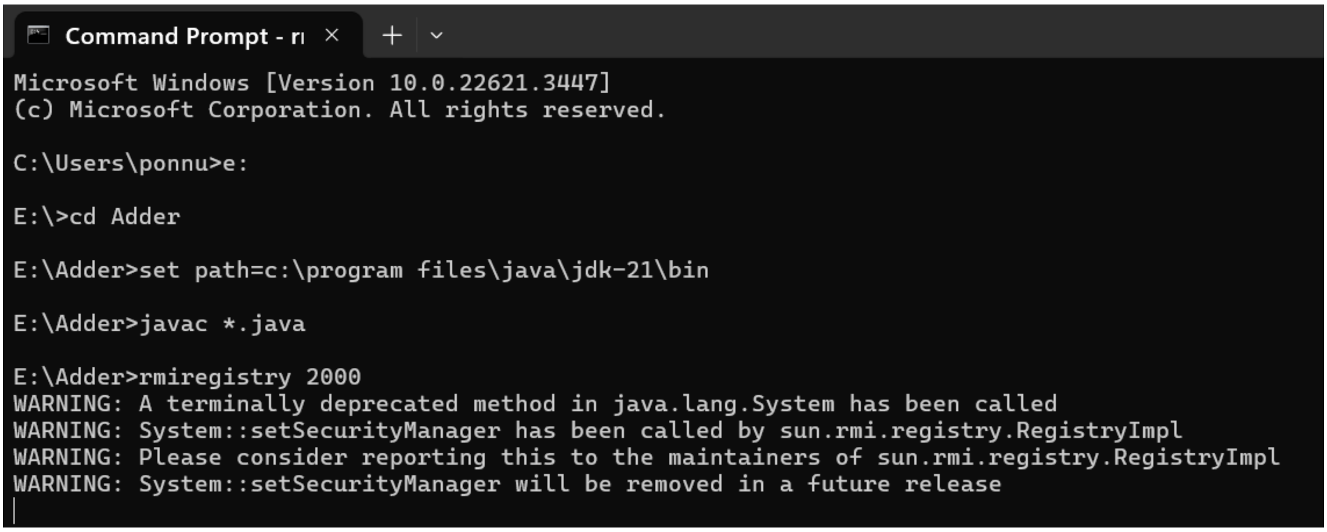
```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class AdderRemote extends UnicastRemoteObject implements Adder {
    AdderRemote() throws RemoteException {
        super();
    }
    public int adder(int x, int y)
   {
        return x + y;
    }
    public static void main(String args[]) {
        try {
            Adder stub = new AdderRemote();
            Naming.rebind("rmi://localhost:2000/Adderservice", stub);
        } catch (Exception e) {
            System.err.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Adder.java (interface)

```
 import java.rmi.Remote;
 import java.rmi.RemoteException;
 public interface Adder extends Remote {
     public int adder(int x, int y) throws RemoteException;
 }
```

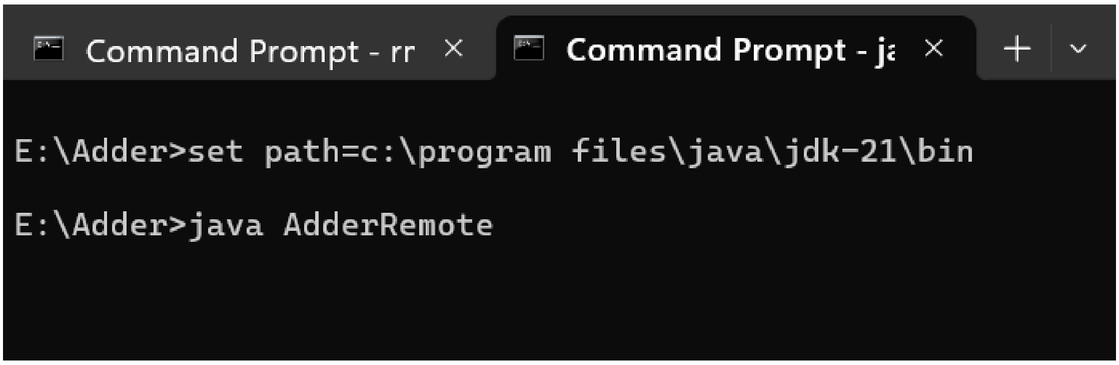
OUTPUTS:

Starting RMI registry at localhost 2000



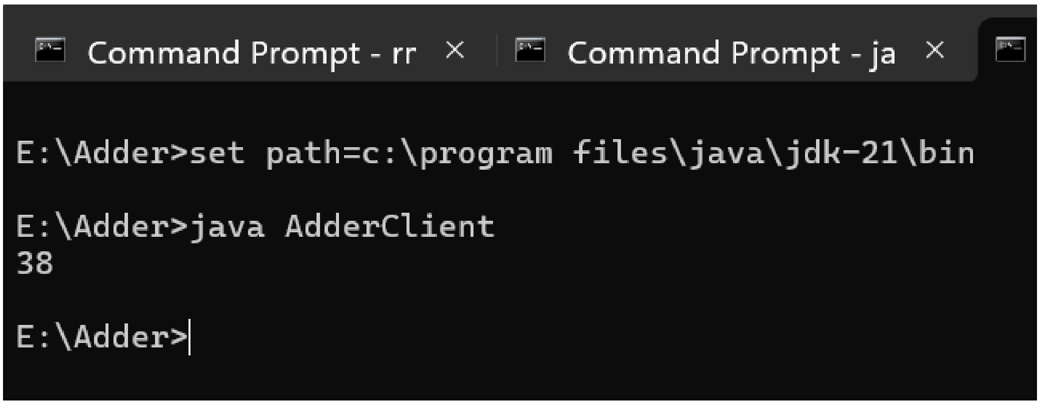
```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>cd Adder

E:\Adder>set path=c:\program files\java\jdk-21\bin

E:\Adder>javac *.java

E:\Adder>rmiregistry 2000
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

AdderRemote (Server-side)



```
E:\Adder>set path=c:\program files\java\jdk-21\bin

E:\Adder>java AdderRemote
```

AdderClient (Client-side)



```
E:\Adder>set path=c:\program files\java\jdk-21\bin

E:\Adder>java AdderClient
38

E:\Adder>
```

RESULT:

Thus the experiment has been executed successfully.

Ex. No: 2 **CONCURRENCY AND SYNCHRONIZATION**

Date: 24.01.2024

AIM:

To develop a model of distributed program using Java's concurrency and synchronization primitives.

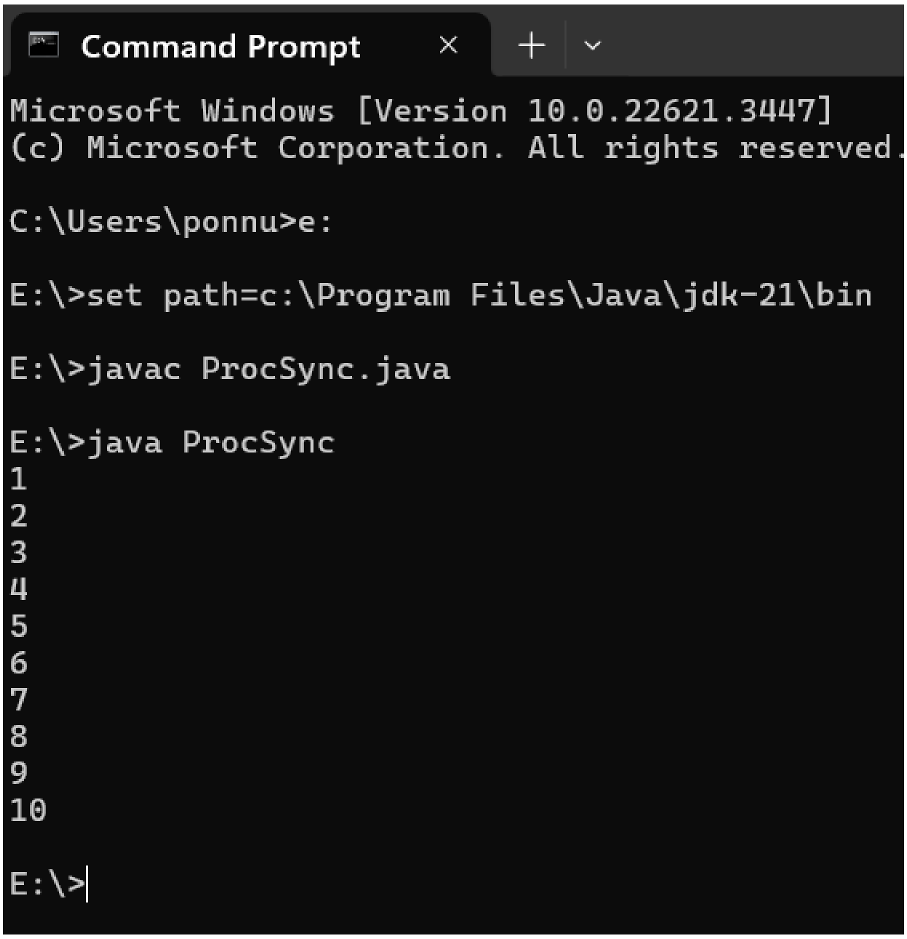
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin
2. Create a class named NumberPrinter that extends the Thread.
3. Create a for loop with i ranging from 1 to 10, using the try..catch block pause the printing value by 1 second to display the values.
4. Create a class ProcSync and call the NumberPrinter thread class and start the thread.
5. Compile the program using the javac command
6. Run and execute the program

PROGRAM:

```
class NumberPrinter extends Thread {
    public void run() {
        for (int i = 1; i <=10 ; i++) {
            System.out.println(i);
            try {
                Thread.sleep(1000); // Pausing for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class ProcSync {
    public static void main(String args[]) {
        NumberPrinter t1 = new NumberPrinter(); // Creating the thread
        t1.start(); // Starting the thread
    }
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>set path=c:\Program Files\Java\jdk-21\bin

E:\>javac ProcSync.java

E:\>java ProcSync
1
2
3
4
5
6
7
8
9
10

E:\>
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 3 **MESSAGING API (JMS) COMMUNICATION BETWEEN NODES**
Date: 31.01.2024

AIM:

To develop a distributed program that uses Java's messaging API (JMS) to communicate between nodes.

STEPS AND REQUIREMENTS:

- Download : ActiveMQ 5.18.3 (Oct 25th, 2023)
- Website : https://activemq.apache.org/components/classic/download/
- Java : All Java 11 and above (JavaSE16 used by me)

Steps:

1. Download the Active Mq (5.18.3) and extract it into a folder. Note the path to this folder.
2. Open Eclipse IDE
3. Create two Java projects JmsConsumer and JmsProducer.

3.1) **JmsConsumer**

1. Create Two Classes ConsumerMain and TopicConsumer
2. The two classes are given in the Drive Link

3.2) **JmsProducer**

3. Create Two Classes ProducerMain and TopicProducer
4. The two classes are given in the Drive Link

4) Now we need to add required jar files to both the projects (JmsConsumer and JmsProducer)

4.1) Right click on the **Project name** > **Build Path** > **Configure Build Path…**

4.2) Go to the **Libraries** tab > Select **Classpath** > then Click **Add External JARs**

4.3) Now Open the folder where we have extracted the ActiveMQ file.

4.4) Select the **activemq-all-5.18.3.jar** file and attach it.

Note:

If any logging unavailable error (Sometimes) is shown then we need to attach additional jars.

For that we can get the needed jars from the ActiveMQ folder itself.

Follow apache-activemq-5.18.3/lib/optional location and attach all log4j-xxx.jar files

5) Start the terminal and change the directory to the bin folder in ActiveMQ extracted folder

```
~$ cd ./apache-activemq-5.18.3/bin
```

6) Now we need to start the ActiveMQ broker type use

```
~$ ./activemq start
```

7) Now open new terminals as many as you want to start the Consumer program

8) Now open new terminal to start the Producer program

9) Run Consumer application

10) Run Producer application

PROGRAM:

```
package publisher;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.DeliveryMode;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;
import org.apache.activemq.ActiveMQConnectionFactory;
public class Publisher {
        public static void main(String[] args) {
        try {
        ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory("tcp://localhost:61616");
        Connection connection = connectionFactory.createConnection();
        connection.start();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer producer = session.createProducer(session.createTopic("MyTopic"));
        producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
        for (int i = 0; i < 10; i++) {
                TextMessage message = session.createTextMessage("Message " + i);
                producer.send(message);
                System.out.println("Sent message: " + message.getText());
        }
        connection.close();
        } catch (Exception e) {
        e.printStackTrace();
        }
        }
}
package subscriber;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.TextMessage;
import org.apache.activemq.ActiveMQConnectionFactory;
public class Subscriber {
        public static void main(String[] args) {
        try {
        ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory("tcp://localhost:61616");
        Connection connection = connectionFactory.createConnection();
        connection.start();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = session.createConsumer(session.createTopic("MyTopic"));
        while (true) {
                Message message = consumer.receive();
                if (message instanceof TextMessage) {
                TextMessage textMessage = (TextMessage) message;
                System.out.println("Received message: " + textMessage.getText());
                }
        }
        } catch (Exception e) {
        e.printStackTrace();
        }
        }
}
```

OUTPUT:

```
[Sent Message] : Value : 0
[Sent Message] : Value : 1
[Sent Message] : Value : 2
[Sent Message] : Value : 3
[Sent Message] : Value : 4
[Sent Message] : Value : 5
[Sent Message] : Value : 6
[Sent Message] : Value : 7
[Sent Message] : Value : 8
[Sent Message] : Value : 9

[Content Received] : Value 0
[Content Received] : Value 1
[Content Received] : Value 2
[Content Received] : Value 3
[Content Received] : Value 4
[Content Received] : Value 5
[Content Received] : Value 6
[Content Received] : Value 7
[Content Received] : Value 8
[Content Received] : Value 9
Finished
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 4 **IMPLEMENTATION OF VECTOR CLOCKS**

Date: 07.02.2024

AIM: To develop a program in Java that implements vector clocks to synchronize the order of events between nodes in a distributed system.

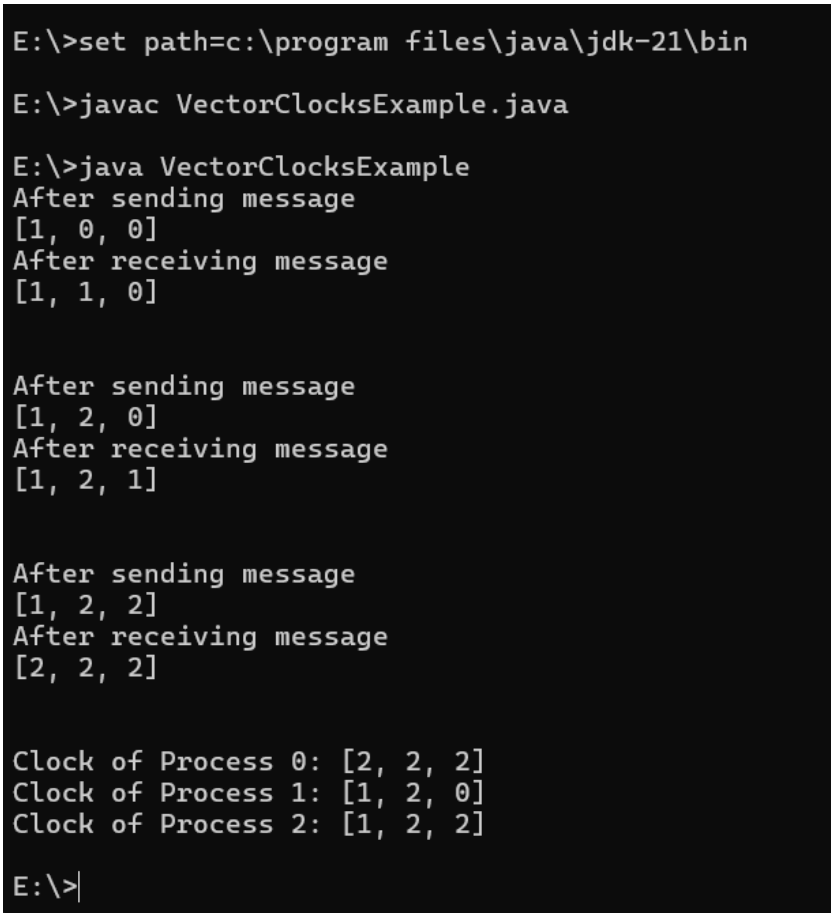
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin.
2. Create a class named VectorClocksExample that creates and initializes vector clocks for 3 processes.
3. Define a SendMessage() function that simulates message sending and receiving functionalities for the 3 processes.
4. Print the final state of the vector clocks.
5. Compile the program using the javac command.
6. Run and execute the program.

PROGRAM:

```
import java.util.Arrays;
public class VectorClocksExample {
    public static void main(String[] args) {
        // Create and initialize vector clocks for 3 processes
        int numberOfProcesses = 3;
        VectorClock[] clocks = new VectorClock[numberOfProcesses];
        for (int i = 0; i < numberOfProcesses; i++) {
            clocks[i] = new VectorClock(numberOfProcesses, i);
        }

        // Simulate message sending and receiving
        sendMessage(clocks[0], clocks[1]); // Process 0 sends a message to Process 1
        sendMessage(clocks[1], clocks[2]); // Process 1 sends a message to Process 2
        sendMessage(clocks[2], clocks[0]); // Process 2 sends a message to Process 0

        // Print final state of vector clocks
        for (int i = 0; i < numberOfProcesses; i++) {
            System.out.println("Clock of Process " + i + ": " + clocks[i]);
        }
    }
 private static void sendMessage(VectorClock sender, VectorClock receiver) {
        sender.increment();
        System.out.println("After sending message");
        System.out.println(sender);
        VectorClock messageClock = new VectorClock(sender);
        receiver.increment();
        receiver.receive(messageClock);
        System.out.println("After receiving message");
        System.out.println(receiver);
        System.out.println("\n");
    }
static class VectorClock {
        private int[] clock;
        private int processId;

        public VectorClock(int size, int processId) {
            this.clock = new int[size];
            this.processId = processId;
        }

        public VectorClock(VectorClock other) {
            this.clock = Arrays.copyOf(other.clock, other.clock.length);
            this.processId = other.processId;
        }

        public void increment() {
          //System.out.println(processId);
            clock[processId]++;
        }
     public void receive(VectorClock messageClock) {
            for (int i = 0; i < clock.length; i++) {
                clock[i] = Math.max(clock[i], messageClock.clock[i]);
            }
        }

        public String toString() {
            return Arrays.toString(clock);
        }
    }
}
```

OUTPUT:



```
E:\>set path=c:\program files\java\jdk-21\bin

E:\>javac VectorClocksExample.java

E:\>java VectorClocksExample
After sending message
[1, 0, 0]
After receiving message
[1, 1, 0]


After sending message
[1, 2, 0]
After receiving message
[1, 2, 1]


After sending message
[1, 2, 2]
After receiving message
[2, 2, 2]


Clock of Process 0: [2, 2, 2]
Clock of Process 1: [1, 2, 0]
Clock of Process 2: [1, 2, 2]

E:\>
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 5 **RECORD THE GLOBAL STATE IN FIFO CHANNEL**

Date: 28.02.2024

AIM: To implement a snapshot algorithm for recording the global state of the distributed system using vector clocks, for FIFO channels using Lamport's Snapshot Algorithm.

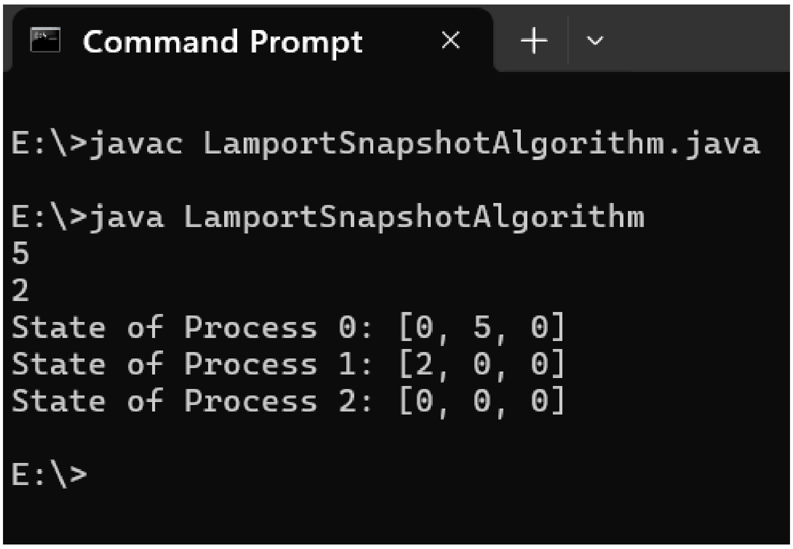
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin.
2. Create a public class named LamportSnapshotAlgorithm that creates and initializes vector clocks for 3 processes.
3. Define a Process() method under Process class that is used for recording the processes and printing the state of the processes.
4. Check if all the processes are completed and print them, otherwise call the pending processes and execute until it becomes empty.
5. Compile the program using the javac command.
6. Run and execute the program.

PROGRAM:

```
import java.util.*;

class Process {
    private int id;
    private int[] state;
    private Queue<Message> pendingQueue;
    private boolean recording;

    Process(int id, int numProcesses) {
        this.id = id;
        this.state = new int[numProcesses];
        this.pendingQueue = new LinkedList<>();
        this.recording = false;
    }

    public void receive(Message message) {
        if (recording) {
            state[message.sender] = message.value;
System.out.println(state[message.sender]);
        } else {
            pendingQueue.add(message);
        }
    }

    public void startRecording() {
        recording = true;
    }

    public void stopRecording() {
        recording = false;
        processPendingMessages();
    }

    public void processPendingMessages() {
        while (!pendingQueue.isEmpty()) {
            Message message = pendingQueue.poll();
            state[message.sender] = message.value;
        }
    }

    public void printState() {
        System.out.println("State of Process " + id + ": " + Arrays.toString(state));
    }
}

class Message {
    int sender;
    int value;

    Message(int sender, int value) {
        this.sender = sender;
        this.value = value;
    }
}

public class LamportSnapshotAlgorithm {
    public static void main(String[] args) {
        int numProcesses = 3;
        Process[] processes = new Process[numProcesses];

        // Create processes
        for (int i = 0; i < numProcesses; i++) {
            processes[i] = new Process(i, numProcesses);
        }

        // Simulate events
        processes[0].startRecording(); // Process 0 initiates snapshot recording
        processes[0].receive(new Message(1, 5));
processes[1].startRecording();
        processes[1].receive(new Message(0, 2));
processes[1].stopRecording();
        processes[0].stopRecording(); // Process 0 stops snapshot recording

        // Print state of all processes after recording
        for (Process process : processes) {
            process.printState();
        }
    }
}
```

OUTPUT:



```
E:\>javac LamportSnapshotAlgorithm.java

E:\>java LamportSnapshotAlgorithm
5
2
State of Process 0: [0, 5, 0]
State of Process 1: [2, 0, 0]
State of Process 2: [0, 0, 0]

E:\>
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 6 **RECORD THE GLOBAL STATE IN NON-FIFO CHANNEL**

Date: 07.03.2024

AIM: To implement a snapshot algorithm for recording the global state of the distributed system using vector clocks, for Non-FIFO channels.

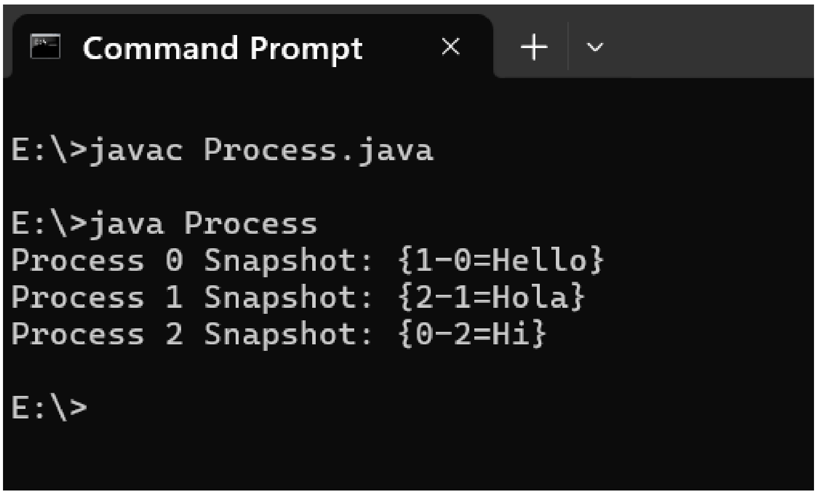
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin.
2. Create a class named Process that creates and initializes vector clocks and markers.
3. Define a Process() method under Process class that is used for recording the processes and e printing the state of the processes and set the Process ID, no. of processes, set the markers to receive and the snapshot of the same.
4. Check if the processes equals the Markers then print them, otherwise call the pending processes and execute until it becomes empty.
5. Compile the program using the javac command.
6. Run and execute the program.

PROGRAM:

```
import java.util.*;
class Process {
    private int processId;
    private int numProcesses;
    private int[] vectorClock;
    private Map<Integer, Queue<Message>> channels;
    private Map<String, String> localState;
    private Map<String, String> snapshot;
    private Set<Integer> receivedMarkers;

    public Process(int processId, int numProcesses) {
        this.processId = processId;
        this.numProcesses = numProcesses;
        this.vectorClock = new int[numProcesses];
        this.channels = new HashMap<>();
        this.localState = new HashMap<>();
        this.snapshot = new HashMap<>();
        this.receivedMarkers = new HashSet<>();

        // Initialize channels
        for (int i = 0; i < numProcesses; i++) {
            channels.put(i, new LinkedList<>());
        }
    }

    public void sendMessage(int destProcess, String message) {
        int[] timestamp = Arrays.copyOf(vectorClock, vectorClock.length);
        timestamp[processId]++;
        channels.get(destProcess).add(new Message(message, timestamp));
    }

    public void receiveMessages() {
        for (Map.Entry<Integer, Queue<Message>> entry : channels.entrySet()) {
            int srcProcess = entry.getKey();
            Queue<Message> channel = entry.getValue();
            for (Message msg : channel) {
                if (!msg.getMessage().equals("MARKER")) {
                    // Update local state with non-marker messages
                    vectorClock = Message.max(vectorClock, msg.getTimestamp());
                    localState.put(srcProcess + "-" + processId, msg.getMessage());
                } else {
                    // Handle marker messages
                    vectorClock = Message.max(vectorClock, msg.getTimestamp());
                    receivedMarkers.add(srcProcess);
                }
            }
        }
    }

    public void initiateSnapshot() {
        // Clear snapshot and receivedMarkers set
        snapshot.clear();
        receivedMarkers.clear();

        // Save local state
        snapshot.putAll(localState);

        // Send marker messages to all processes
        int[] markerTimestamp = Arrays.copyOf(vectorClock, vectorClock.length);
        markerTimestamp[processId]++;
        for (int destProcess = 0; destProcess < numProcesses; destProcess++) {
            channels.get(destProcess).add(new Message("MARKER", markerTimestamp));
        }

        // Receive messages until a marker message is received from each process
        while (receivedMarkers.size() < numProcesses) {
            receiveMessages();
        }

        // Update the snapshot with the received non-marker messages
        for (Map.Entry<Integer, Queue<Message>> entry : channels.entrySet()) {
            int srcProcess = entry.getKey();
            Queue<Message> channel = entry.getValue();
            for (Message msg : channel) {
                if (!msg.getMessage().equals("MARKER")) {
                    snapshot.put(srcProcess + "-" + processId, msg.getMessage());
                }
            }
        }

        // Print the snapshot
        System.out.println("Process " + processId + " Snapshot: " + snapshot);
    }

    public static void main(String[] args) {
        int numProcesses = 3;
        Process[] processes = new Process[numProcesses];

        for (int i = 0; i < numProcesses; i++) {
            processes[i] = new Process(i, numProcesses);
        }

        // Simulate some communication
        processes[0].sendMessage(1, "Hello");
        processes[2].sendMessage(0, "Hi");
        processes[1].sendMessage(2, "Hola");

        // Initiate snapshots
        for (Process process : processes) {
            process.initiateSnapshot();
        }
    }
}

class Message {
    private String message;
    private int[] timestamp;
    public Message(String message, int[] timestamp) {
        this.message = message;
        this.timestamp = timestamp;
    }

    public String getMessage() {
        return message;
    }

    public int[] getTimestamp() {
        return timestamp;
    }
    public static int[] max(int[] arr1, int[] arr2) {
        int[] result = new int[arr1.length];
        for (int i = 0; i < arr1.length; i++) {
            result[i] = Math.max(arr1[i], arr2[i]);
        }
        return result;
    }
}
```

OUTPUT:



```
E:\>javac Process.java

E:\>java Process
Process 0 Snapshot: {1-0=Hello}
Process 1 Snapshot: {2-1=Hola}
Process 2 Snapshot: {0-2=Hi}

E:\>
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 7 **LAMPORT'S MUTUAL EXCLUSION ALGORITHM**

Date: 11.03.2024

AIM: To implement the Lamport's Mutual Exclusion algorithm in the distributed system.

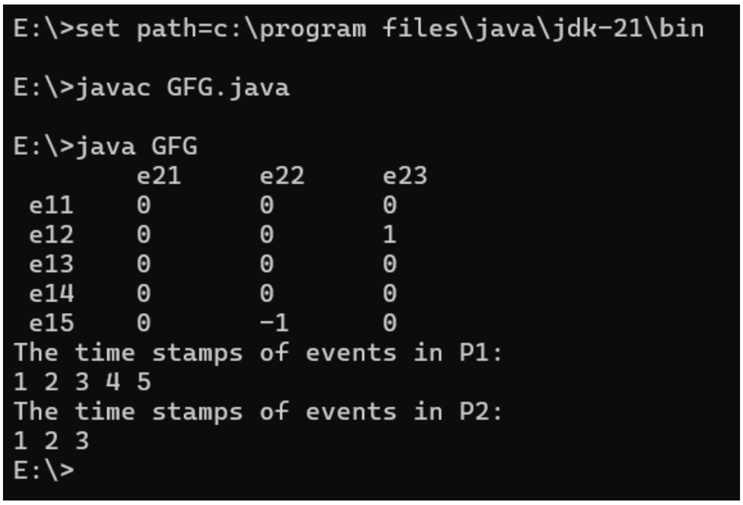
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin.
2. Create a public class named GFG that defines a function to find the maximum timestamp between 2 events and define another function to display the logical timestamp of events.
3. Define a function to find the timestamp of events.
4. Change the timestamp if the message if sent and received.
5. Display the timestamps of the processes.
6. Compile the program using the javac command.
7. Run and execute the program.

PROGRAM:

```
import java.util.*;
public class GFG {
   // Function to find the maximum timestamp
  // between 2 events
  static int max1(int a, int b)
  {
    // Return the greatest of the two
    if (a > b)
      return a;
    else
      return b;
  }

  // Function to display the logical timestamp
  static void display(int e1, int e2, int p1[], int p2[])
  {
    int i;
    System.out.print(
    "\nThe time stamps of events in P1:\n");
  for (i = 0; i < e1; i++) {
  System.out.print(p1[i] + " ");
  }
  System.out.println( "\nThe time stamps of events in P2:");
  // Print the array p2[]
  for (i = 0; i < e2; i++)
    System.out.print(p2[i] + " ");
}

// Function to find the timestamp of events
static void lamportLogicalClock(int e1, int e2,
                                int m[][])
{
  int i, j, k;
  int p1[] = new int[e1];
  int p2[] = new int[e2];
  // Initialize p1[] and p2[]
  for (i = 0; i < e1; i++)
  p1[i] = i + 1;

  for (i = 0; i < e2; i++)
    p2[i] = i + 1;
  for (i = 0; i < e2; i++)
    System.out.print("\te2" + (i + 1));

  for (i = 0; i < e1; i++) {
    System.out.print("\n e1" + (i + 1) + "\t");
    for (j = 0; j < e2; j++)
    System.out.print(m[i][j] + "\t");
  }

  for (i = 0; i < e1; i++) {
    for (j = 0; j < e2; j++) {
        // Change the timestamp if the
        // message is sent
        if (m[i][j] == 1) {
          p2[j] = max1(p2[j], p1[i] + 1);
          for (k = j + 1; k < e2; k++)
          p2[k] = p2[k - 1] + 1;
        }

        // Change the timestamp if the
        // message is received
        if (m[i][j] == -1) {
          p1[i] = max1(p1[i], p2[j] + 1);
          for (k = i + 1; k < e1; k++)
          p1[k] = p1[k - 1] + 1;
        }
      }
    }

    // Function Call
    display(e1, e2, p1, p2);
  }

  public static void main(String args[])
  {
    int e1 = 5, e2 = 3;
    int m[][] = new int[5][3];
    // message is sent and received
    // between two process

    /*dep[i][j] = 1, if message is sent
                   from ei to ej
        dep[i][j] = -1, if message is received
                   by ei from ej
        dep[i][j] = 0, otherwise*/
      m[0][0] = 0;
      m[0][1] = 0;
      m[0][2] = 0;
      m[1][0] = 0;
      m[1][1] = 0;
      m[1][2] = 1;
      m[2][0] = 0;
      m[2][1] = 0;
      m[2][2] = 0;
      m[3][0] = 0;
      m[3][1] = 0;
      m[3][2] = 0;
      m[4][0] = 0;
      m[4][1] = -1;
      m[4][2] = 0;

      // Function Call
      lamportLogicalClock(e1, e2, m);
    }
  }
```

OUTPUT:



```
E:\>set path=c:\program files\java\jdk-21\bin

E:\>javac GFG.java

E:\>java GFG
        e21     e22     e23
 e11    0       0       0
 e12    0       0       1
 e13    0       0       0
 e14    0       0       0
 e15    0       -1      0
The time stamps of events in P1:
1 2 3 4 5
The time stamps of events in P2:
1 2 3
E:\>
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 8 **MAEKAWA'S MUTUAL EXCLUSION ALGORITHM**

Date: 13.03.2024

AIM: To implement the Maekawa's Mutual Exclusion algorithm in the distributed system.

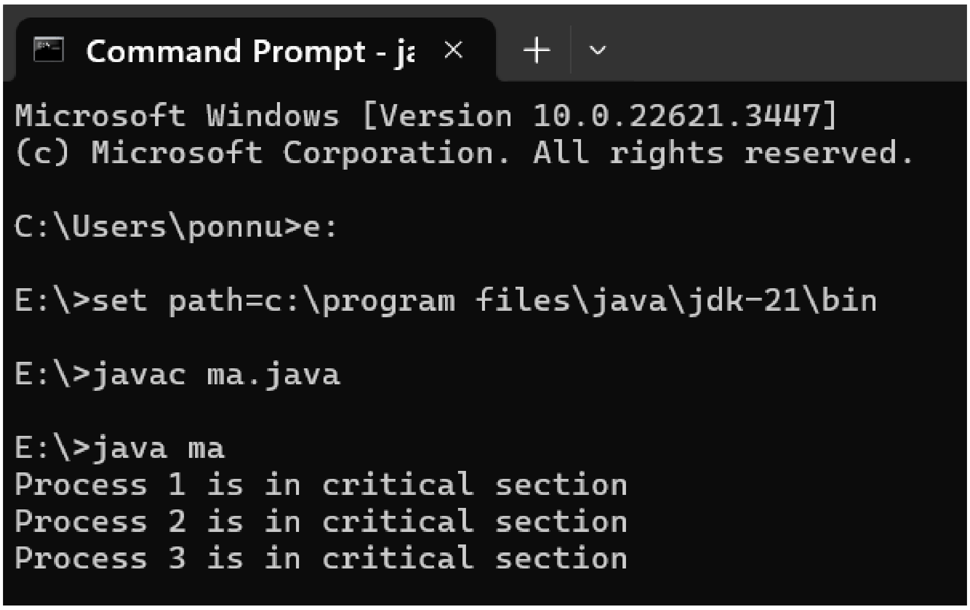
STEPS:

1. Set path=C:\Program Files\Java\JDK1.2\bin.
2. As Maekawa's algorithm is a quorum-based algorithm, we set the message type of enumerated data type with messages – REQUEST, REPLY, RELEASE
3. Create a public class named **ma** and set the quorums for the Processes.
4. Define a method enterCritialSection() for a Process that requests to enter the critical section.
5. Define a method exitCriticalSection() for a Process that exits from the critical section.
6. Simulate the processes involved in the critical section.
7. Compile the program using the javac command.
8. Run and execute the program.

PROGRAM:

```
import java.util.*;
import java.util.concurrent.ConcurrentLinkedQueue;
enum MessageType {
    REQUEST, REPLY, RELEASE
}
class Message {
    public MessageType type;
    public Process sender;
    public Message(MessageType type, Process sender) {
        this.type = type;
        this.sender = sender;
    }
}

class Process extends Thread {
    private final int id;
    private final Set<Process> quorum;
private boolean inCriticalSection = false;
private Queue<Message> messageQueue = new ConcurrentLinkedQueue<>();
private Set<Process> granted = new HashSet<>();
private Process requestingProcess = null;

public Process(int id, Set<Process> quorum) {
    this.id = id;
    this.quorum = quorum;
}

public synchronized void receiveMessage(Message msg) {
    messageQueue.add(msg);
    notify(); // Wake up the thread if it's waiting for messages
}

public void run() {
    try {
        // Requesting critical section
        enterCriticalSection();
        // Simulate critical section work
        Thread.sleep(1000);
        // Exiting critical section
        exitCriticalSection();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

private void enterCriticalSection() throws InterruptedException {
    for (Process p : quorum) {
        p.receiveMessage(new Message(MessageType.REQUEST, this));
    }

    synchronized (this) {
        while (granted.size() < quorum.size()) {
            wait();
        }
        inCriticalSection = true;
        System.out.println("Process " + id + " is in critical section");
    }
}

private void exitCriticalSection() {
    synchronized (this) {
        inCriticalSection = false;
        for (Process p : quorum) {
            p.receiveMessage(new Message(MessageType.RELEASE, this));
        }
        granted.clear();
        if (requestingProcess != null) {
            requestingProcess.receiveMessage(new Message(MessageType.REPLY, this));
            requestingProcess = null;
        }
    }
    System.out.println("Process " + id + " has exited critical section");
}

public void processMessages() {
    synchronized (this) {
        while (!messageQueue.isEmpty()) {
            Message msg = messageQueue.poll();
            switch (msg.type) {
                case REQUEST:
                    if (!inCriticalSection && requestingProcess == null) {
                        msg.sender.receiveMessage(new Message(MessageType.REPLY, this));
                        granted.add(msg.sender);
                    } else {
                        requestingProcess = msg.sender;
                        }
                        break;
                    case REPLY:
                        granted.add(msg.sender);
                        break;
                    case RELEASE:
                        if (requestingProcess != null) {
                            requestingProcess.receiveMessage(new Message(MessageType.REPLY, this));
                            granted.add(requestingProcess);
                            requestingProcess = null;
                        }
                        break;
                }
            }
        }
    }
}

public class ma {
    public static void main(String[] args) {
        Set<Process> quorum1 = new HashSet<>();
        Set<Process> quorum2 = new HashSet<>();
        Set<Process> quorum3 = new HashSet<>();
        Process p1 = new Process(1, quorum1);
        Process p2 = new Process(2, quorum2);
        Process p3 = new Process(3, quorum3);
        quorum1.addAll(Arrays.asList(p2, p3));
        quorum2.addAll(Arrays.asList(p1, p3));
        quorum3.addAll(Arrays.asList(p1, p2));
        p1.start();
        p2.start();
        p3.start();
        new Thread(() -> {
            while (true) {
                p1.processMessages();
                p2.processMessages();
                p3.processMessages();
                try {
                    Thread.sleep(10); // Small delay to avoid CPU overuse
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ponnu>e:

E:\>set path=c:\program files\java\jdk-21\bin

E:\>javac ma.java

E:\>java ma
Process 1 is in critical section
Process 2 is in critical section
Process 3 is in critical section
```

RESULT:

Thus, the experiment has been executed successfully.

Ex. No: 9 **INSTALLATION OF VIRTUALBOX WITH LINUX OS**

Date: 20.03.2024

AIM: To Install VirtualBox with different flavours of Linux OS on top of windows7 or 8 or 10 OS.

PROCEDURE: The installation is divided into

1. Installation of VirtualBox on Windows 10/8/7
2. Creation of Ubuntu (Linux) VM
3. Installation of Linux OS on VirtualBox

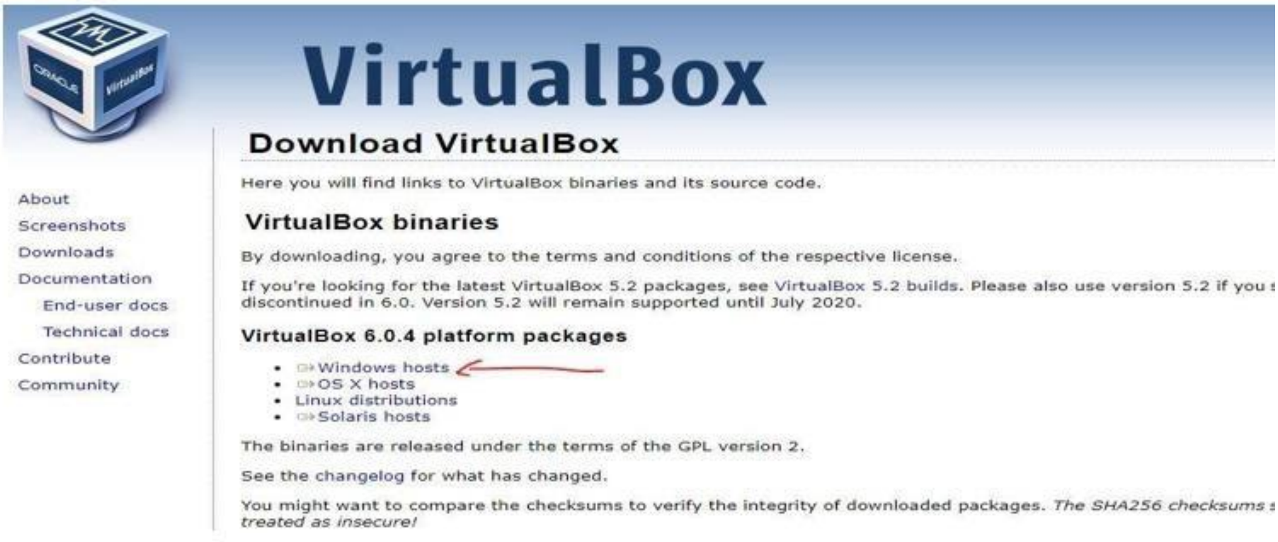
Installation of VirtualBox on Windows 10/8/7

1. **Download** VirtualBox software from **Oracle official website.**
2. **Double-click** on the downloaded **VirtualBox** Win.exe file to bring up the welcome screen. Click **Next**.
3. Installation files and set the installation path. If you are not familiar, then keep the default configuration, select the **Next** button.
4. Leave the pre-selected **VirtualBox** shortcuts as it is and click on **Next** button.
5. When installing VirtualBox, it involves network functions. The wizard will automatically create a **virtual network card,** which will temporarily interrupt your network. But of course, it will return to normal immediately. So, click **Yes**.
6. Now you can go to install this virtualization software. Click **Install**.During the period, you can see that the current network was interrupted and immediately resumed.
7. Click **Finish** to launch Oracle VM VirtualBox.

Screenshots of the above steps:

Step 1: Download VirtualBox for Windows 10/8/7

Download VirtualBox software from **Oracle official website**: **Download VirtualBox**



Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 5.2 packages, see VirtualBox 5.2 builds. Please also use version 5.2 if you
discontinued in 6.0. Version 5.2 will remain supported until July 2020.

VirtualBox 6.0.4 platform packages

- Windows hosts
- OS X hosts
- Linux distributions
- Solaris hosts

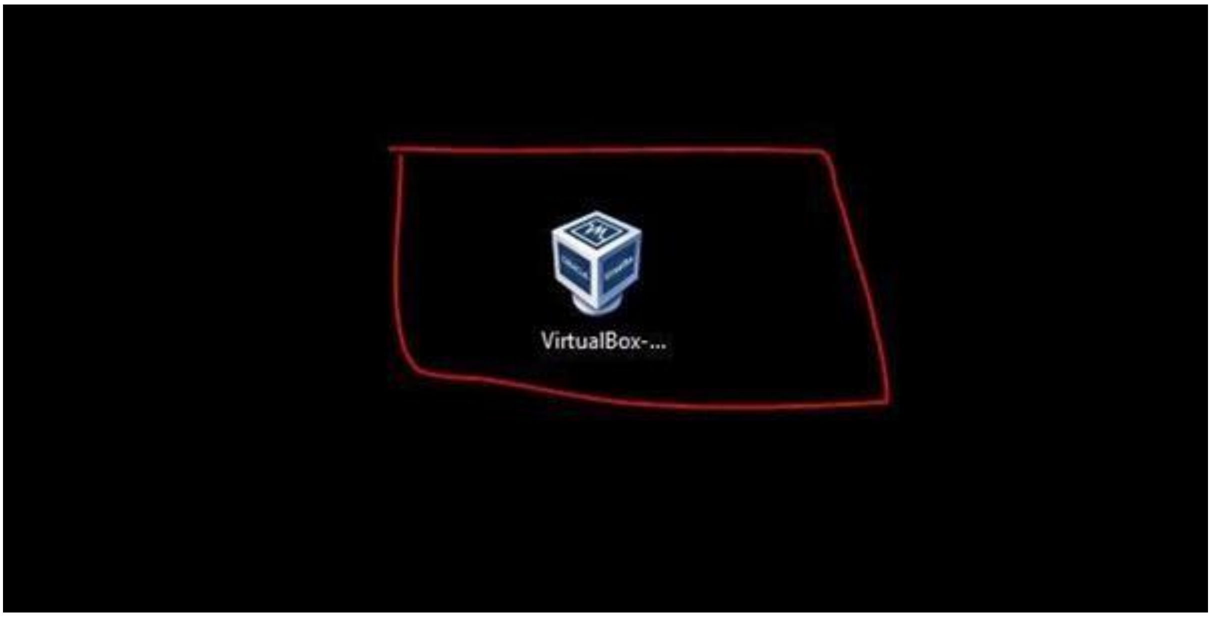
The binaries are released under the terms of the GPL version 2.

See the changelog for what has changed.

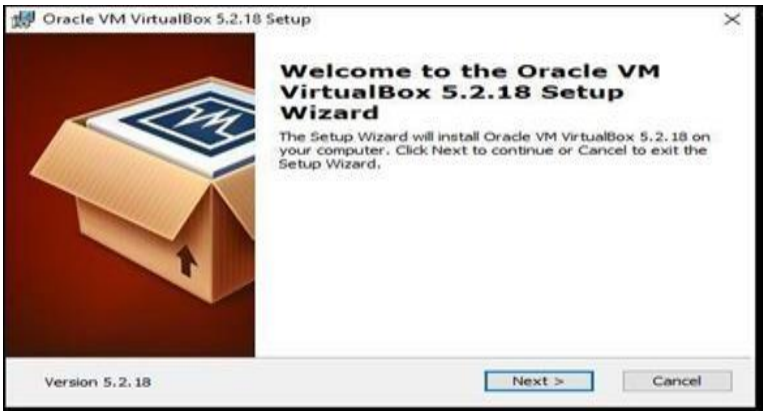
You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums*
treated as insecure!

Step 2: Run the VirtualBox.exe file.

The downloaded VirtualBox file will be in EXE format to run that just double click on it and run it as administrator.



Click on **Next** button to start Oracle VirtualBox installation Setup Wizard.



Oracle VM VirtualBox 5.2.18 Setup

Welcome to the Oracle VM VirtualBox 5.2.18 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 5.2.18 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 5.2.18

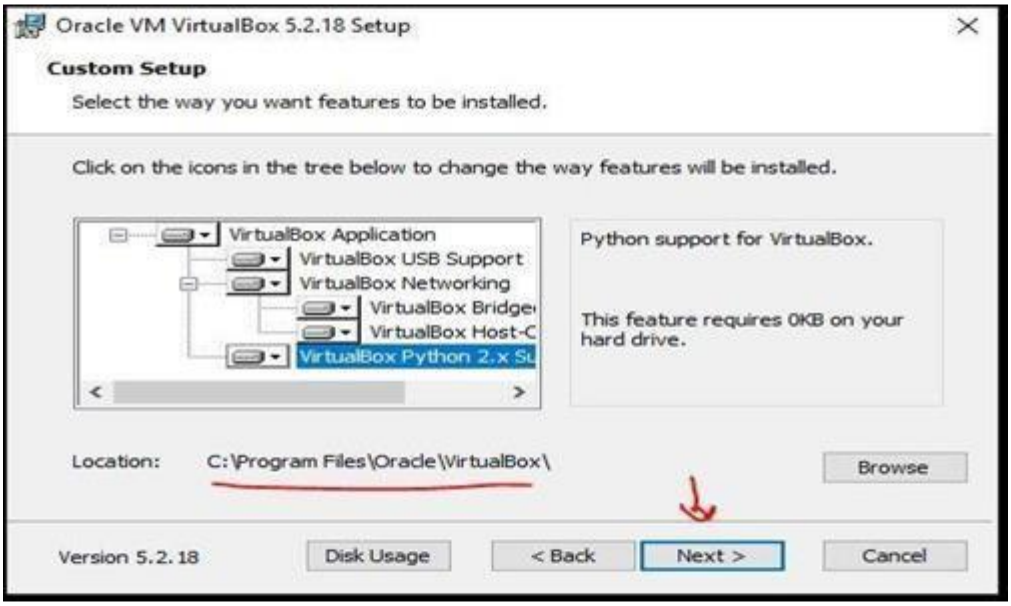
Step 3: VirtualBox shortcuts

At this stage, you will see multiple shortcuts:

- **Create start menu entries:** To create a Virtualbox shortcut in the start menu of the Windows 10/8/7
- **Create a shortcut on the desktop**: This will create a shortcut on Desktop
- **Create a shortcut in the Quick Launch Bar**: You will get a shortcut in the Taskbar.
- **Register file associations**: Create Virtualbox file entries in Windows registries. Leave them as it is and click on the **NEXT** button.

Step 4: File Location

By default the VirtualBox will install its core files in the C: Drive. In case you have low space on the C: Drive, then just click on the Browse button and select the location where you want to install it. However, if you are not acquainted with this option then simply leave it as default and click on the **NEXT** button.



Oracle VM VirtualBox 5.2.18 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

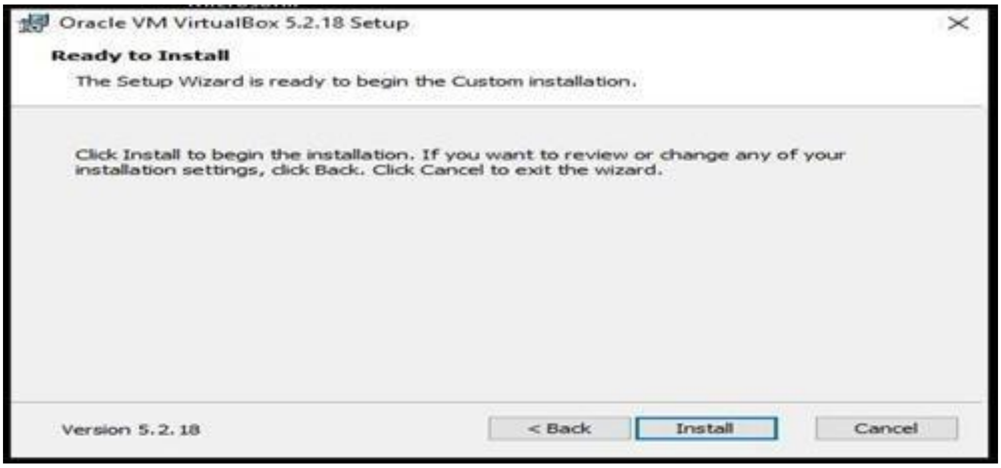
Python support for VirtualBox.

This feature requires 0KB on your hard drive.

Location: C:\Program Files\Oracle\VirtualBox\

Step 5: Install VirtualBox

Click on the **Install** button to begin the installation.



Oracle VM VirtualBox 5.2.18 Setup

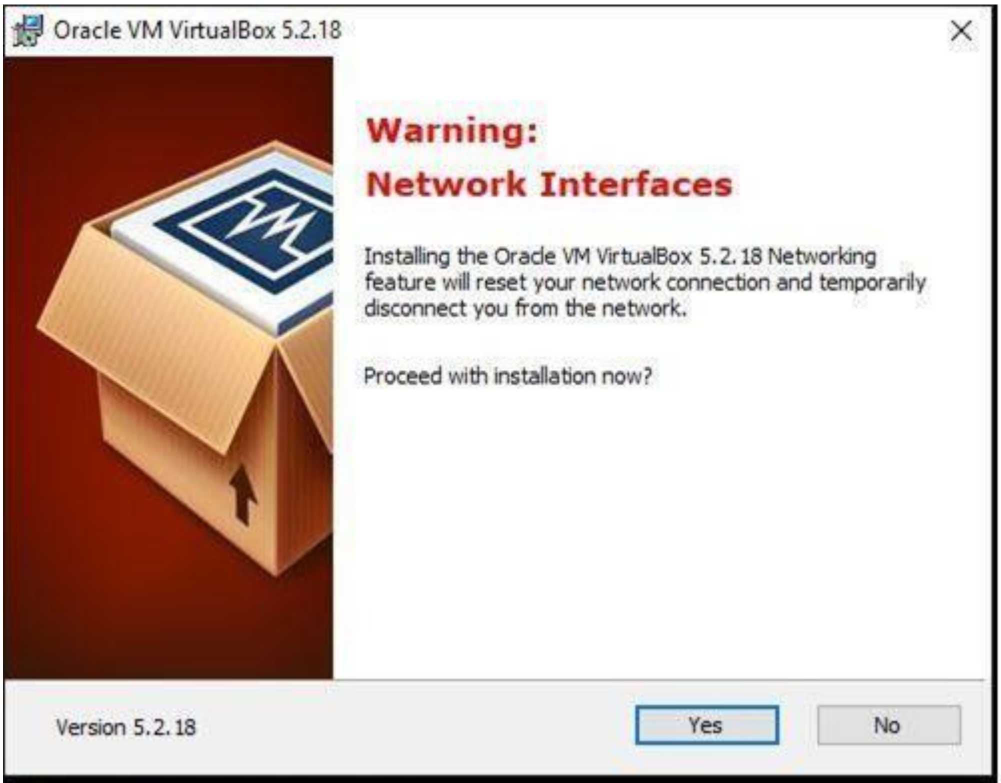
Ready to Install

The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Step 6: Warning: Network Interfaces

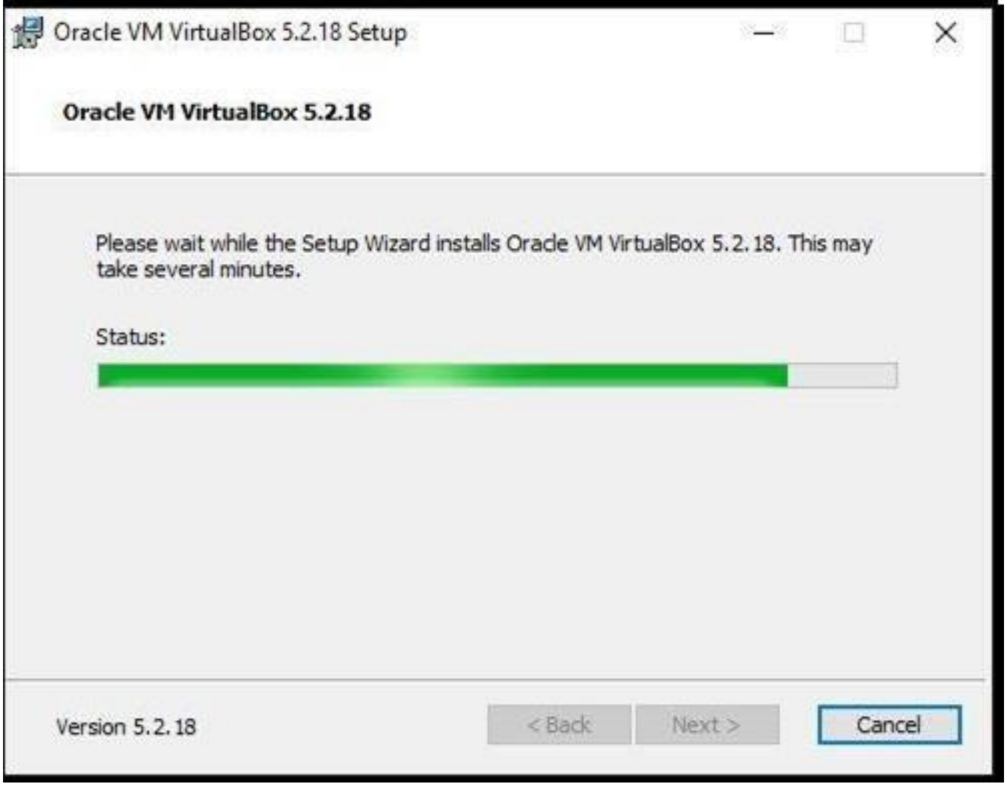
To create Virtual Adapters, the VirtualBox will reset your network connection and disconnect it temporarily for a few seconds and then again it will return to its normal state. So, click on the **YES** button.



Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 5.2.18 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?



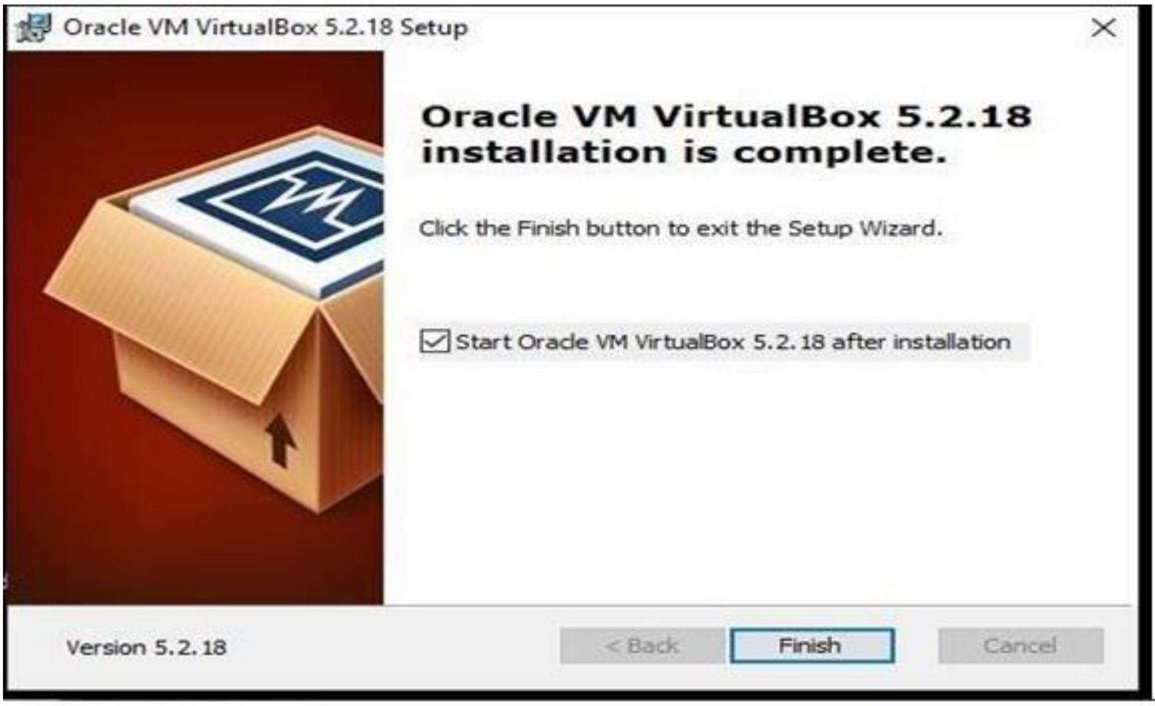
Oracle VM VirtualBox 5.2.18

Please wait while the Setup Wizard installs Oracle VM VirtualBox 5.2.18. This may take several minutes.

Status:

Step 7: Installation is completed

After installing, the installation wizard will show you a **Finish** button, click on that and it will start the VirtualBox on your Windows 10/7/8 machines.



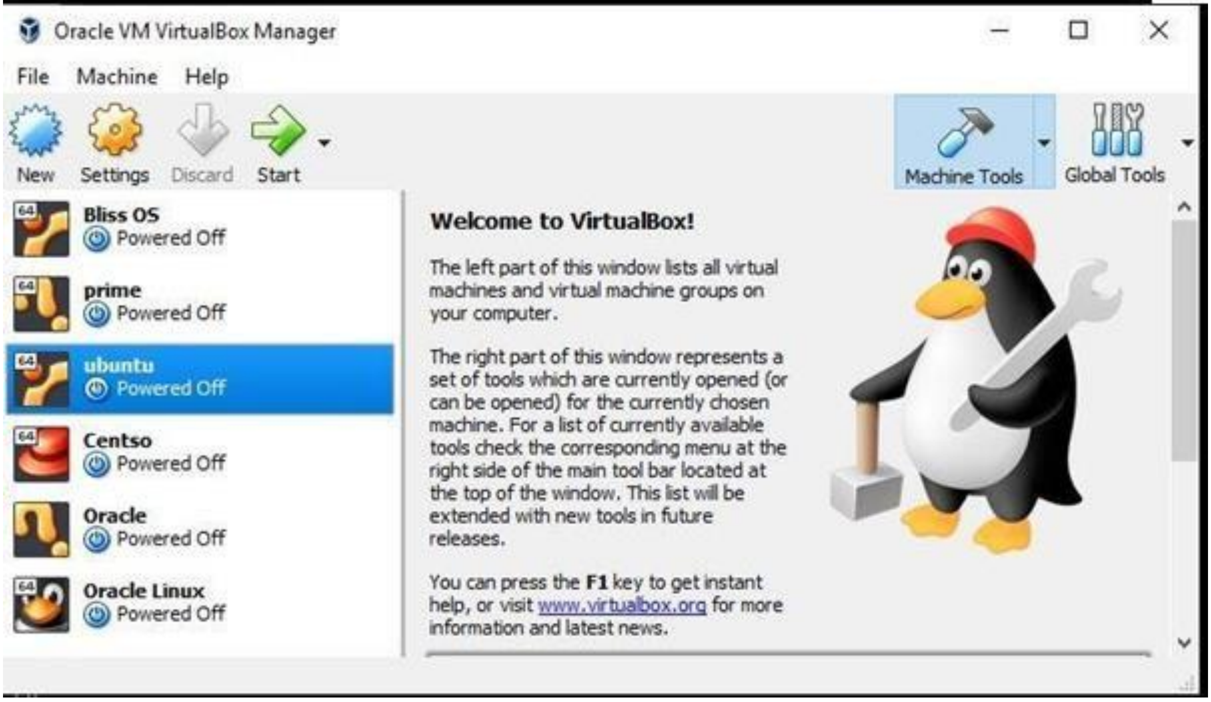
Oracle VM VirtualBox 5.2.18 Setup

Oracle VM VirtualBox 5.2.18 installation is complete.

Click the Finish button to exit the Setup Wizard.

☑ Start Oracle VM VirtualBox 5.2.18 after installation

Version 5.2.18



Oracle VM VirtualBox Manager

Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

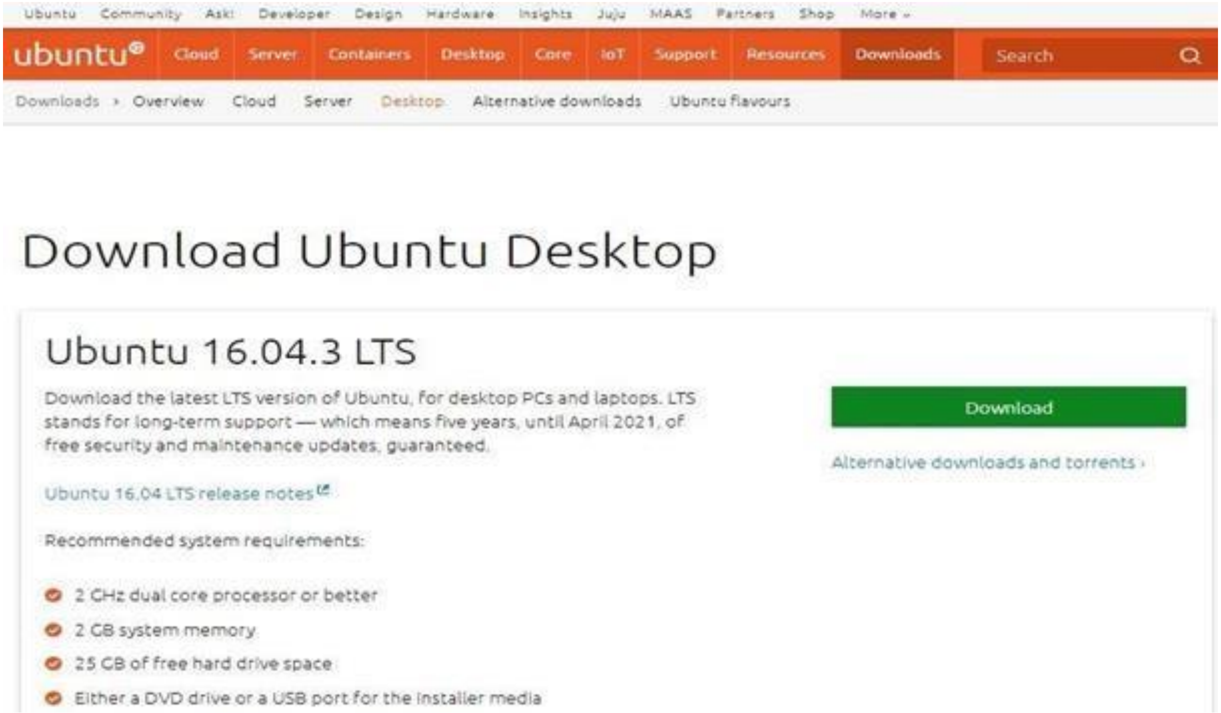
You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Creation of Ubuntu VM in VirtualBox

Step 1: Download Ubuntu OS

The open source Ubuntu Linux comes in different flavors and you can download any of them from the official Ubuntu's website. Here is the Link: www.ubuntu.com/download/desktop.

Note: If you already have the Ubuntu.iso file then leave this step.


Ubuntu Community Ask! Developer Design Hardware Insights Juju MAAS Partners Shop More
ubuntu Cloud Server Containers Desktop Core IoT Support Resources Downloads Search
Downloads › Overview Cloud Server Desktop Alternative downloads Ubuntu flavours

Download Ubuntu Desktop

Ubuntu 16.04.3 LTS

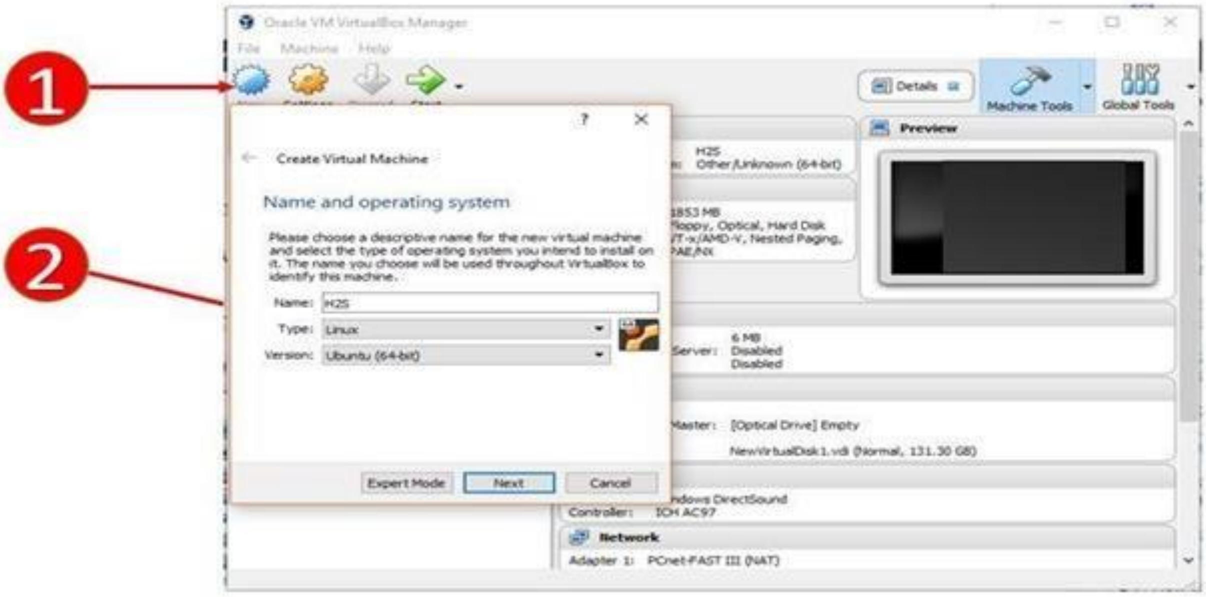
Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2021, of free security and maintenance updates, guaranteed.

Ubuntu 16.04 LTS release notes

Recommended system requirements:

- 2 GHz dual core processor or better
- 2 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media

Step 2: After successful Virtualbox installation, run it to create an **Ubuntu VM**. Click on **New** and give some name to your Ubuntu VM. For example, here we have used **H2S**. From the type drop-down box select the OS type which is Linux and Version is Ubuntu 64 bit. If you have Ubuntu 32 bit version then please select that.



Create Virtual Machine

Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: H2S

Type: Linux

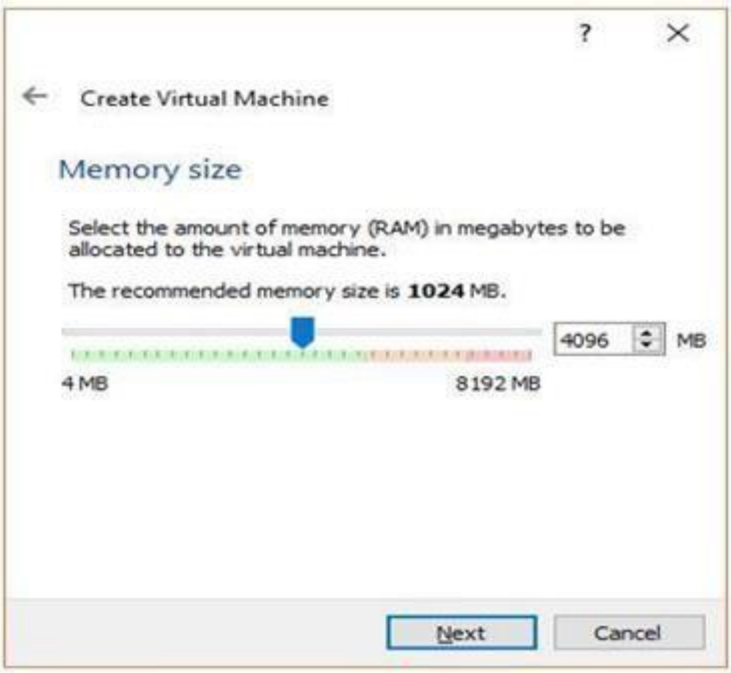
Version: Ubuntu (64-bit)

H2S
Other/Unknown (64-bit)
1853 MB
Floppy, Optical, Hard Disk
VT-x/AMD-V, Nested Paging, PAE/NX
Server: 6 MB
Disabled
Disabled
Master: [Optical Drive] Empty
NewVirtualDisk1.vdi (Normal, 131.30 GB)
Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: PCnet-FAST III (NAT)

Step 3: How Much Memory Do You Give Your Virtual Machine

In this step, the Vitrualbox will ask to set the Virtual Machine Memory Size for Ubuntu VM. The recommended RAM for Ubuntu OS is 2 GB or 2048 MB but you can assign more for better performance.

For example, here in the Windows 10 PC, we have maximum 8GB memory and out of that, we are going to assign 4GB to the Ubuntu VM.



Create Virtual Machine

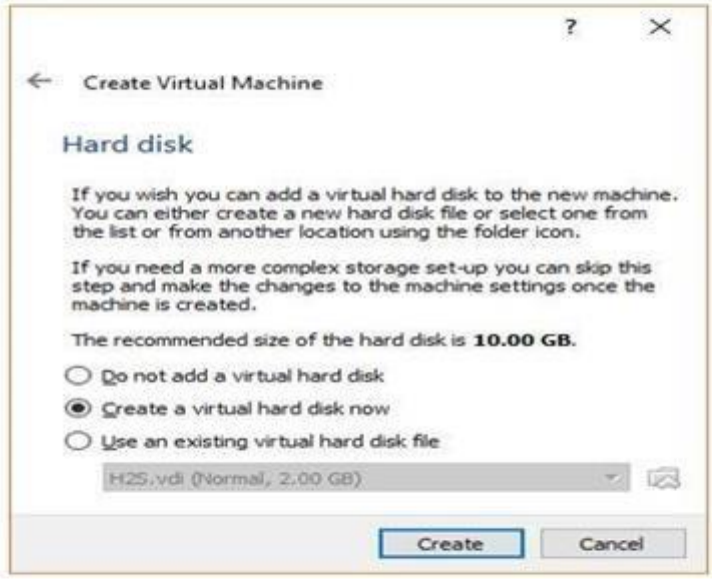
Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

Step 4: Create A Virtual Hard Drive For Ubuntu VM (Virtual Machine)

After assigning the memory, its time to provide some space for the installation of Ubuntu VM. To create a new virtual hard disk select the option "**Create a virtual hard disk now**" and click "Create".



Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.

○ Do not add a virtual hard disk

● Create a virtual hard disk now

○ Use an existing virtual hard disk file

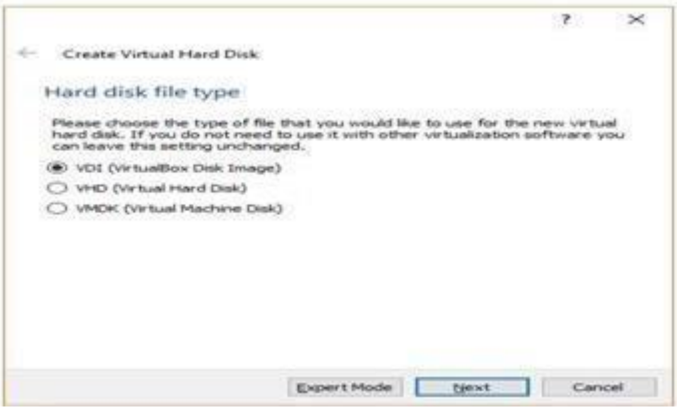
H2S.vdi (Normal, 2.00 GB)

Step 5: Choose the Virtual Hard disk. Type. The Virtualbox offers three types of Virtual hard drives:

1. Virtual Disk Image (VDI)
2. Virtual Hard Disk (VHD)
3. Virtual Machine Disk (VMDK)

If you are planning to use the Virtual hard drive with some other virtualization software in future such as with VM player or Windows Hyper-V then you choose according to that otherwise leave it as it is "***VDI***" and click on ***NEXT***.

There are a number of different hard drive types that you can choose from. Choose "VDI" and click "Next".



Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ● VDI (VirtualBox Disk Image)
- ○ VHD (Virtual Hard Disk)
- ○ VMDK (Virtual Machine Disk)

Step 6: Storage on Physical Hard disk for Ubuntu VM

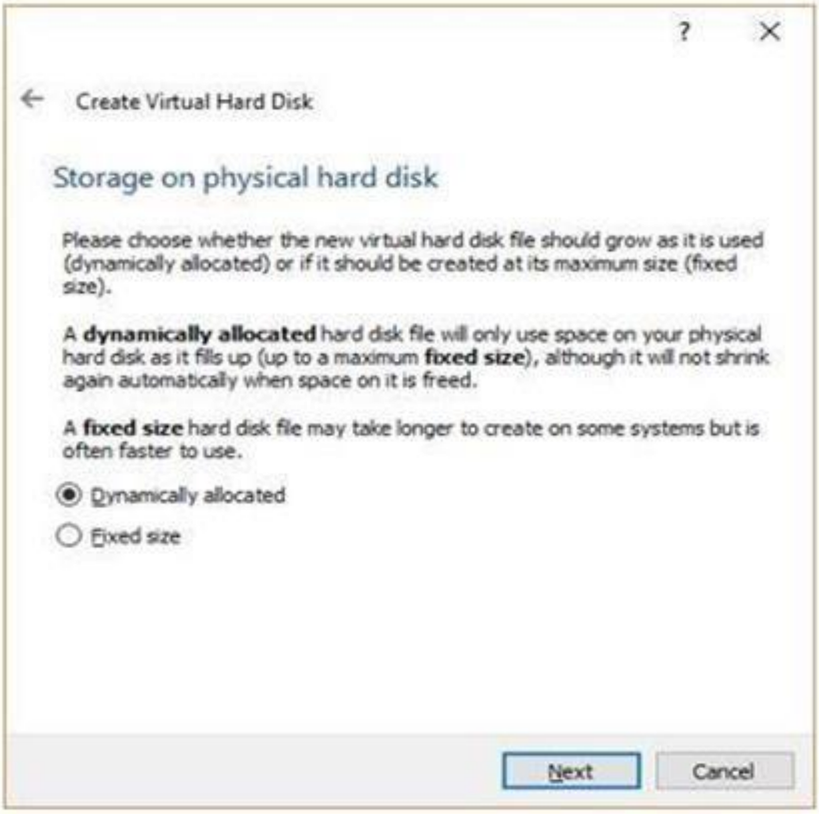
To install Ubuntu Virtual Machine files on physical storage of Windows 10, the Virtualbox offers two options:

1. Dynamically allocated
2. Fixed Size

The Dynamically allocated hard disk option will only use space as it is required. For example, you assigned 30 GB to the Ubuntu VM but if it requires 10Gb initially then the Virtualbox uses only that and is not going to block the whole 30GB. And in the future, it requires more, expands according to that. It is good in terms of disk space but not performance wise.

Fixed Size on another hand blocks the whole space you have assigned to the VM. For example, if you allocated the 30GB, then the machine will straight away assign that portion from the physical hard drive to the Ubuntu VM. The Fixed size allocation is better for performance but take some time to create if you are assigning a large amount of space.

Choose the option you would like and click “Next”.



Create Virtual Hard Disk

Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

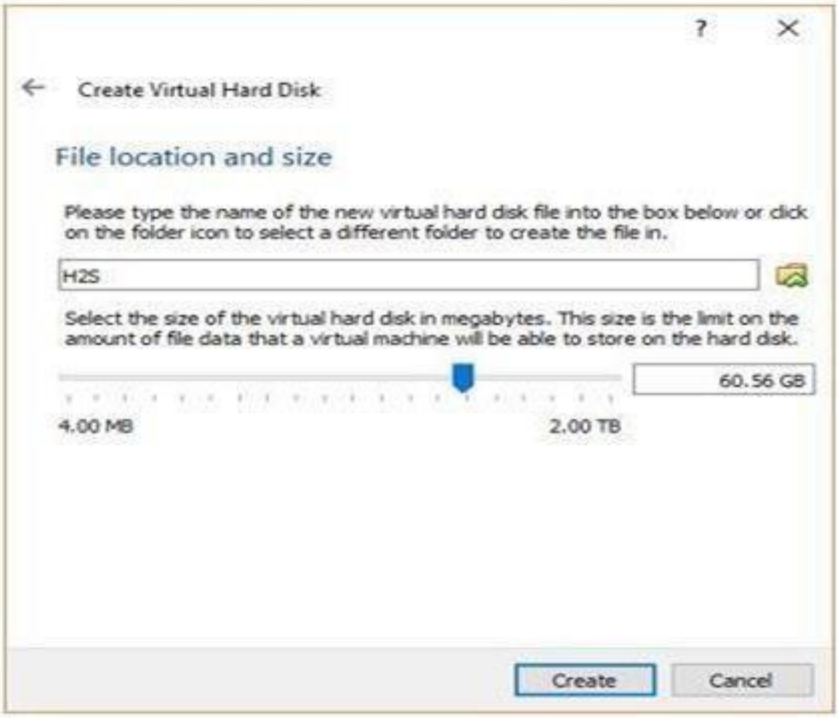
A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

● Dynamically allocated

○ Fixed size

Step 7: Virtual Hardrive File location and Size

Give some name to your virtual hard disk and select the amount of space you want to assign the Ubuntu VM. The minimum recommended space is 25GB. You can assign more for better performance.



Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

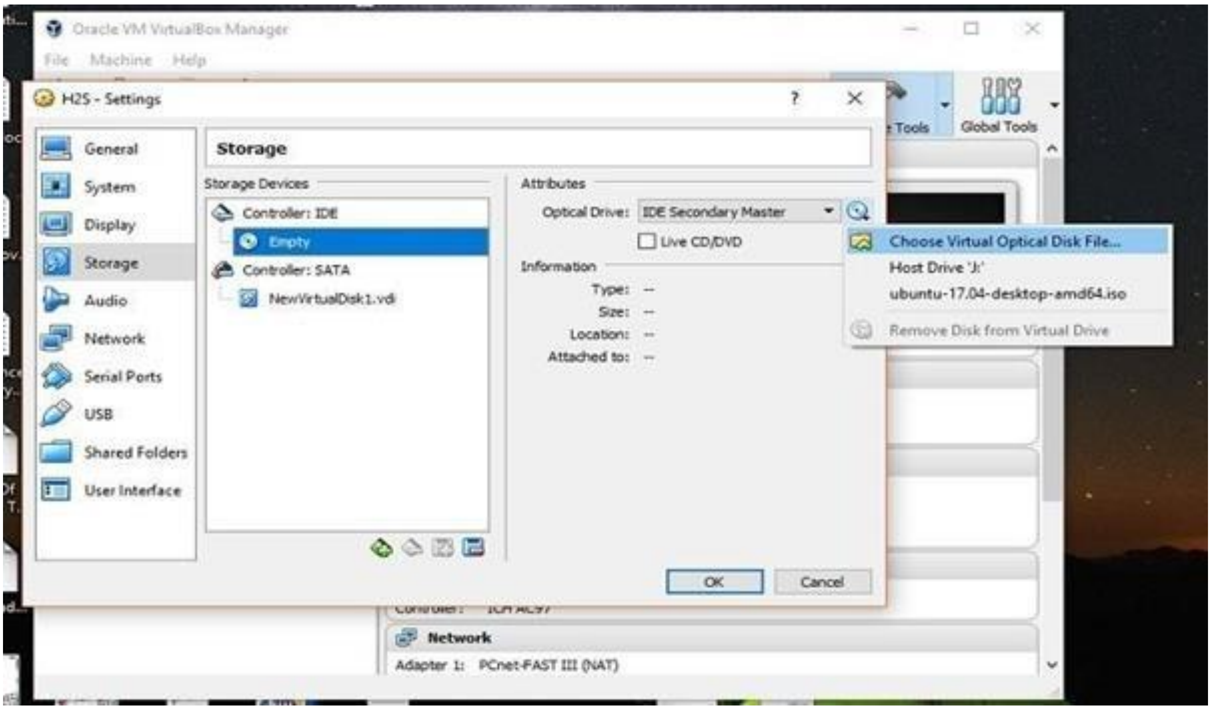
H2S

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

60.56 GB

Step 8: Assign Ubuntu ISO to VirtualBox

Go to settings and from storage click on the empty CD-Rom icon and from the Optical drive option **choose the Virtual optical Disk File** and select the Ubuntu.iso file which is our downloaded beginning of this article. After selecting the ISO file click **OK**.



Storage

Storage Devices

- Controller: IDE
  - Empty
- Controller: SATA
  - NewVirtualDisk1.vdi

Attributes

Optical Drive: IDE Secondary Master

□ Live CD/DVD

Information

Type: --

Size: --

Location: --

Attached to: --

Controller: ICH AC97

Network

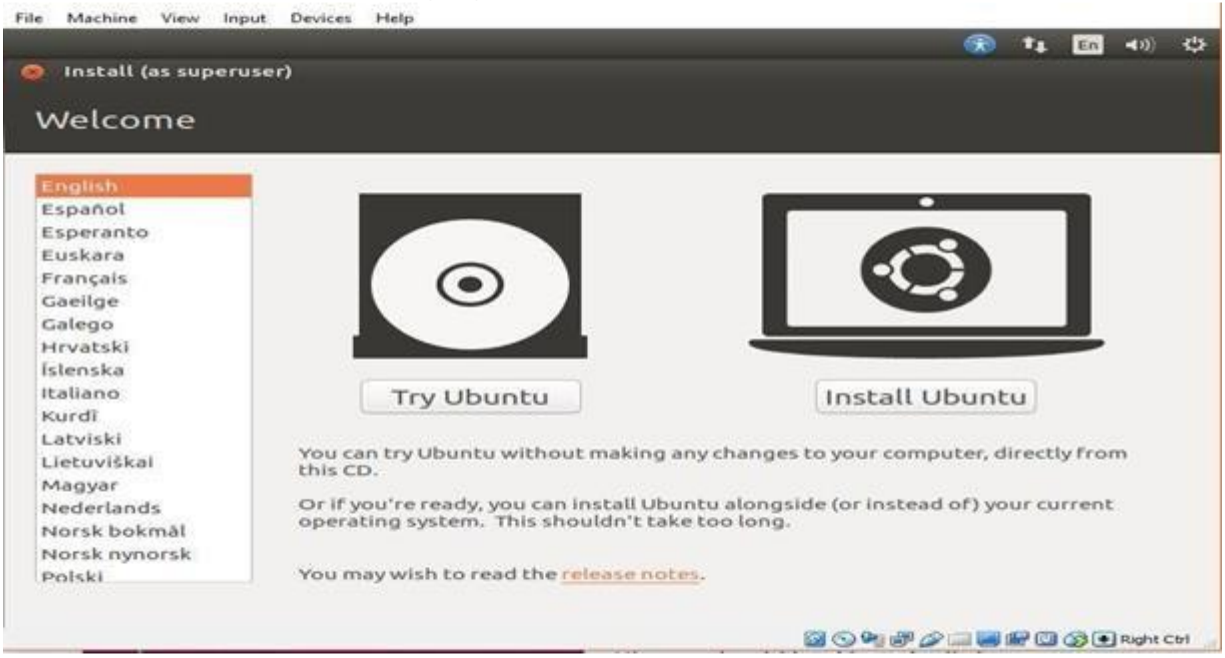
Adapter 1: PCnet-FAST III (NAT)

Installation of Ubuntu OS on VirtualBox

Step 1: On the top of the Virtual box you will have an option “**START**“, click on that to initialize the Ubuntu installation process on Windows 10.

Step 2: The Ubuntu first screen will load two options **Try Ubuntu** or **Install Ubuntu.**

Select the installation language and after that the “**Install Ubuntu**” option.



Welcome

English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Italiano
Kurdî
Latviski
Lietuviškai
Magyar
Nederlands
Norsk bokmål
Norsk nynorsk
Polski

You can try Ubuntu without making any changes to your computer, directly from this CD.

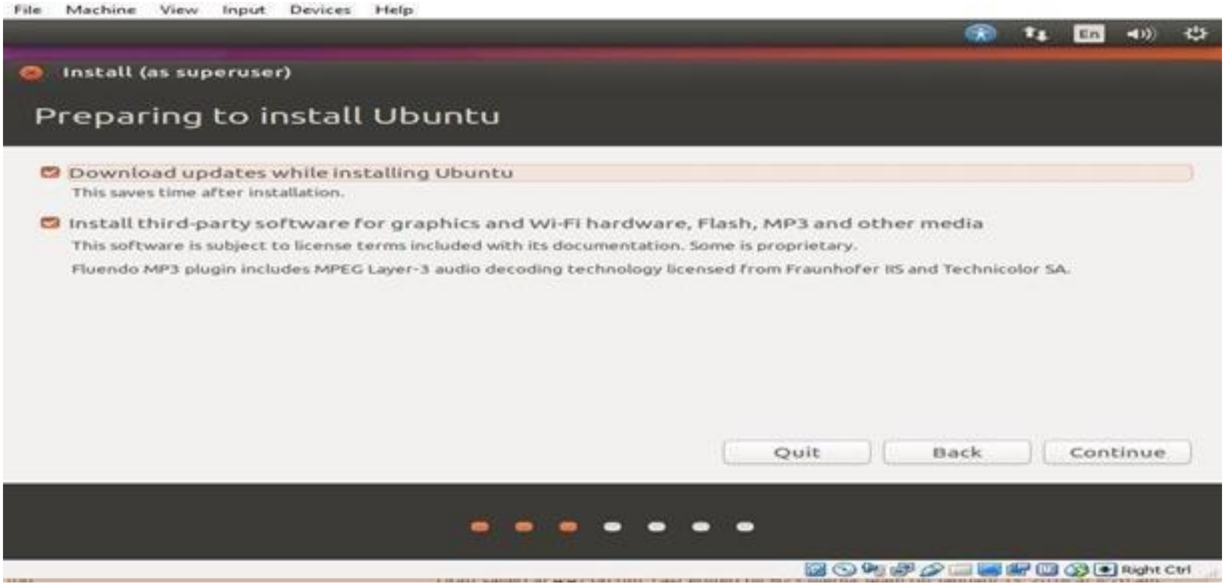
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

Step 3: If you have enough internet bandwidth and then you can select the option to download the updates while installing Ubuntu.

The second option doesn’t require an internet connection and recommended to select it to install third party software such as graphics driver, Mp3 player, flash and other media files.

Click “Continue”.



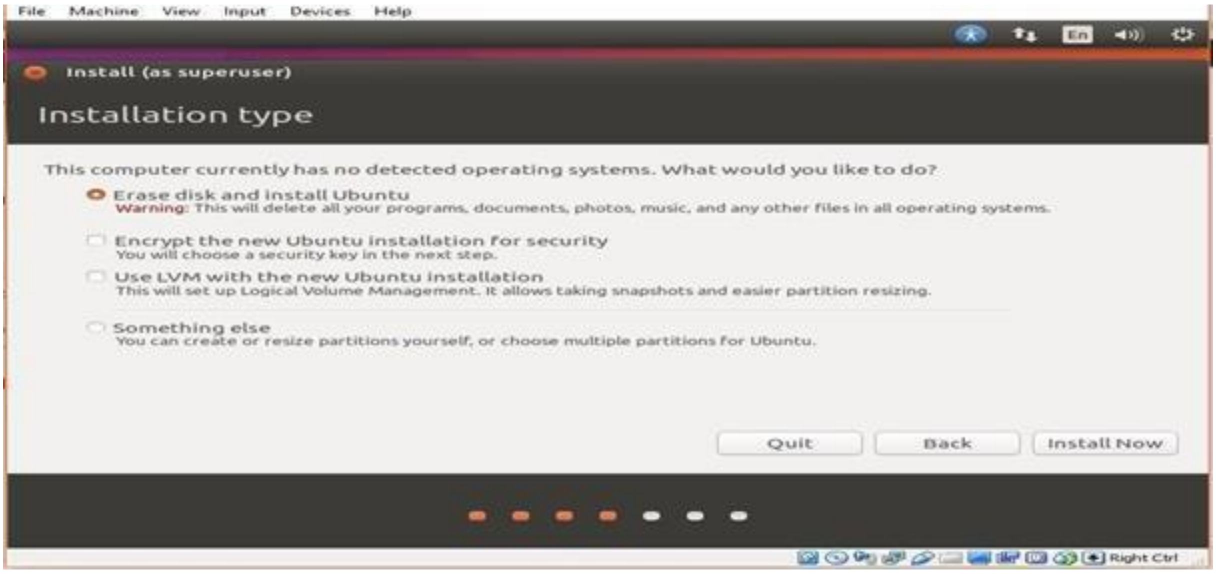
Preparing to install Ubuntu

☑ Download updates while installing Ubuntu
This saves time after installation.

☑ Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media
This software is subject to license terms included with its documentation. Some is proprietary.
Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Step 4: In this step, you will decide how you want to decide the Ubuntu either clean installation or dual boot with some other OS. Leave the default option the “Erase disk and install Ubuntu” option because it is on the virtual machine won’t going to affect the physical Windows 10 machine.

Click “***Install Now***” and “***Continue***“.



Installation type

This computer currently has no detected operating systems. What would you like to do?

● Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

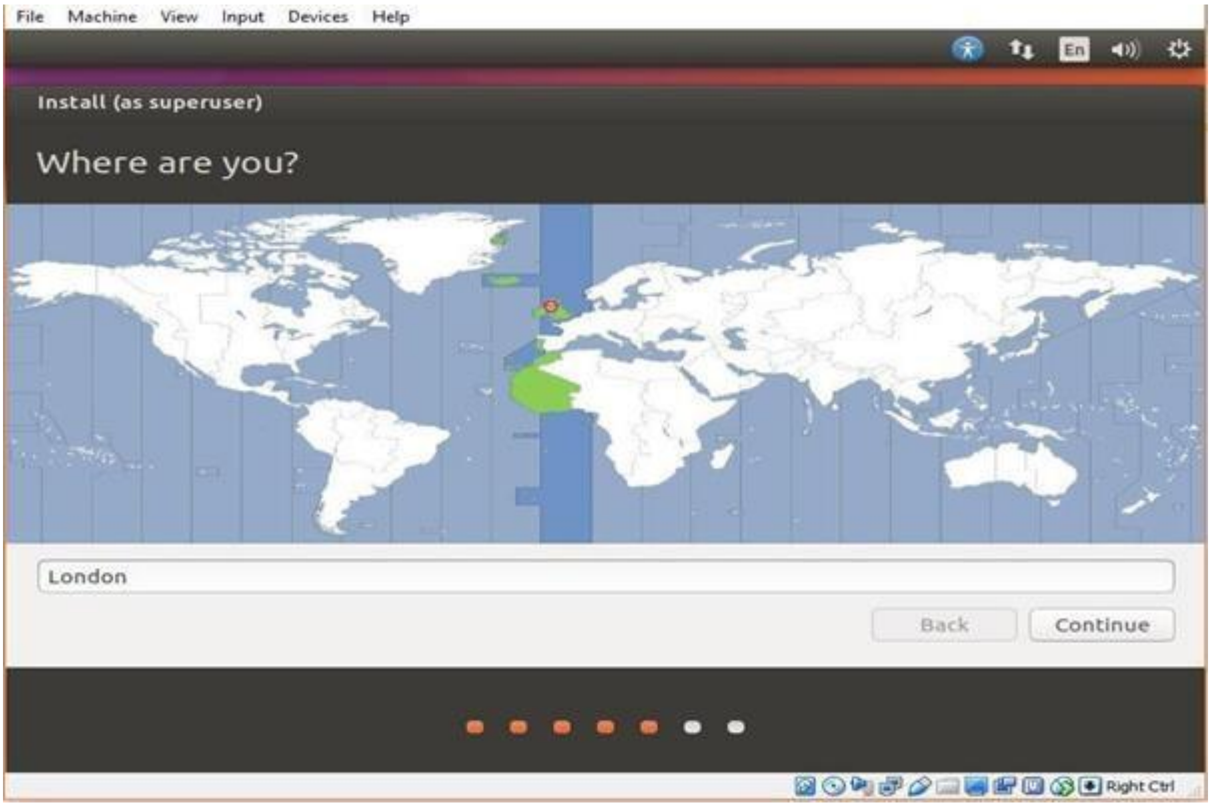
□ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

□ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Step 5: Select your country to sync the Ubuntu OS time zone with your’s and click “Continue”.

Step 6: Click on the “**Detect Keyboard Layout**” to automatically detect your keyboard layout and if the machine is not able to do it, you can select it manually. Click “Continue”.

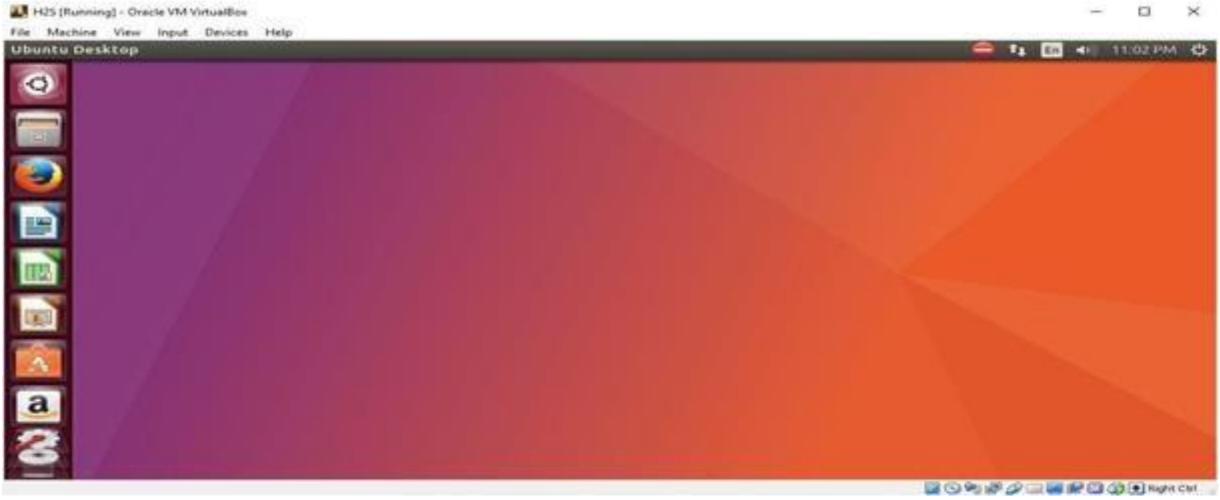


Install (as superuser)

Where are you?

London

Step 7: Create a user and set the password for your Virtual Ubuntu machine and click on continue to install the Ubuntu Virtualbox.


H2S [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu Desktop
En
11:02 PM
Right Ctrl

Finally, the Ubuntu is installed on **VirtualBox on Windows 10** as host machine

RESULT:

Thus the Virtualbox with Linux OS has been installed successfully.

Ex. No: 10 **CONFIGURING A SINGLE-NODE HADOOP CLUSTER**
Date: 20.03.2024

AIM: To set-up and configure a single-node Hadoop cluster.

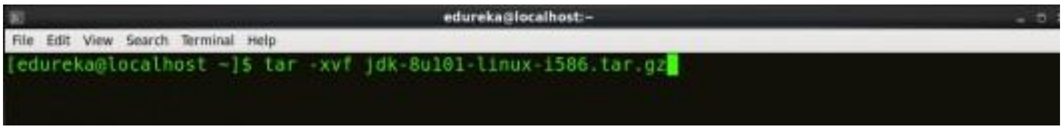
PRE-REQUISITES:

- VIRTUAL BOX: it is used for installing the operating system on it.
- OPERATING SYSTEM: You can install Hadoop on Linux-based operating systems. Ubuntu and CentOS are very commonly used. In this tutorial, we are using CentOS.
- JAVA: You need to install the Java 8 package on your system.
- HADOOP: You require the Hadoop 2.7.3 package and Install Hadoop.

Step 1: *Download the Java 8 or above Package. Save this file in your home directory.*

Step 2: *Extract the Java Tar File.*

Command: tar -xvf jdk-8u101-linux-i586.tar.gz

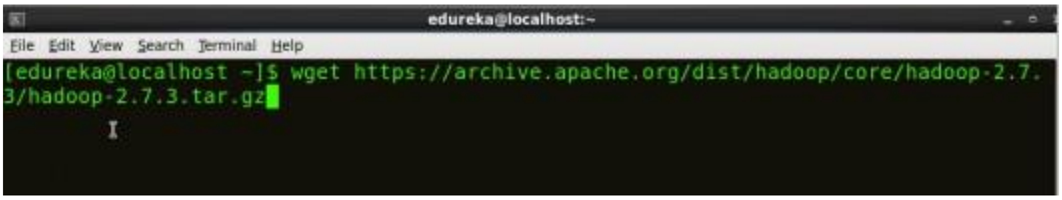


```
[edureka@localhost ~]$ tar -xvf jdk-8u101-linux-i586.tar.gz
```

Fig: Hadoop Installation – Extracting Java Files

Step 3: *Download the Hadoop 2.7.3 Package.*

***Command*:** wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz

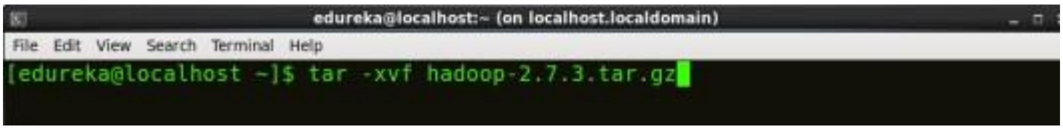


```
[edureka@localhost ~]$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.
3/hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Downloading Hadoop

Step 4: *Extract the Hadoop tar File.*

Command: tar -xvf hadoop-2.7.3.tar.gz



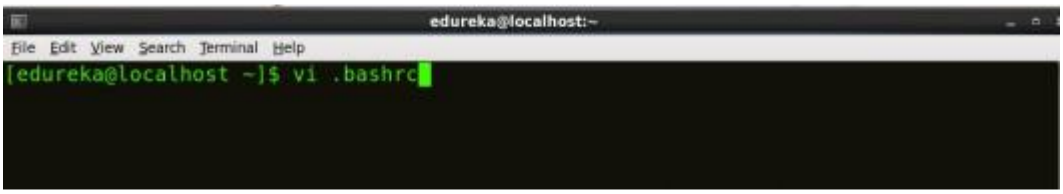
```
[edureka@localhost ~]$ tar -xvf hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Extracting Hadoop Files

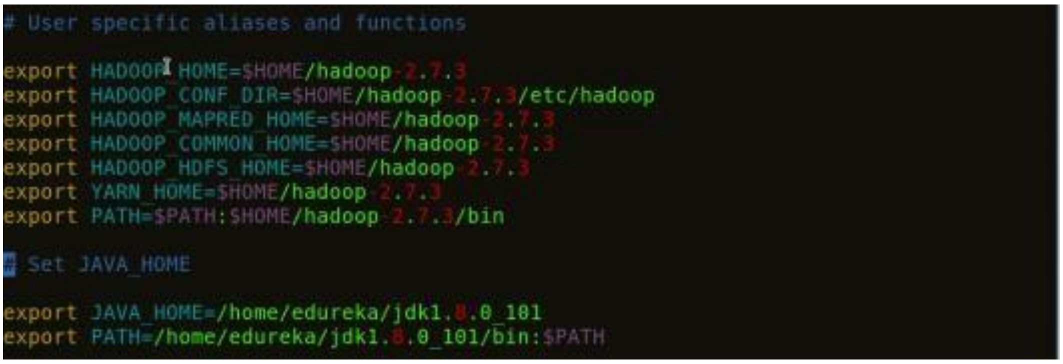
Step 5: *Add the Hadoop and Java paths in the bash file (.bashrc).*

Open**. bashrc** file. Now, add Hadoop and Java Path as shown below.

Command: vi .bashrc



```
[edureka@localhost ~]$ vi .bashrc
```



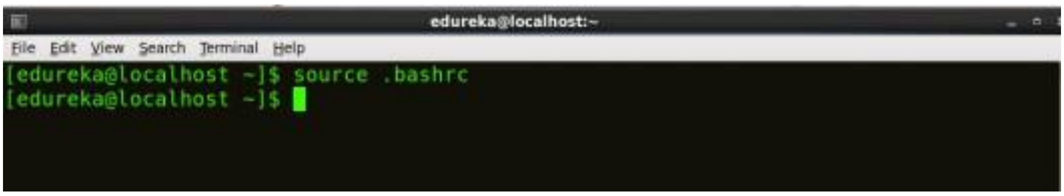
```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.

Command: source .bashrc

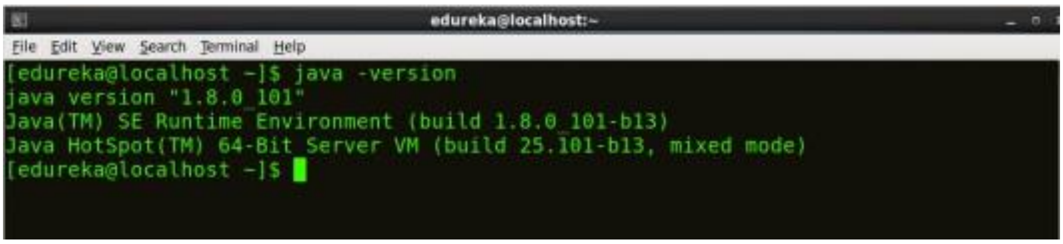


```
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$ 
```

Fig: Hadoop Installation – Refreshing environment variables

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.

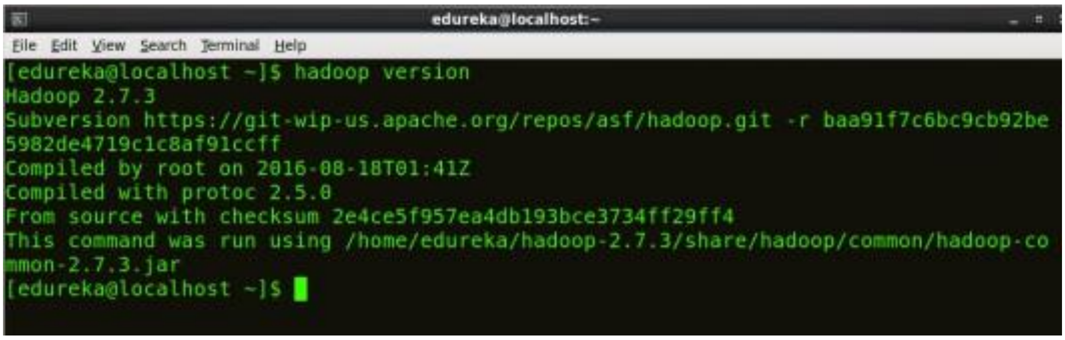
Command: java -version



```
[edureka@localhost ~]$ java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
[edureka@localhost ~]$ 
```

Fig: Hadoop Installation – Checking Java Version

Command: hadoop version



```
[edureka@localhost ~]$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be
5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /home/edureka/hadoop-2.7.3/share/hadoop/common/hadoop-co
mmon-2.7.3.jar
[edureka@localhost ~]$
```

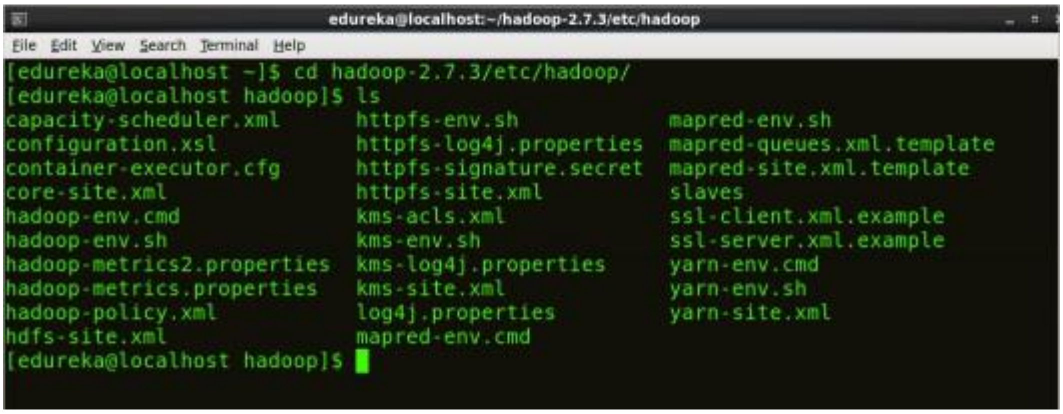
Fig: Hadoop Installation – Checking Hadoop Version

Step 6: *Edit the Hadoop Configuration files.*

Command: cd hadoop-2.7.3/etc/hadoop/

Command: ls

All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:



edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

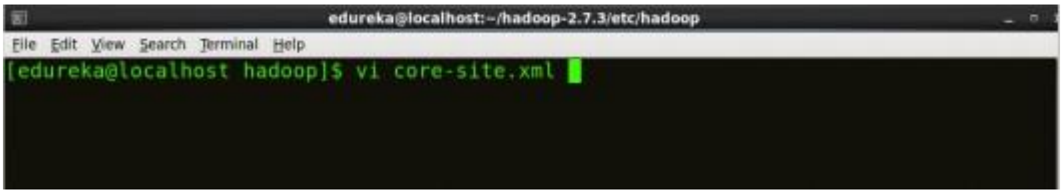
```
[edureka@localhost ~]$ cd hadoop-2.7.3/etc/hadoop/
[edureka@localhost hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh              mapred-env.sh
configuration.xsl           httpfs-log4j.properties    mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret    mapred-site.xml.template
core-site.xml               httpfs-site.xml            slaves
hadoop-env.cmd              kms-acls.xml               ssl-client.xml.example
hadoop-env.sh               kms-env.sh                 ssl-server.xml.example
hadoop-metrics2.properties  kms-log4j.properties       yarn-env.cmd
hadoop-metrics.properties   kms-site.xml               yarn-env.sh
hadoop-policy.xml           log4j.properties           yarn-site.xml
hdfs-site.xml               mapred-env.cmd
[edureka@localhost hadoop]$
```

Fig: Hadoop Installation – Hadoop Configuration Files

Step 7: *Open* ***core-site.xml*** *and edit the property mentioned below inside configuration tag:*

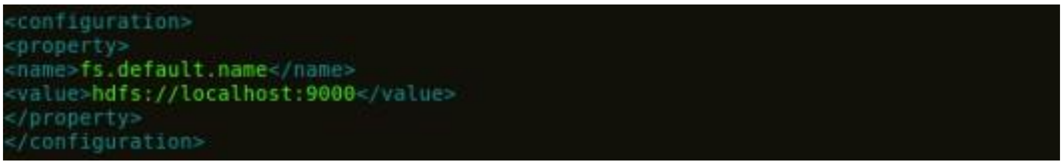
core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.

Command: vi core-site.xml



edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ vi core-site.xml
```



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

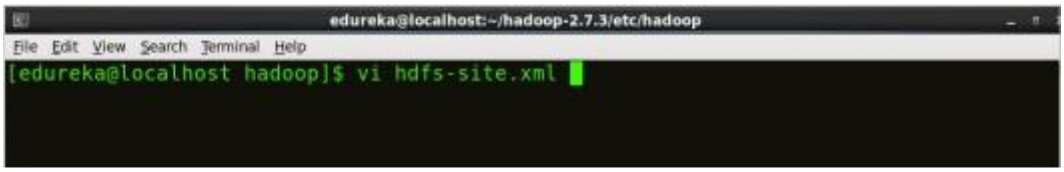
Fig: Hadoop Installation – Configuring core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>
```

***Step 8:** Edit **hdfs-site.xml** and edit the property mentioned below inside configuration tag:*

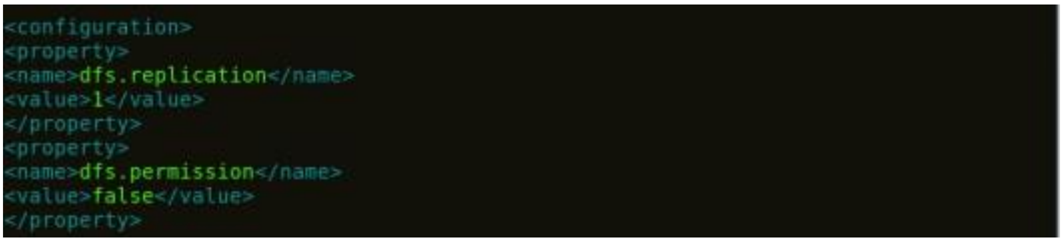
hdfs-site.xml contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

Command: vi hdfs-site.xml



edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ vi hdfs-site.xml
```



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

Fig: Hadoop Installation – Configuring hdfs-site.xml

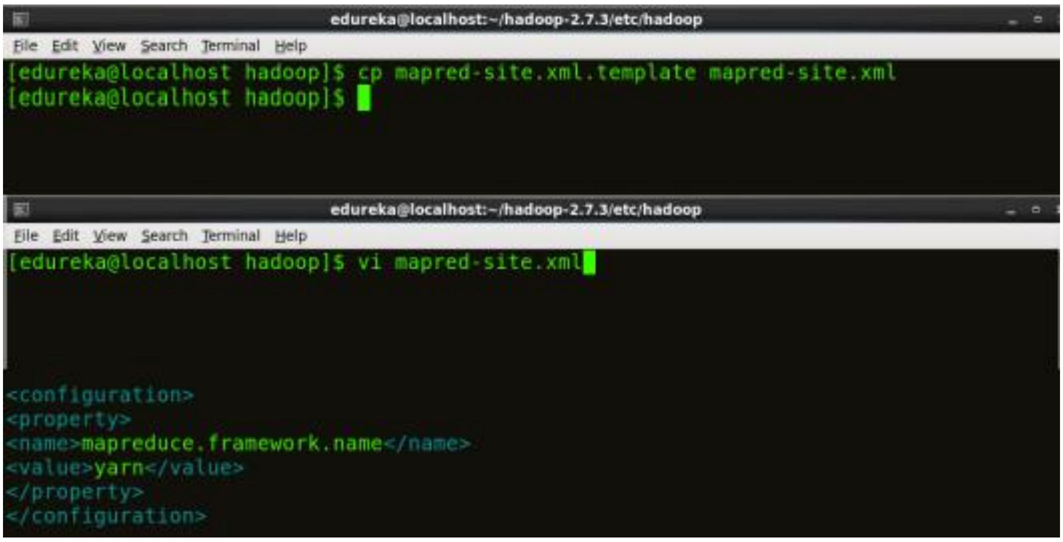
```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permission</name>

<value>false</value>

</property>

</configuration>
```

Step 9: *Edit the mapred-site.xml file and edit the property mentioned below inside configuration tag:* **mapred-site.xml** contains configuration settings of MapReduce applications like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.

In some cases, **mapred-site.xml** file is not available. So, we have to create the **mapred-site.xml** file using the **mapred-site.xml** template.

Command: cp mapred-site.xml.template mapred-site.xml

Command: vi mapred-site.xml.


edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$
```

```
[edureka@localhost hadoop]$ vi mapred-site.xml
```

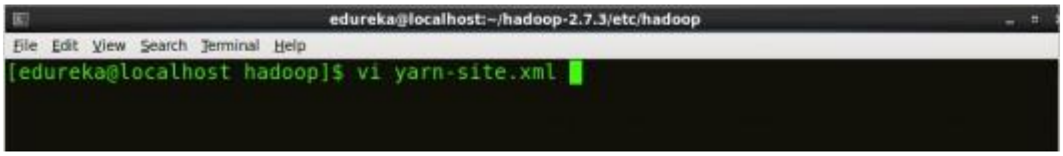
```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Fig: Hadoop Installation – Configuring mapred-site.xml

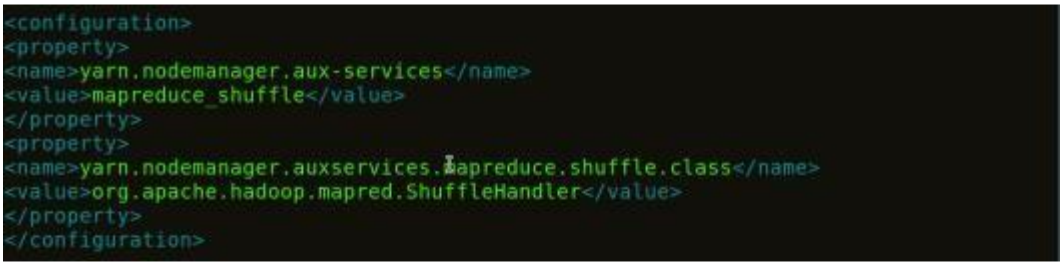
```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>
```

***Step 10:** Edit **yarn-site.xml** and edit the property mentioned below inside configuration tag:*
yarn-site.xml contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

***Command*: vi yarn-site.xml**


edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help

```
[edureka@localhost hadoop]$ vi yarn-site.xml
```



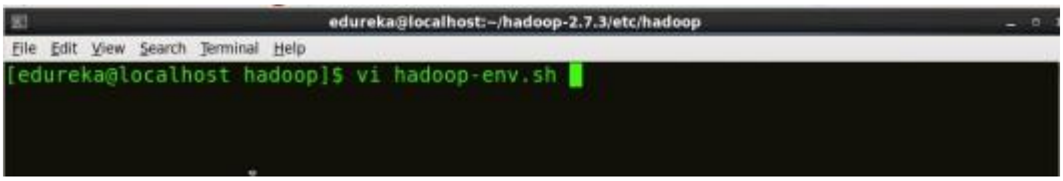
```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Fig: Hadoop Installation – Configuring yarn-site.xml

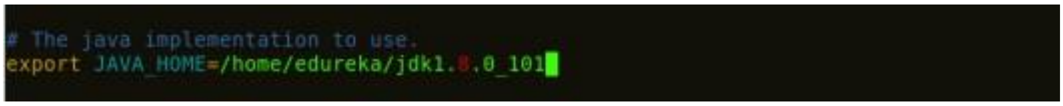
```
<?xml version="1.0">
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

***Step 11:** Edit **hadoop-env.sh** and add the Java Path as mentioned below:*
hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

Command: vi hadoop–env.sh



```
[edureka@localhost hadoop]$ vi hadoop-env.sh
```

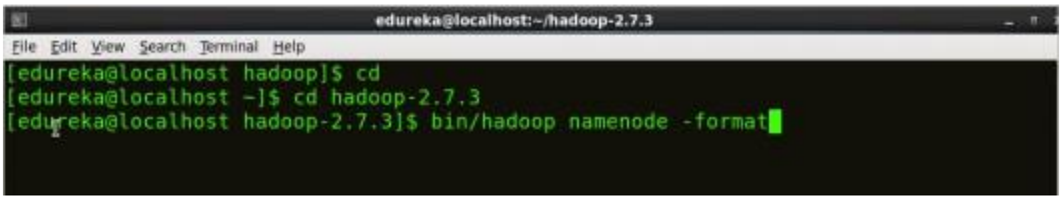


```
# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

Fig: Hadoop Installation – Configuring hadoop-env.sh

Step 12: *Go to the Hadoop home directory and format the NameNode.*

Command: cd
Command: cd hadoop-2.7.3
Command: bin/hadoop namenode -format



```
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

Fig: Hadoop Installation – Formatting NameNode

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the **dfs.name.dir** variable.

Never format, up and running Hadoop filesystem. All stored data will be lost in the HDFS.

Step 13: *Once the NameNode is formatted, go to* ***hadoop-2.7.3/sbin*** *directory and start all the daemons.*

Command: cd hadoop-2.7.3/sbin

Either start all daemons with a single command or do it individually.

Command: ./start-all.sh

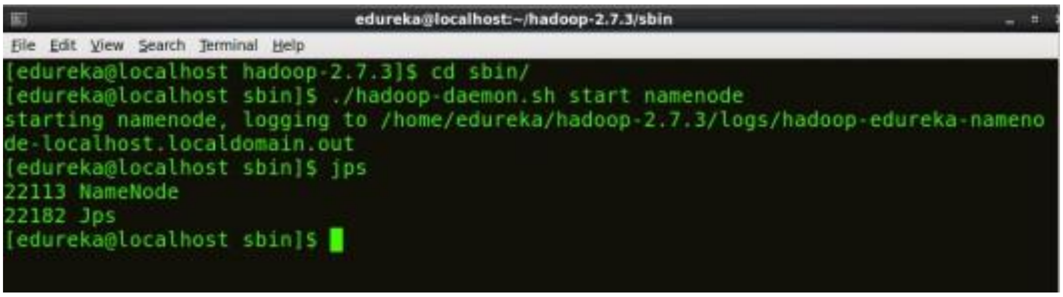
The above command is a combination of **start-dfs.sh, start-yarn.sh** & **mr-jobhistory-daemon.sh**

Or can run all the services individually as below:

Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the files stored across the cluster.

Command: ./hadoop-daemon.sh start namenode



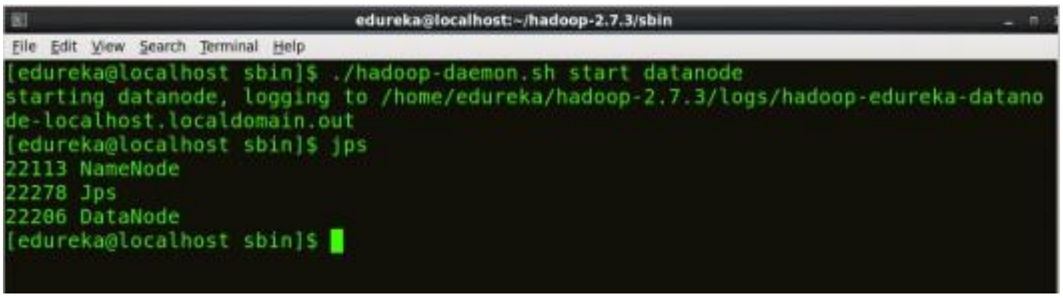
```
[edureka@localhost hadoop-2.7.3]$ cd sbin/
[edureka@localhost sbin]$ ./hadoop-daemon.sh start namenode
starting namenode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-nameno
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22182 Jps
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting NameNode

Start DataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.

Command: ./hadoop-daemon.sh start datanode



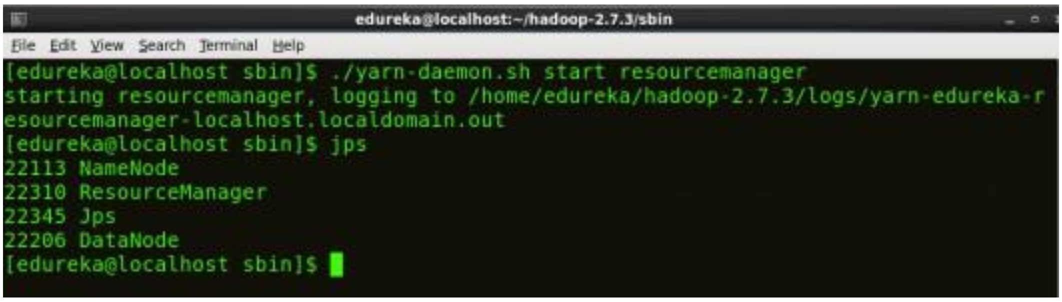
```
[edureka@localhost sbin]$ ./hadoop-daemon.sh start datanode
starting datanode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datano
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22278 Jps
22206 DataNode
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting DataNode

Start ResourceManager:

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and each application’s ApplicationMaster.

Command: ./yarn-daemon.sh start resourcemanager



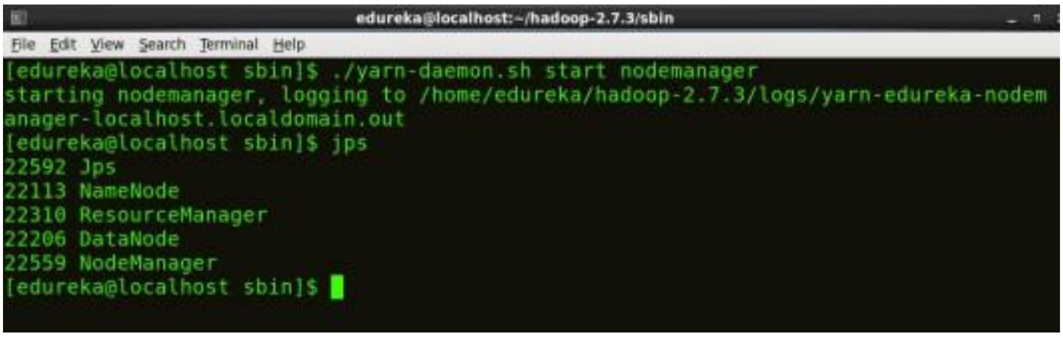
```
[edureka@localhost sbin]$ ./yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-r
esourcemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22345 Jps
22206 DataNode
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting ResourceManager

Start NodeManager:

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

Command: ./yarn-daemon.sh start nodemanager


edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help

```
[edureka@localhost sbin]$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-nodem
anager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22592 Jps
22113 NameNode
22310 ResourceManager
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

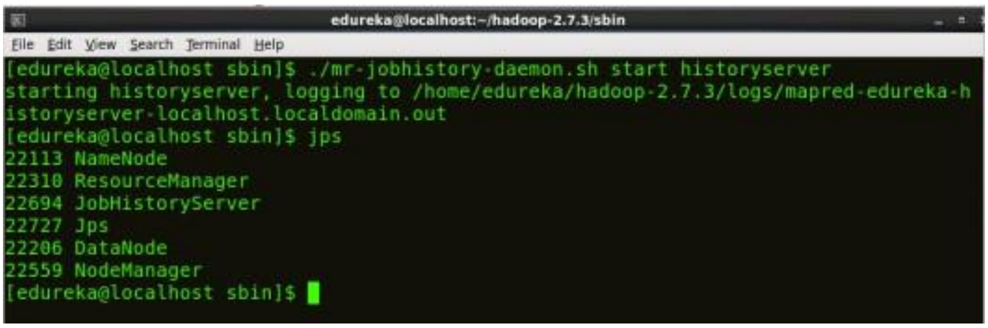
Fig: Hadoop Installation – Starting NodeManager

Start JobHistoryServer:
JobHistoryServer is responsible for servicing all job history related requests from client.

Command: ./mr-jobhistory-daemon.sh start historyserver

Step 14: *To check that all the Hadoop services are up and running, run the below command.*

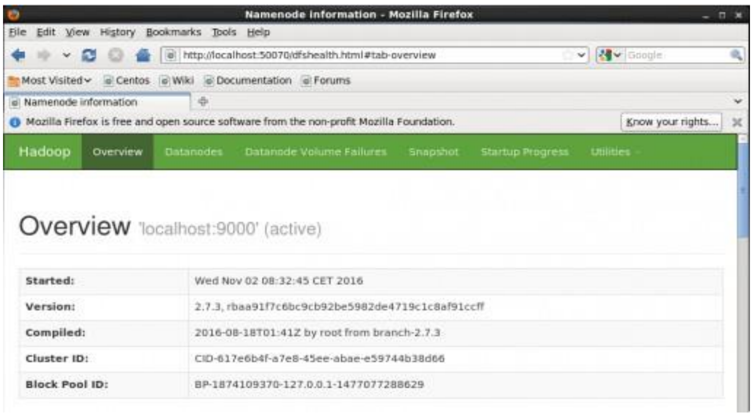
Command: jps


edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help

```
[edureka@localhost sbin]$ ./mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/edureka/hadoop-2.7.3/logs/mapred-edureka-h
istoryserver-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22694 JobHistoryServer
22727 Jps
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Checking Daemons

Step 15: *Now open the Mozilla browser and go to* ***localhost:50070/dfshealth.html*** *to check the NameNode interface.*


Namenode information - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:50070/dfshealth.html#tab-overview
Google
Most Visited
Centos
Wiki
Documentation
Forums
Namenode information
Mozilla Firefox is free and open source software from the non-profit Mozilla Foundation.
Know your rights...
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'localhost:9000' (active)

Started:	Wed Nov 02 08:32:45 CET 2016
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-617e6b4f-a7e8-45ee-abae-e59744b38d66
Block Pool ID:	BP-1874109370-127.0.0.1-1477077288629

Fig: Hadoop Installation – Starting WebUI

RESULT:

Thus a Single-Node Hadoop Cluster has been configured successfully.

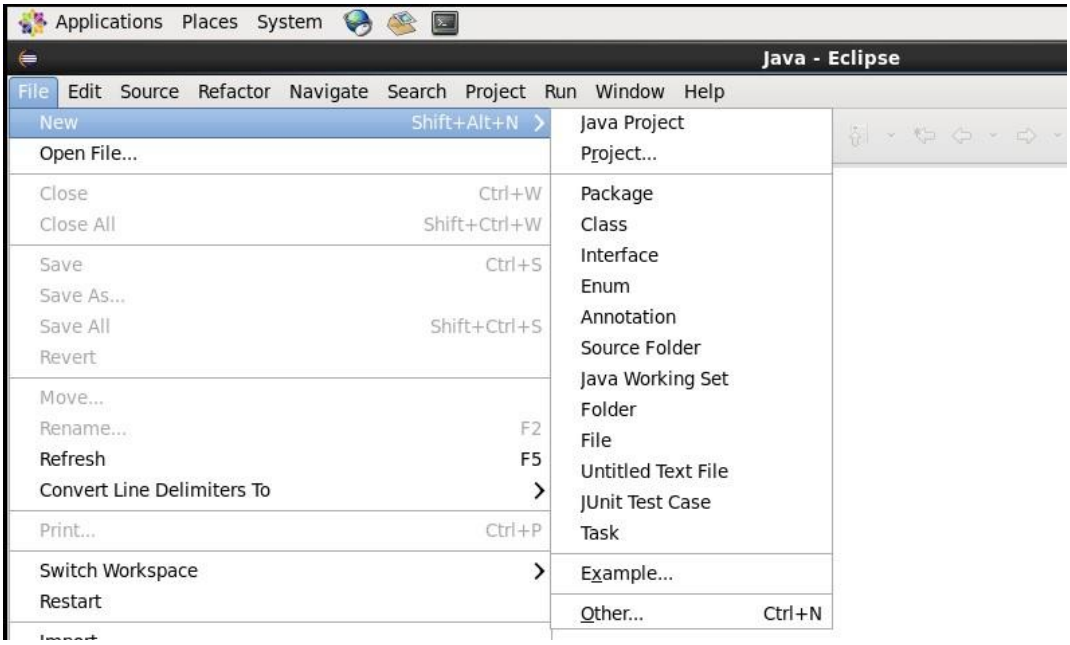
Ex. No: 11 **WORD COUNT IN HADOOP**

Date: 27.03.2024

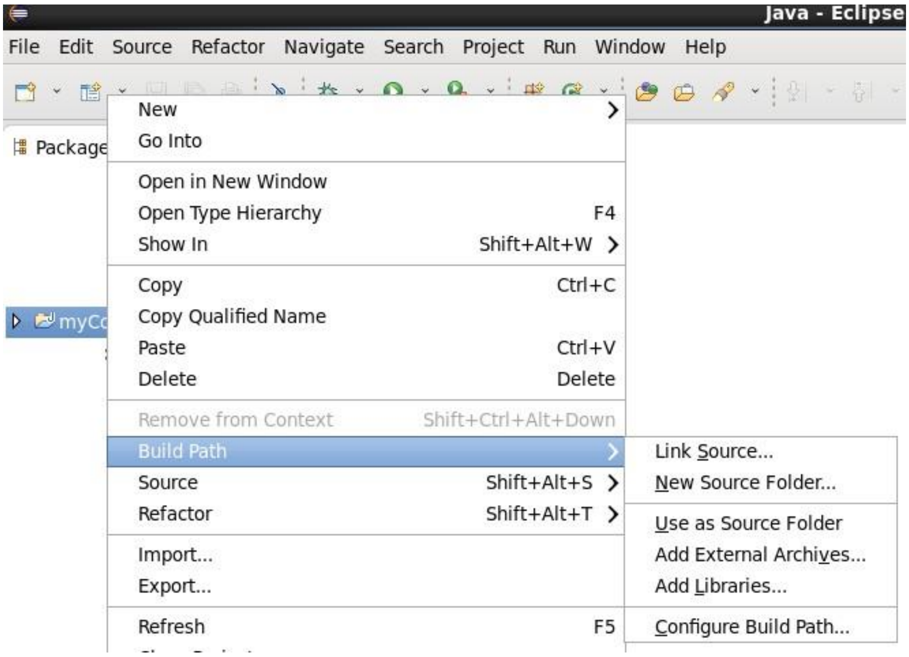
AIM: To run the word count program in Hadoop.

STEPS:

First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** ->Name it **WordCount** -> then **Finish**.



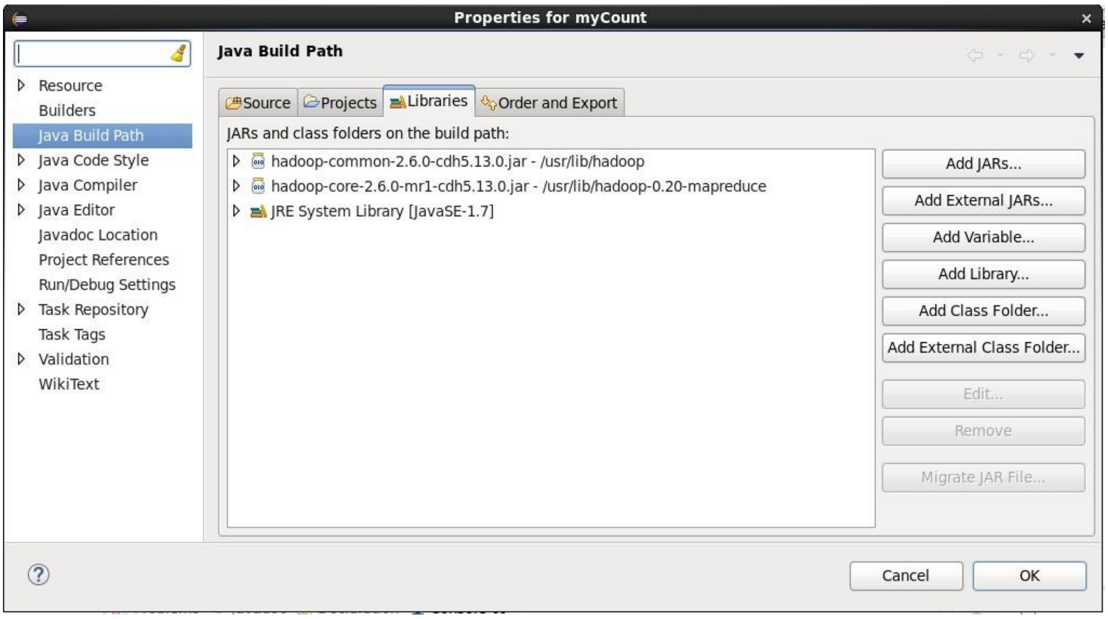
- Create 3 Java Classes into the project. Name them **WCDriver**(having the main function), **WCMapper**, **WCReducer**.
- Include two Reference Libraries for that:
  Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**



In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mentioned files. You can find these files in */usr/lib/*

1. /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar

2. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar



Properties for myCount

Java Build Path

JARs and class folders on the build path:

- hadoop-common-2.6.0-cdh5.13.0.jar - /usr/lib/hadoop
- hadoop-core-2.6.0-mr1-cdh5.13.0.jar - /usr/lib/hadoop-0.20-mapreduce
- JRE System Library [JavaSE-1.7]

Mapper Code: Copy and paste this program into the WCMapper Java Class file.
// Importing libraries

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements Mapper<LongWritable,
                                                                                    Text, Text,
IntWritable> {
   // Map function
   public void map(LongWritable key, Text value, OutputCollector<Text,
                              IntWritable> output, Reporter rep) throws IOException
   {
          String line = value.toString();
          // Splitting the line on spaces
          for (String word : line.split(" "))
          {
                 if (word.length() > 0)
                 {
                        output.collect(new Text(word), new IntWritable(1));
                 }
          }
```

```
    }
}
```

Reducer Code: Copy paste this program into the WCReducer Java Class file.

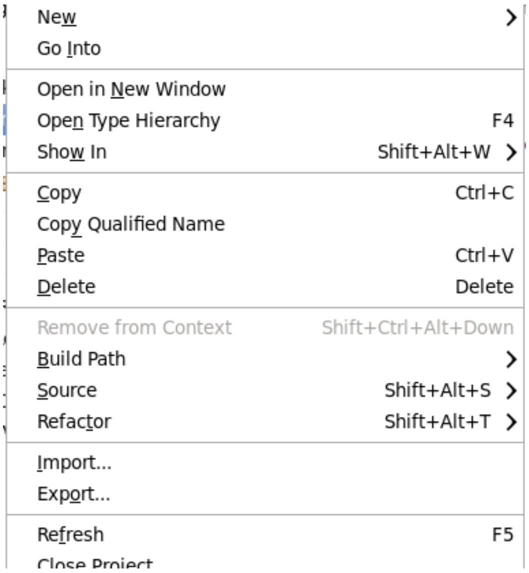
```
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import  org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
 public class WCReducer extends MapReduceBase implements Reducer<Text,
                                                    IntWritable, Text, IntWritable> {
    // Reduce function
    public void reduce(Text key, Iterator<IntWritable> value,
                        OutputCollector<Text, IntWritable> output,
                        Reporter rep) throws IOException
    {
        int count = 0;
        // Counting the frequency of each words
        while (value.hasNext())
        {
            IntWritable i = value.next();
            count += i.get();
        }
        output.collect(key, new IntWritable(count));
    }
}
```

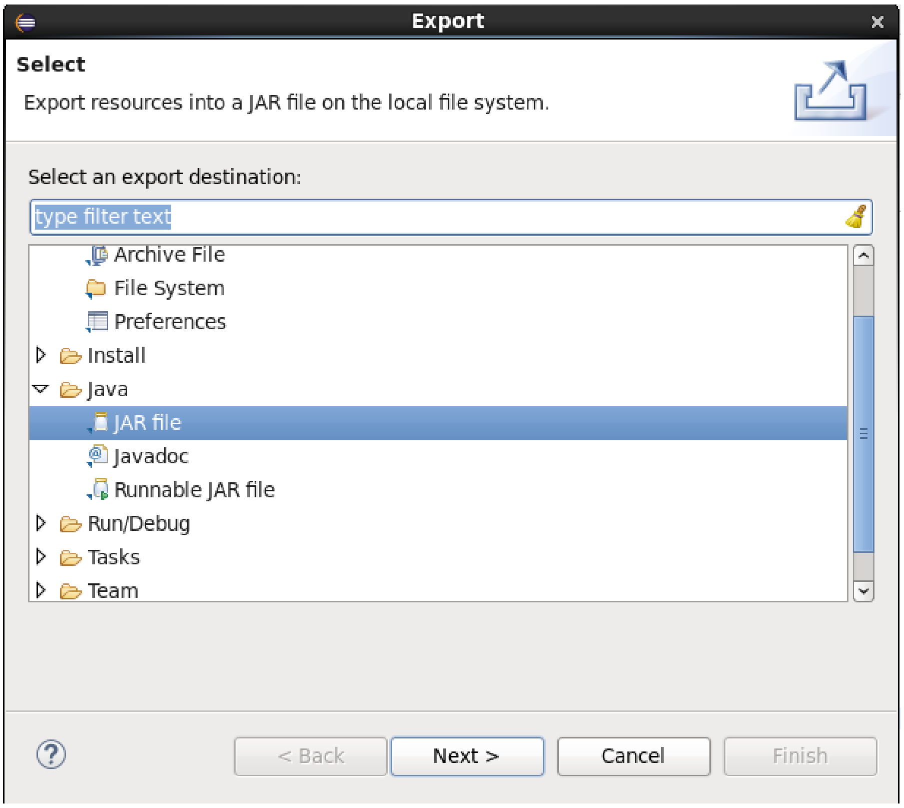
Driver Code: Copy paste this program into the WCDriver Java Class file.

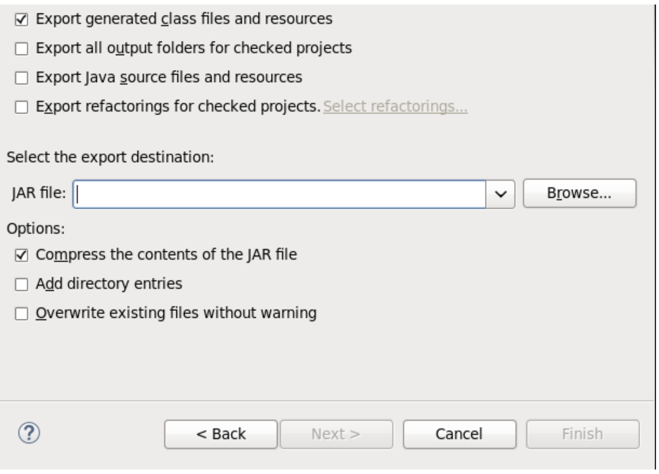
```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException
    {
        if (args.length < 2)
```

```
            {
                        System.out.println("Please give valid inputs");
                        return -1;
            }
            JobConf conf = new JobConf(WCDriver.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            conf.setMapperClass(WCMapper.class);
            conf.setReducerClass(WCReducer.class);
            conf.setMapOutputKeyClass(Text.class);
            conf.setMapOutputValueClass(IntWritable.class);
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            JobClient.runJob(conf);
            return 0;
    }
    // Main Method
    public static void main(String args[]) throws Exception
    {
            int exitCode = ToolRunner.run(new WCDriver(), args);
            System.out.println(exitCode);
    }
}
```

Now you have to make a jar file. Right Click on **Project**-> **Click on Export**-> **Select export destination as Jar File**-> **Name the jar File**(WordCount.jar) -> **Click on next** -> at last **Click on Finish**. Now copy this file into the Workspace directory.







☑ Export generated class files and resources
☐ Export all output folders for checked projects
☐ Export Java source files and resources
☐ Export refactorings for checked projects. Select refactorings...

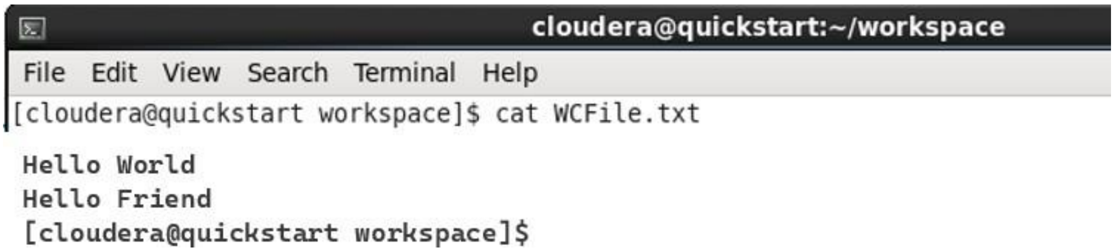
Select the export destination:

JAR file:

Options:

☑ Compress the contents of the JAR file
☐ Add directory entries
☐ Overwrite existing files without warning

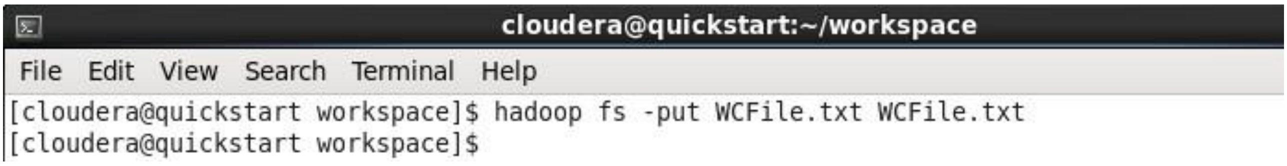
Open the terminal on CDH and change the directory to the workspace. This can be done by using the “cd workspace/” command. Now, Create a text file(**WCFile.txt**) and move it to HDFS. Open the terminal and write this code (remember you should be in the same directory as jar file you have created just now).



```
[cloudera@quickstart workspace]$ cat WCFile.txt

Hello World
Hello Friend
[cloudera@quickstart workspace]$
```

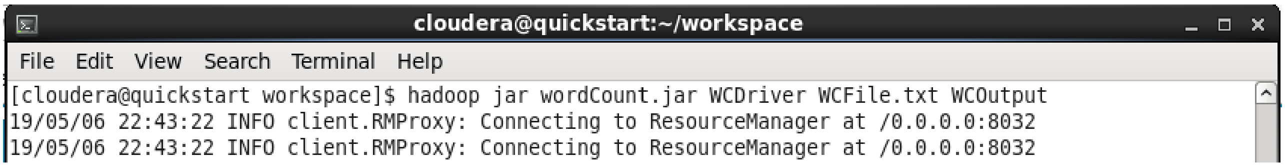
- Now, run this command to copy the file input file into the HDFS.

```
hadoop fs -put WCFile.txt WCFile.txt
```


cloudera@quickstart:~/workspace
File Edit View Search Terminal Help

```
[cloudera@quickstart workspace]$ hadoop fs -put WCFile.txt WCFile.txt
[cloudera@quickstart workspace]$
```

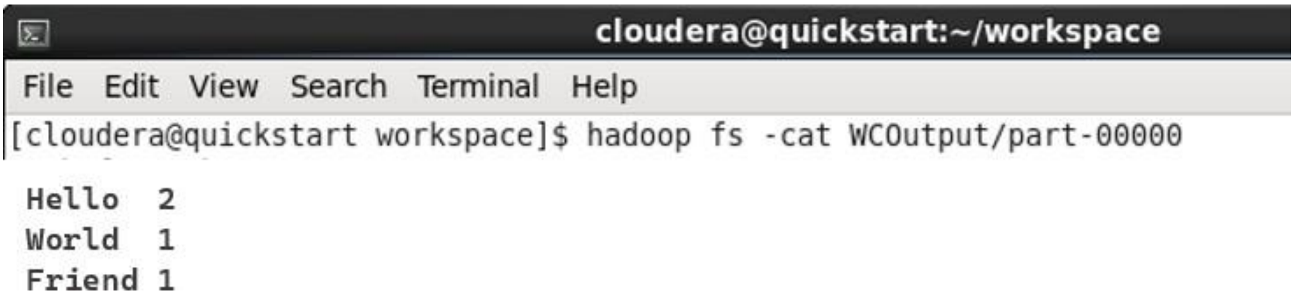
Now to run the jar file by writing the code as shown in the screenshot.


cloudera@quickstart:~/workspace
File Edit View Search Terminal Help

```
[cloudera@quickstart workspace]$ hadoop jar wordCount.jar WCDriver WCFile.txt WCOutput
19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
```

- After Executing the code, the result can be viewed in the WCOutput file or by writing the following command on terminal.

```
hadoop fs -cat WCOutput/part-00000
```


cloudera@quickstart:~/workspace
File Edit View Search Terminal Help

```
[cloudera@quickstart workspace]$ hadoop fs -cat WCOutput/part-00000

Hello  2
World  1
Friend 1
```

RESULT:

Thus the program has been executed successfully.

Ex. No: 12
Date: 17.04.2024

DEPLOYMENT OF MICROSERVICES USING KUBERNETES IN GOOGLE CLOUD

AIM: To deploy a microservices architecture using a container orchestration tool like Kubernetes or Docker Swarm.

STEPS:

The steps to deploy a microservice architecture using Kubernetes on Gcloud are:

1. Create microservice to be deployed
2. Place application in your docker container
3. Create a new Kubernetes project
4. Create new Cluster
5. Allow access from your local machine
6. Create service account
7. Activate service account
8. Connect to cluster
9. Gcloud initialization
10. Generate access token
11. Deploy and start Kubernetes dashboard
12. Deploy microservice

Step-1: Create a microservice to be deployed. We can use **https://start.spring.io/** for this goal. Create HelloController like this:

```
package com.example.demojooq.controllers;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/api/v1")
public class HelloController {
@GetMapping("/say-hello")
public String sayHello() {
       return "Hello world";
   }
}
```

Step-2: Place application in your docker container.

We have a microservice, need to put this microservice in a docker container and upload it on Kubernetes. Now, Kubernetes will orchestrate the container according to your settings. Let us create the first image from the microservice. It is called Dockerfile (without any extension), and the content is:

Dockerfile

```
FROM adoptopenjdk/openjdk11:jre-11.0.8_10-debianslim
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
```

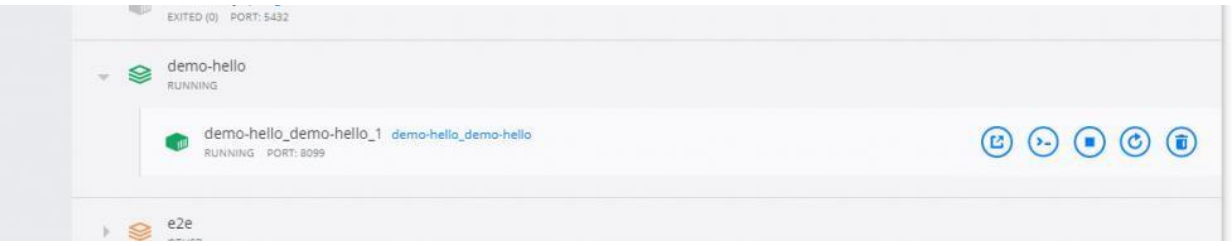
```
ENTRYPOINT ["java","-jar","app.jar"]
```

The next step is to create the docker-compose file. For that purpose, a call to Dockerfile will be made to build the image. You can do it manually, but the best way is from the docker-compose file, as you have a permanent track of the solution. This is a .yaml file. (picture below)

Docker-compose.yaml

```
version: "3"
services:
hello-world:
   build: .
   ports:
     - "8099:8080"
```

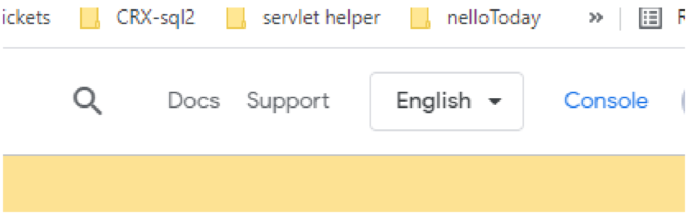
After starting docker, go to the folder where docker-compose is located and execute the command “docker-compose up”. The expectation is to reach this microservice on the 8099 port. If everything is ok, in your docker will be something like this:



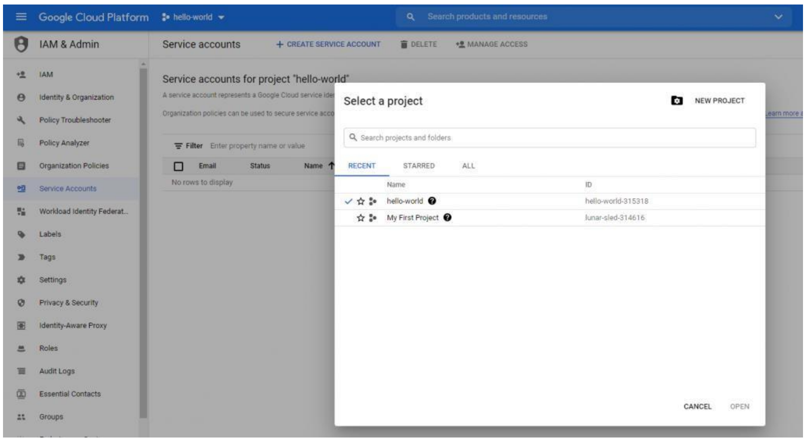
Check microservice docker installation with postman calling http://localhost:8099/api/v1/say-hello. In response, you have “Hello World”.

Step-3: Create a Kubernetes project.

Open up your Google account, sign in and go to the console.



Create a new project from the main dashboard; the name of the new project is “hello-world”. From now on, this is your active project.



Service accounts for project "hello-world"

A service account represents a Google Cloud service ide

Organization policies can be used to secure service acco

Filter Enter property name or value

□	Email	Status	Name ↑

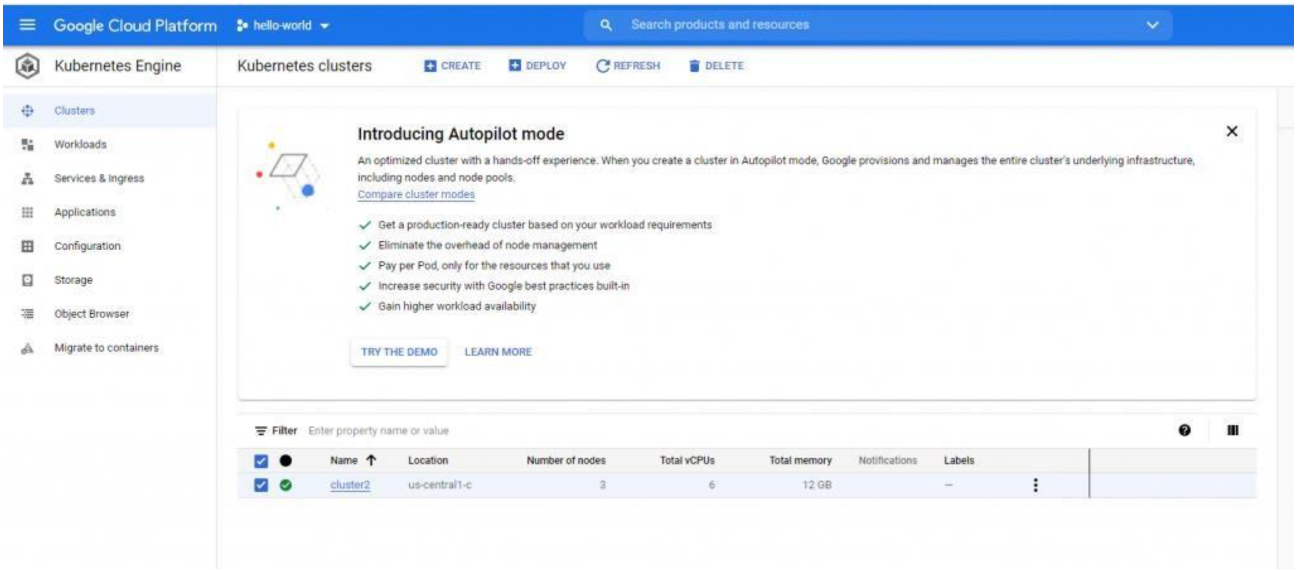
No rows to display

Select a project

	Name	ID
✓ ☆	hello-world	hello-world-315318
☆	My First Project	lunar-sled-314616

Step 4: Create new cluster

Create a new cluster (named it cluster2). Accept default values for other fields.



Introducing Autopilot mode

An optimized cluster with a hands-off experience. When you create a cluster in Autopilot mode, Google provisions and manages the entire cluster's underlying infrastructure, including nodes and node pools.

Compare cluster modes

- ✓ Get a production-ready cluster based on your workload requirements
- ✓ Eliminate the overhead of node management
- ✓ Pay per Pod, only for the resources that you use
- ✓ Increase security with Google best practices built-in
- ✓ Gain higher workload availability

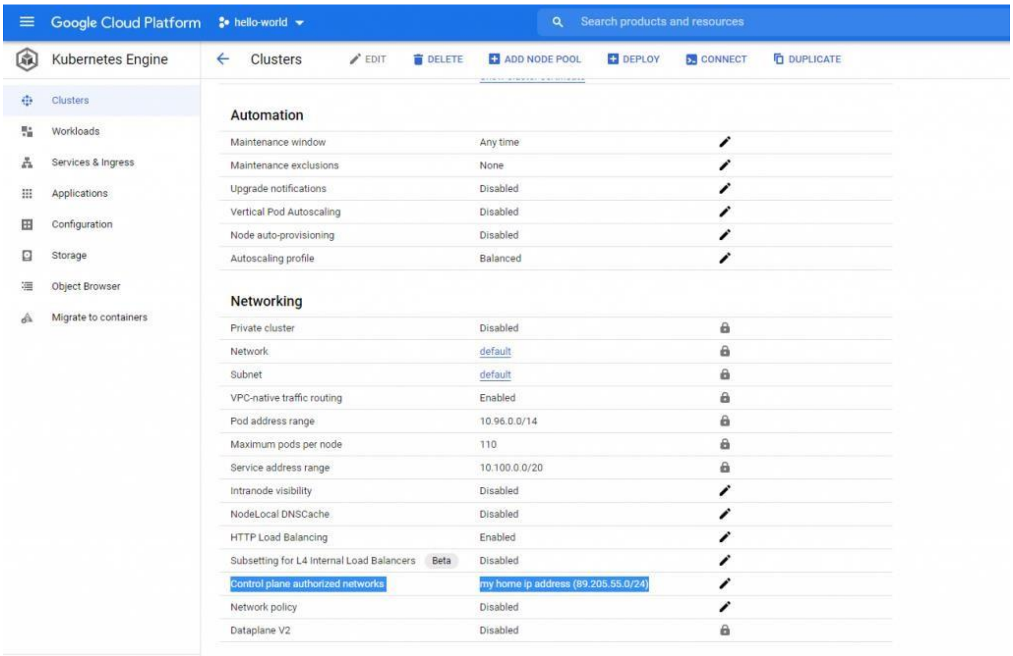
Filter Enter property name or value

☑	●	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☑	✓	cluster2	us-central1-c	3	6	12 GB		–

Step 5: Allow access from your local machine

Now, we must allow access from our local machine to Kubernetes, via kubectl. For that purpose, we need to follow these steps:

1. Click on cluster2
2. Find your local IP address and add it here according to the CIDR standard in the Edit control plane authorized networks



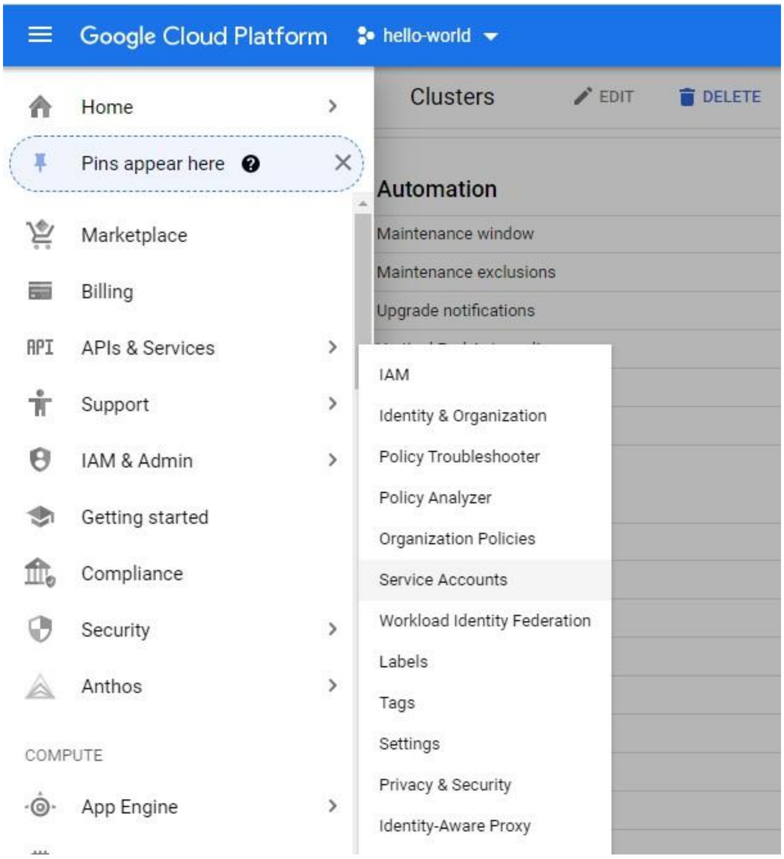
Automation

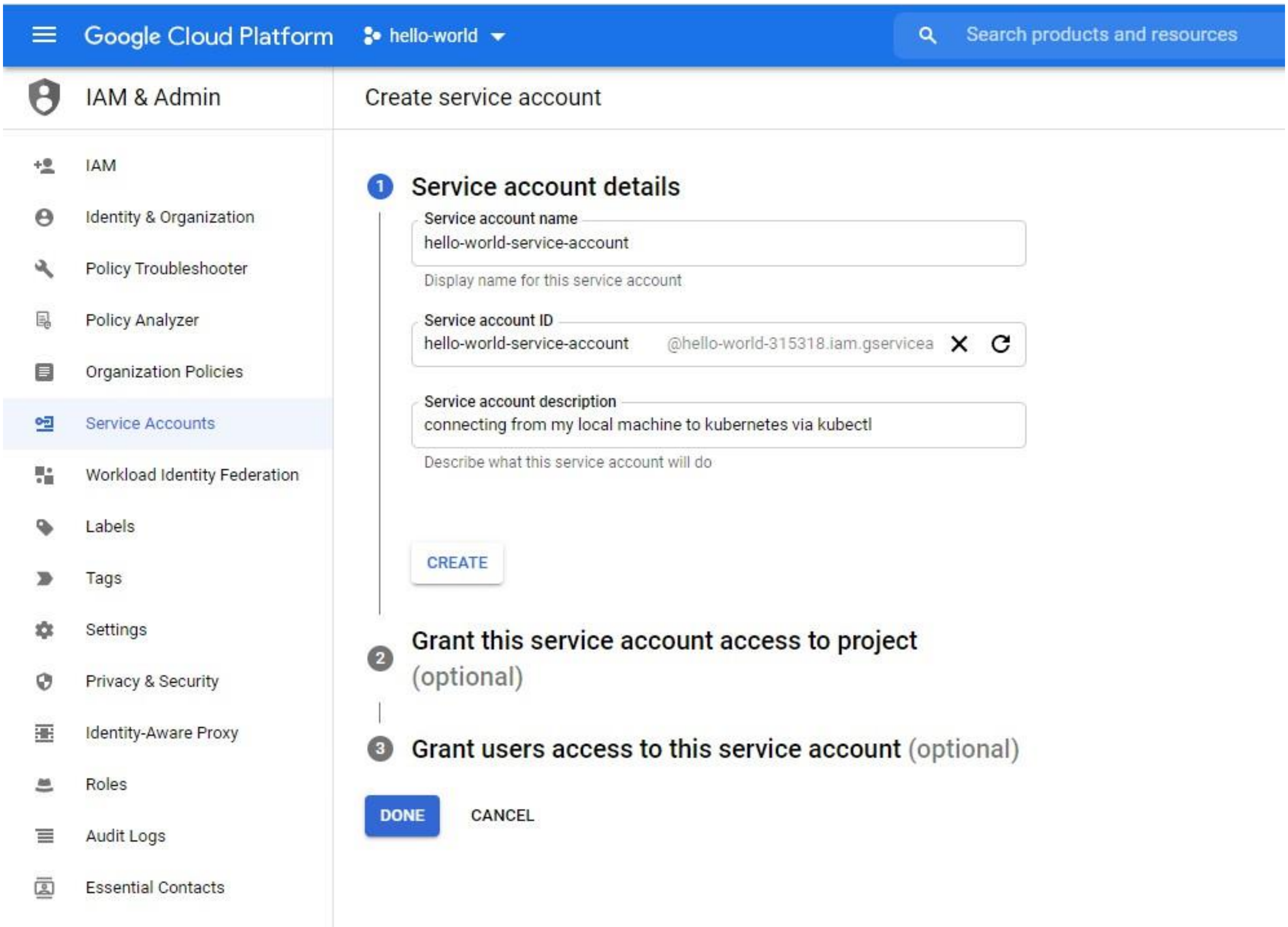
Maintenance window	Any time
Maintenance exclusions	None
Upgrade notifications	Disabled
Vertical Pod Autoscaling	Disabled
Node auto-provisioning	Disabled
Autoscaling profile	Balanced

Networking

Private cluster	Disabled
Network	default
Subnet	default
VPC-native traffic routing	Enabled
Pod address range	10.96.0.0/14
Maximum pods per node	110
Service address range	10.100.0.0/20
Intranode visibility	Disabled
NodeLocal DNSCache	Disabled
HTTP Load Balancing	Enabled
Subsetting for L4 Internal Load Balancers Beta	Disabled
Control plane authorized networks	my home ip address (89.205.55.0/24)
Network policy	Disabled
Dataplane V2	Disabled

Step 6: Create service account





Create service account

1 Service account details

Service account name: hello-world-service-account

Display name for this service account

Service account ID: hello-world-service-account @hello-world-315318.iam.gservicea

Service account description: connecting from my local machine to kubernetes via kubectl

Describe what this service account will do

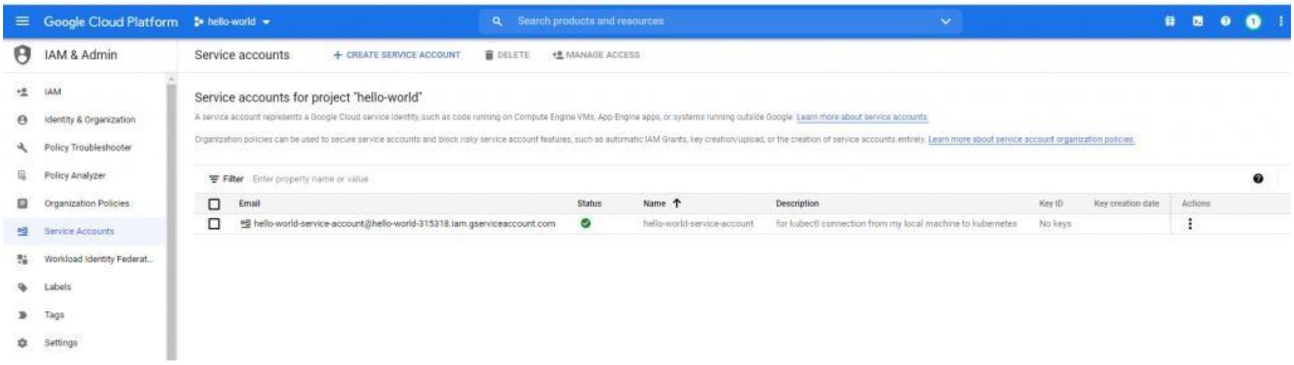
CREATE

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

DONE CANCEL

Give a new account role “Owner”. Accept default values for other fields. After a service account is created, you should have something like this:



Service accounts

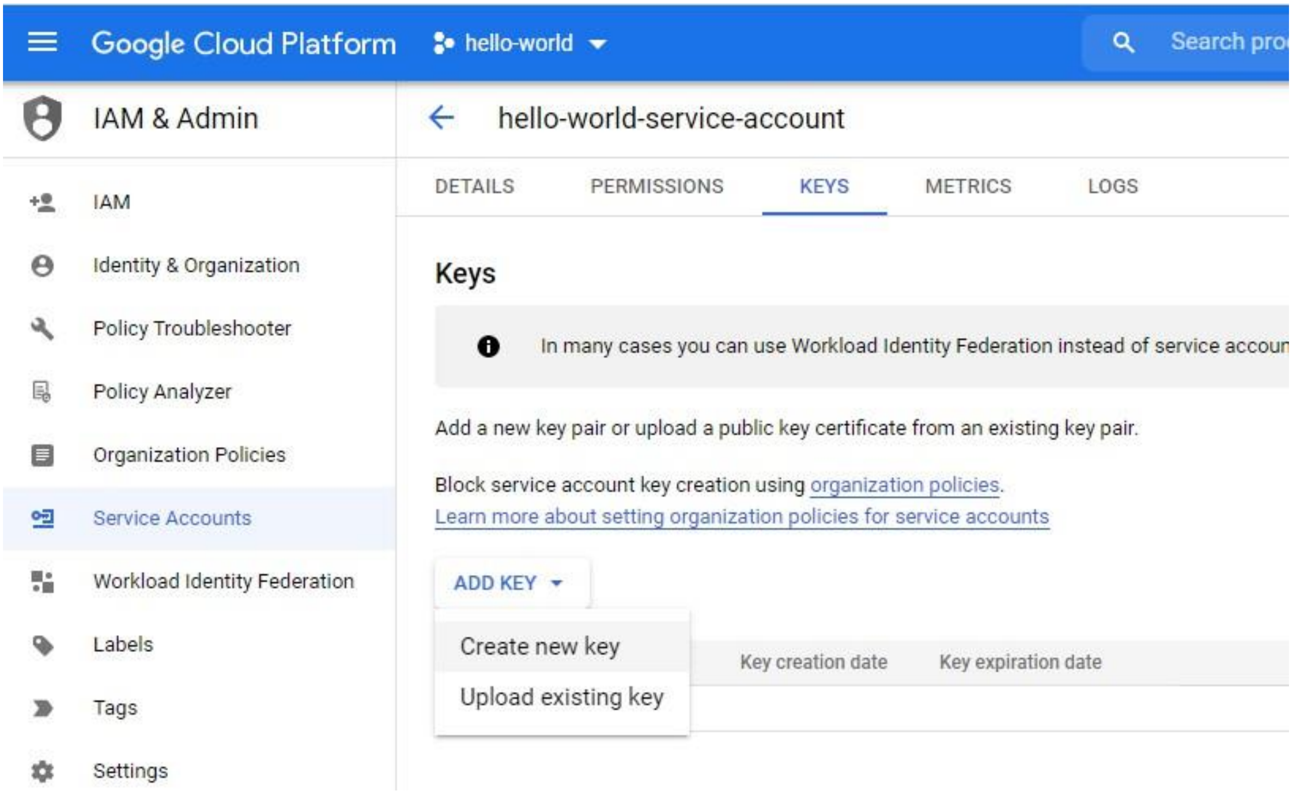
Service accounts for project "hello-world"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.

□	Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
□	hello-world-service-account@hello-world-315318.iam.gserviceaccount.com	✓	hello-world-service-account	for kubectl connection from my local machine to kubernetes	No keys		⋮

Generate keys for this service account with key type JSON. When the key is downloaded, it has some random name like hello-world-315318-ab0c74d58a70.json. Keep this file in a safe place, we will need it later.



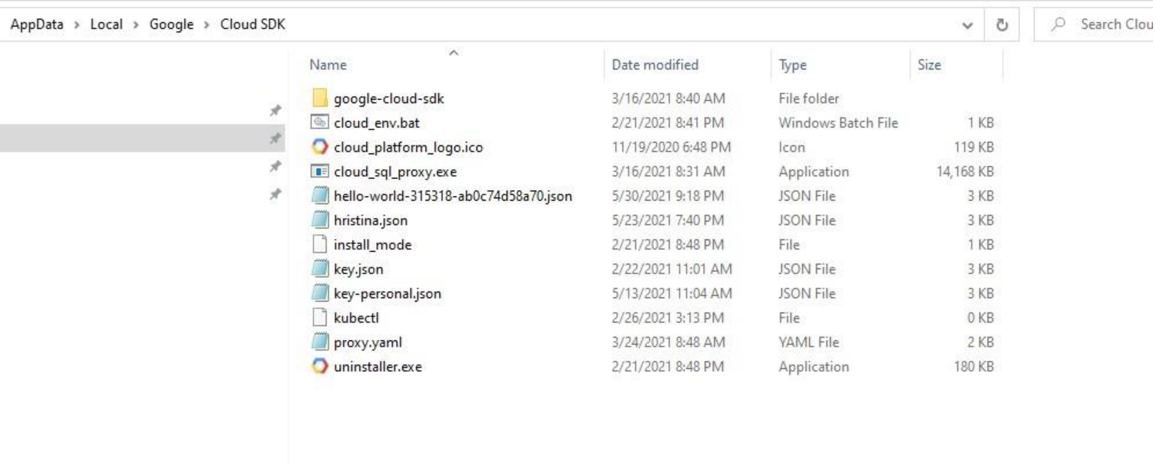
Keys

In many cases you can use Workload Identity Federation instead of service accoun

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts

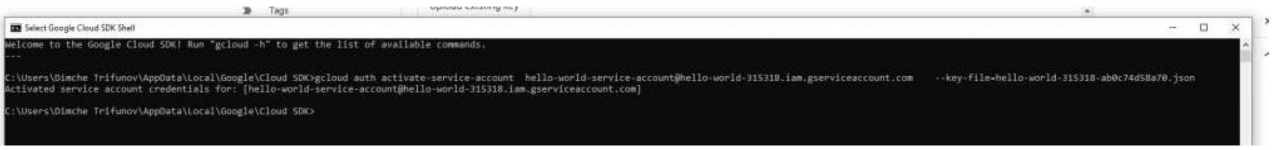
Now, install Google Cloud SDK Shell on your machine according to your OS. Let’s do the configuration so kubectl can reach cluster2. Copy the file hello-world-315318-ab0c74d58a70.json and put it in the CLOUD SDK folder. For the Windows environment, it looks like this:



Name	Date modified	Type	Size
google-cloud-sdk	3/16/2021 8:40 AM	File folder	
cloud_env.bat	2/21/2021 8:41 PM	Windows Batch File	1 KB
cloud_platform_logo.ico	11/19/2020 6:48 PM	Icon	119 KB
cloud_sql_proxy.exe	3/16/2021 8:31 AM	Application	14,168 KB
hello-world-315318-ab0c74d58a70.json	5/30/2021 9:18 PM	JSON File	3 KB
hristina.json	5/23/2021 7:40 PM	JSON File	3 KB
install_mode	2/21/2021 8:48 PM	File	1 KB
key.json	2/22/2021 11:01 AM	JSON File	3 KB
key-personal.json	5/13/2021 11:04 AM	JSON File	3 KB
kubectl	2/26/2021 3:13 PM	File	0 KB
proxy.yaml	3/24/2021 8:48 AM	YAML File	2 KB
uninstaller.exe	2/21/2021 8:48 PM	Application	180 KB

Step 7: Activate service account

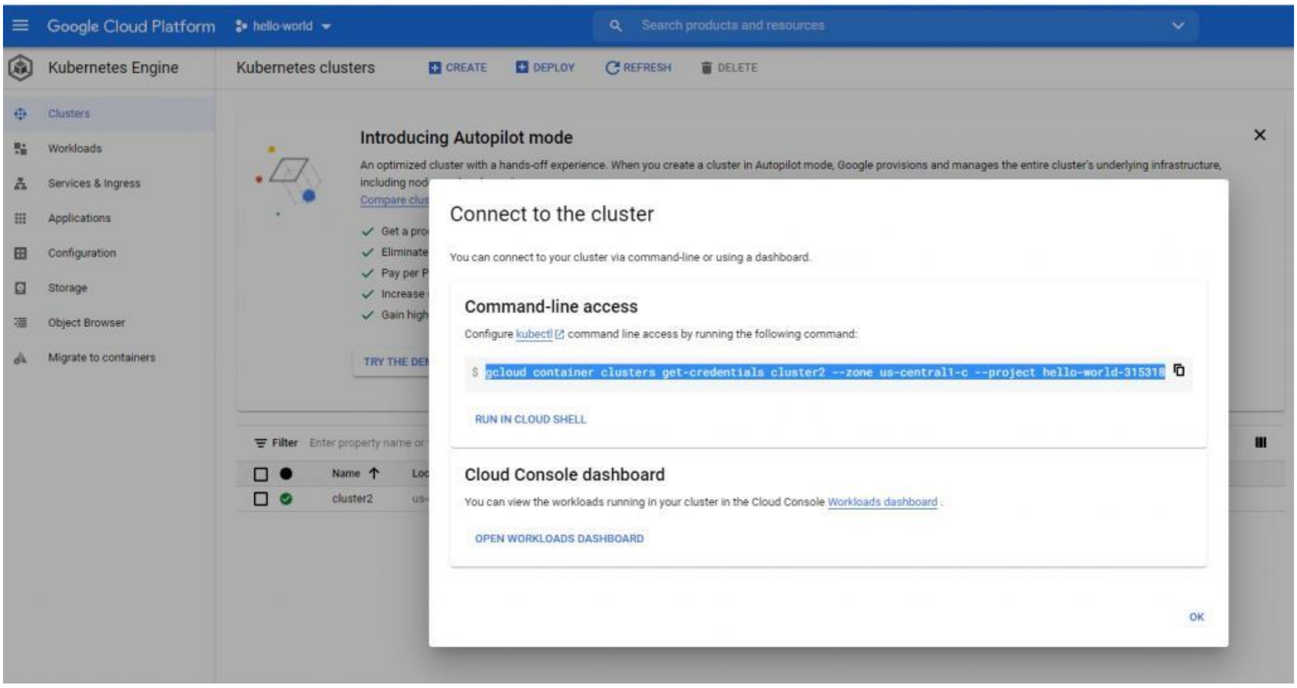
The first thing to do is to activate the service account with the command: gcloud auth activate-service-account hello-world-service-account@hello-world-315318.iam.gserviceaccount.com –key-file=hello-world-315318-ab0c74d58a70.json



```
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
---

C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>gcloud auth activate-service-account  hello-world-service-account@hello-world-315318.iam.gserviceaccount.com    --key-file=hello-world-315318-ab0c74d58a70.json
Activated service account credentials for: [hello-world-service-account@hello-world-315318.iam.gserviceaccount.com]

C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>
```

Step 8: Connect to cluster

Now go to cluster2 again and find the connection string to connect to the new cluster



An optimized cluster with a hands-off experience. When you create a cluster in Autopilot mode, Google provisions and manages the entire cluster's underlying infrastructure, including nod

Connect to the cluster

You can connect to your cluster via command-line or using a dashboard.

Command-line access

Configure kubectl command line access by running the following command:

```
$ gcloud container clusters get-credentials cluster2 --zone us-central1-c --project hello-world-315318
```

RUN IN CLOUD SHELL

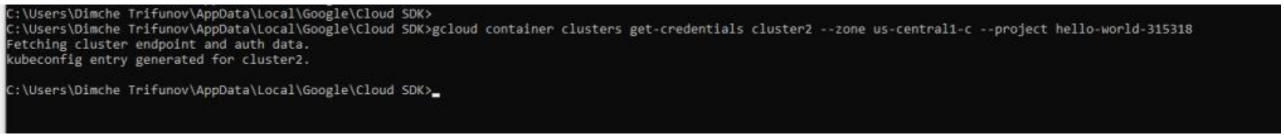
Cloud Console dashboard

You can view the workloads running in your cluster in the Cloud Console Workloads dashboard .

OPEN WORKLOADS DASHBOARD

OK

Execute this connection string in Google Cloud Shell: gcloud container clusters get-credentials cluster2 –zone us-central1-c –project hello-world-315318



```
C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>
C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>gcloud container clusters get-credentials cluster2 --zone us-central1-c --project hello-world-315318
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster2.

C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>_
```

Step 9: Gcloud initialization

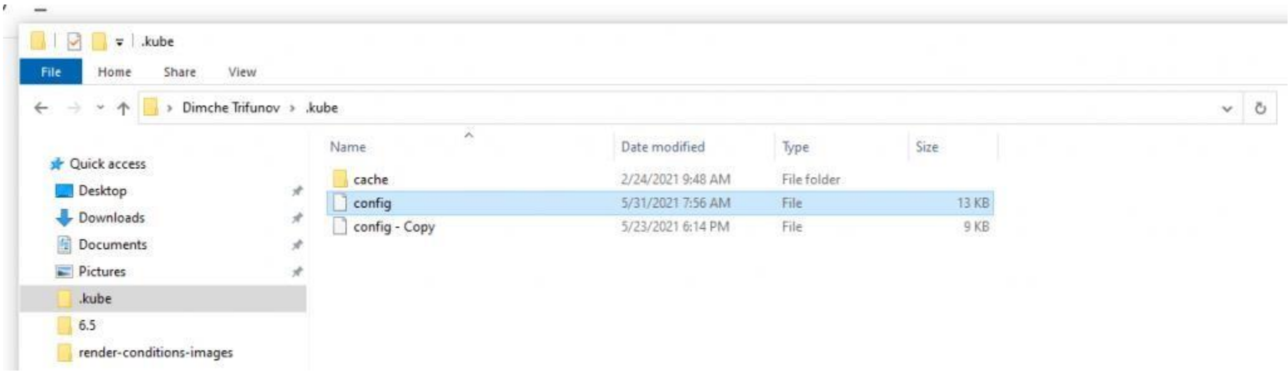
The next command to execute is gcloud init, to initialize connection with the new project. Here is the complete code on how to do that from the Gcloud Shell:

```
C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [dev] are:
accessibility:
  screen_reader: 'False'
compute:
  region:  europe-west3
zone:   europe-west3-a
core:
  account: hello-world-service-account@hello-world-315318.iam.gserviceaccount.com
  disable_usage_reporting: 'True'
  project: dops-containers

Pick configuration to use:
 [1] Re-initialize this configuration [dev] with new settings
 [2] Create a new configuration
 [3] Switch to and re-initialize existing configuration: [database-connection]
 [4] Switch to and re-initialize existing configuration: [default]
Please enter your numeric choice:  2

Enter configuration name. Names start with a lower case letter and
contain only lower case letters a-z, digits 0-9, and hyphens '-':  hello-world
Your current configuration has been set to: [hello-world]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for
this configuration:
 [1] cicd-worker@devops-platform-n47.iam.gserviceaccount.com
 [2] d.trifunov74@gmail.com
 [3] dimche.trifunov@north-47.com
 [4] dtrifunov@lunar-sled-314616.iam.gserviceaccount.com
 [5] hello-world-service-account@hello-world-315318.iam.gserviceaccount.com
 [6] service-account-demo-dime@blissful-epoch-305214.iam.gserviceaccount.com
 [7] Log in with a new account
Please enter your numeric choice:  5

You are logged in as: [hello-world-service-account@hello-world-315318.iam.gserviceaccount.com].

API [cloudresourcemanager.googleapis.com] not enabled on project
[580325979968]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [cloudresourcemanager.googleapis.com] on project [580325979968]...
Operation "operations/acf.p2-580325979968-f1bf2515-deea-49d5-ae35-a0adfef9973e" finished
successfully.
Pick cloud project to use:
 [1] hello-world-315318
 [2] Create a new project
Please enter numeric choice or text value (must exactly match list item):  1

Your current project has been set to: [hello-world-315318].

Do you want to configure a default Compute Region and Zone? (Y/n)?  n

Error creating a default .boto configuration file. Please run [gsutil config -n] if you would like to create
this file.
Your Google Cloud SDK is configured and ready to use!

* Commands that require authentication will use hello-world-service-account@hello-world-
315318.iam.gserviceaccount.com by default
```

```
* Commands will reference project `hello-world-315318` by default
Run `gcloud help config` to learn how to change individual settings

This gcloud configuration is called [hello-world]. You can create additional configurations if you work
with multiple accounts and/or projects.
Run `gcloud topic configurations` to learn more.

Some things to try next:

* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help
COMMAND` to get help on any gcloud command.
* Run `gcloud topic --help` to learn about advanced features of the SDK like arg files and output
formatting
```

Step 10: Generate access token

Type kubectl get namespace, access token is generated in .kube folder (in home folder), in config file:



Name	Date modified	Type	Size
cache	2/24/2021 9:48 AM	File folder	
config	5/31/2021 7:56 AM	File	13 KB
config - Copy	5/23/2021 6:14 PM	File	9 KB

If you open this config file, you will find your access token. You will need this later.

Step 11: Deploy and start Kubernetes dashboard

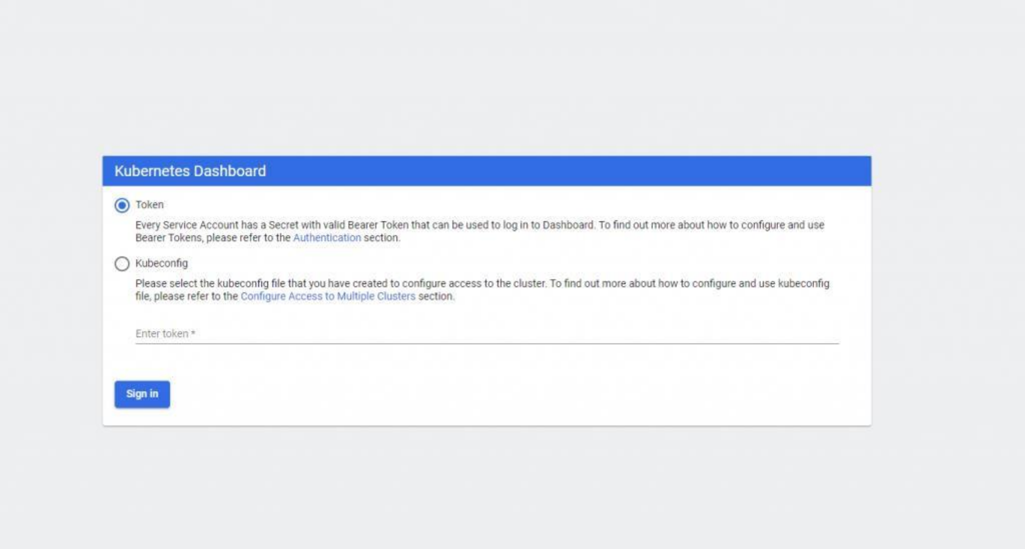
Now, deploy Kubernetes dashboard with the next command: kubectl apply -f

```
C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

```
C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK>kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Start the dashboard with kubectl proxy command. Now open the dashboard from the link: http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/overview?namespace=default

This screen will appear:



Kubernetes Dashboard

● Token

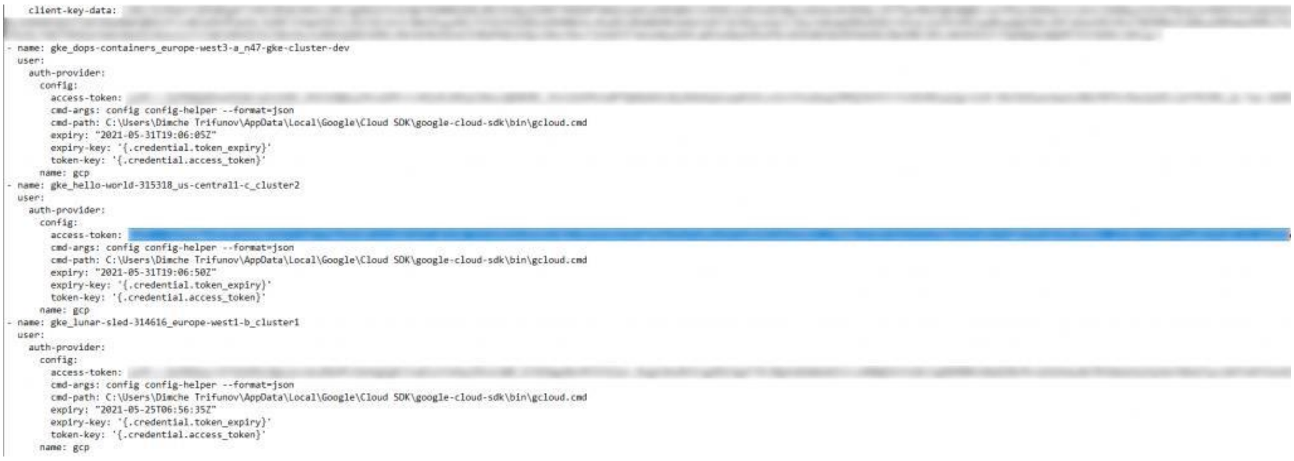
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

○ Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

Enter token *

Now, you need the token from the config file that we spoke about a moment ago. Open the config file with Notepad (on Windows), find your access token, and copy from there and paste it in the *Enter token** field. Be careful when you are copying a token from the config file as there might be several tokens. You must choose yours (image below).



```
    client-key-data: [redacted]
- name: gke_dops-containers_europe-west3-a_n47-gke-cluster-dev
  user:
    auth-provider:
      config:
        access-token: [redacted]
        cmd-args: config config-helper --format=json
        cmd-path: C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd
        expiry: "2021-05-31T19:06:05Z"
        expiry-key: '{.credential.token_expiry}'
        token-key: '{.credential.access_token}'
      name: gcp
- name: gke_hello-world-315318_us-central1-c_cluster2
  user:
    auth-provider:
      config:
        access-token: [redacted]
        cmd-args: config config-helper --format=json
        cmd-path: C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd
        expiry: "2021-05-31T19:06:50Z"
        expiry-key: '{.credential.token_expiry}'
        token-key: '{.credential.access_token}'
      name: gcp
- name: gke_lunar-sled-314616_europe-west1-b_cluster1
  user:
    auth-provider:
      config:
        access-token: [redacted]
        cmd-args: config config-helper --format=json
        cmd-path: C:\Users\Dimche Trifunov\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\gcloud.cmd
        expiry: "2021-05-25T06:56:35Z"
        expiry-key: '{.credential.token_expiry}'
        token-key: '{.credential.access_token}'
      name: gcp
```

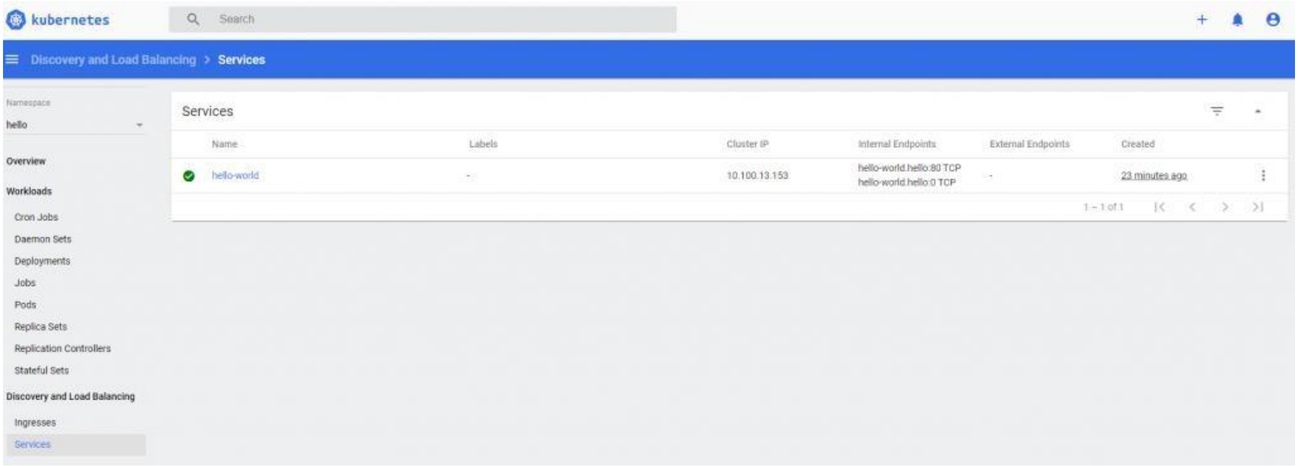
Finally, the stage is prepared to deploy microservice.

Step 12: Deploy microservice

Build the docker image from Dockerfile with the command: *docker build -t docker2222/dimac:latest*. docker2222/dimac is my public docker repository. Push the image on docker hub with the command: *docker image push docker2222/dimac:latest*. Execute *kubectl apply -f k8s.yaml* where k8s.yaml is the file below:

```
---

apiVersion: v1
kind: Namespace
metadata:
  name: hello

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
  namespace: hello
  annotations:
    buildNumber: "1.0"
spec:
  selector:
    matchLabels:
      app: hello-world
  replicas: 1
  template:
    metadata:
      labels:
        app: hello-world
      annotations:
        buildNumber: "1.0"
    spec:
      containers:
        - name: hello-world
          image: docker2222/dimac:latest
          readinessProbe:
            httpGet:
              path: "/actuator/health/readiness"
              port: 8080
            initialDelaySeconds: 5
          ports:
            - containerPort: 8080
          env:
            - name: APPLICATION_VERSION
              value: "1.0"
---


apiVersion: v1
kind: Service
metadata:
  name: hello-world
namespace: hello
spec:
  selector:
    app: hello-world
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
---
```

Open the Kubernetes dashboard. Now, the service can be seen.



Services

	Name	Labels	Cluster IP	Internal Endpoints	External Endpoints	Created
✓	hello-world	-	10.100.13.153	hello-world.hello:80 TCP hello-world.hello:0 TCP	-	23 minutes ago

1 – 1 of 1

RESULT:

Thus the Microservices using Kubernetes in Google Cloud has been deployed successfully.