1. Design a TCP socket-based communication system.

**Server:**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h> // read(), write(), close()

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int connfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(connfd, buff, sizeof(buff));

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

bzero(buff, MAX);

n = 0;

while ((buff[n++] = getchar()) != '\n');

write(connfd, buff, sizeof(buff));

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}}}

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server accept failed...\n");

exit(0);

}

else

printf("server accept the client...\n");

func(connfd);

close(sockfd);

}

**Client:**

#include <arpa/inet.h> // inet\_addr()

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <strings.h> // bzero()

#include <sys/socket.h>

#include <unistd.h> // read(), write(), close()

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));

bzero(buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

break;

}}}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr))

!= 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

func(sockfd);

close(sockfd);

}

1. Design a UDP socket-based communication system

Server:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from server";

struct sockaddr\_in servaddr, cliaddr;

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(PORT);

if ( bind(sockfd, (const struct sockaddr \*)&servaddr,

sizeof(servaddr)) < 0 )

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

int len, n;

len = sizeof(cliaddr);

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, ( struct sockaddr \*) &cliaddr,

&len);

buffer[n] = '\0';

printf("Client : %s\n", buffer);

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &cliaddr,

len);

printf("Hello message sent.\n");

return 0;

}

Client:

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from client";

struct sockaddr\_in servaddr;

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(PORT);

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

int n, len;

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &servaddr,

sizeof(servaddr));

printf("Hello message sent.\n");

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, (struct sockaddr \*) &servaddr,

&len);

printf("Server : %s\n", buffer);

close(sockfd);

return 0;

}

1. Design a Web service using Simple Object Access Protocol (SOAP).

**Source Code:**

**// MD5WebService.java**

package vce.webservices.server;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import javax.jws.WebMethod;

import javax.jws.WebService;

@WebService

public class MD5WebService {

@WebMethod

public String hashString(String input) {

try {

MessageDigest msgDigest = MessageDigest.getInstance("MD5");

byte[] inputBytes = input.getBytes();

byte[] hashedBytes = msgDigest.digest(inputBytes);

StringBuffer sb = new StringBuffer();

for (int i = 0; i < hashedBytes.length; i++) {

sb.append(Integer.toString((hashedBytes[i] & 0xff) + 0x100, 16)

.substring(1));

}

return sb.toString();

} catch (NoSuchAlgorithmException ex) {

ex.printStackTrace();

return "";

}

}

}

**//WebServiceServer.java**

package vce.webservices.server;

import javax.xml.ws.Endpoint;

public class WebServiceServer {

public static void main(String[] args) {

String bindingURI = "http://localhost:9898/md5WebService";

MD5WebService webService = new MD5WebService();

Endpoint.publish(bindingURI, webService);

System.out.println("Server started at: " + bindingURI);

}

}

**//WebServiceClient.java**

package vce.webservices.client;

public class WebServiceClient {

public static void main(String[] args) {

MD5WebServiceService client = new MD5WebServiceService();

MD5WebService md5Webservice = client.getMD5WebServicePort();

String hash = md5Webservice.hashString("hyderabad");

System.out.println("MD5 hash string: " + hash);

}

}

1. Develop a Multi-chat application.

**Server:**

import java.io.\*;

import java.util.\*;

import java.net.\*;

public class Server

{

    static Vector<ClientHandler> ar = new Vector<>();

    static int i = 0;

    public static void main(String[] args) throws IOException

    {

        ServerSocket ss = new ServerSocket(1234);

        Socket s;

        while (true)

        {

            s = ss.accept();

            System.out.println("New client request received : " + s);

            DataInputStream dis = new DataInputStream(s.getInputStream());

            DataOutputStream dos = new DataOutputStream(s.getOutputStream());

            System.out.println("Creating a new handler for this client...");

            ClientHandler mtch = new ClientHandler(s,"client " + i, dis, dos);

            Thread t = new Thread(mtch);

            System.out.println("Adding this client to active client list");

            ar.add(mtch);

            t.start();

            i++;

}}}

class ClientHandler implements Runnable

{

    Scanner scn = new Scanner(System.in);

    private String name;

    final DataInputStream dis;

    final DataOutputStream dos;

    Socket s;

    boolean isloggedin;

    public ClientHandler(Socket s, String name,

                            DataInputStream dis, DataOutputStream dos) {

        this.dis = dis;

        this.dos = dos;

        this.name = name;

        this.s = s;

        this.isloggedin=true;

    }

    public void run() {

        String received;

        while (true)

        {

            try

            {

                received = dis.readUTF();

                System.out.println(received);

                if(received.equals("logout")){

                    this.isloggedin=false;

                    this.s.close();

                    break;

                }

                StringTokenizer st = new StringTokenizer(received, "#");

                String MsgToSend = st.nextToken();

                String recipient = st.nextToken();

                for (ClientHandler mc : Server.ar)

                {

                    if (mc.name.equals(recipient) && mc.isloggedin==true)

                    {

                        mc.dos.writeUTF(this.name+" : "+MsgToSend);

                        break;

}}} catch (IOException e) {

                e.printStackTrace();

}}

        try

        {

            this.dis.close();

            this.dos.close();

        }catch(IOException e){

            e.printStackTrace();

        }

    }

}

**Client:**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class Client

{

    final static int ServerPort = 1234;

    public static void main(String args[]) throws UnknownHostException, IOException

    {

        Scanner scn = new Scanner(System.in);

        InetAddress ip = InetAddress.getByName("localhost");

        Socket s = new Socket(ip, ServerPort);

        DataInputStream dis = new DataInputStream(s.getInputStream());

        DataOutputStream dos = new DataOutputStream(s.getOutputStream());

        Thread sendMessage = new Thread(new Runnable()

        {

            @Override

            public void run() {

                while (true) {

                    String msg = scn.nextLine();

                    try {

                        // write on the output stream

                        dos.writeUTF(msg);

                    } catch (IOException e) {

                        e.printStackTrace();

                   }}}});

        Thread readMessage = new Thread(new Runnable()

        {

            @Override

            public void run() {

                while (true) {

                    try {

                        String msg = dis.readUTF();

                        System.out.println(msg);

                    } catch (IOException e) {

                        e.printStackTrace();

                    }}}});

        sendMessage.start();

        readMessage.start();    }

}

1. Implement a 2PC protocol for distributed transaction management.

**Server.java :**

import java.io.\*;

import java.net.\*;

class Clients

{

static int n;

static String[] status= new String[2];

Clients(int num){

n=num;

for (int j=0;j<n ;j++ )

{

status[j] = new String("NotPrepared");}}}

class Coordinator implements Runnable{

public static int i=-1;

int flag=1;

Socket s; Thread t;

MulticastSocket ms =null;

InetAddress group ;

Coordinator(Socket c){

try{

ms = new MulticastSocket(8899);

group= InetAddress.getByName("228.5.6.7");

ms.joinGroup(group);

}

catch (Exception e){

e.printStackTrace();

}

s=c;

t = new Thread(this);

t.start();

i++;

}

public void run(){

int index = i;

String clientSattus;

try{

DataInputStream input=new

DataInputStream(s.getInputStream());

DataOutputStream output=new

DataOutputStream(s.getOutputStream());

while (true){

clientSattus = input.readUTF();

System.out.println("Client "+index+" "+clientSattus);

Clients.status[index] = new String (clientSattus);

for (int k=0;k<Clients.n; k++){

System.out.println(Clients.status[k]);

if (Clients.status[k].equalsIgnoreCase("prepared"))

continue;

else

flag=0;

if (flag==1){

byte[] msg = new String("commit").getBytes();

DatagramPacket msgpack = new

DatagramPacket(msg,msg.length, group, 8899);

ms.send(msgpack);

System.out.println("BroadCasted msg "+new String(msg));

}

flag=1;

}}

catch (Exception e){

e.printStackTrace();

}}}

class Server {

public static ServerSocket ss; Server(){}

public static void main(String args[]) throws Exception{

ss = new ServerSocket(8088);

int num;

num = Integer.parseInt(args[0]);

new Clients(num);

while (true){

System.out.println("Server waiting: ");

Socket s = ss.accept();

new Coordinator(s);

}}}

**MulticastPeer.java:**

Import java.net.\*;

Import java.io.\*;

public class MulticastPeer{

public static void main(String args[]){

MulticastSocket s =null;

try {

InetAddress group = InetAddress.getByName(args[1]);

s = new MulticastSocket(6789);

s.joinGroup(group);

byte [] m = args[0].getBytes();

DatagramPacket messageOut = new DatagramPacket(m, m.length, group, 6789);

s.send(messageOut);

byte[] buffer = new byte[1024];

for(int i=0; i< 3;i++) {

DatagramPacket messageIn = new DatagramPacket(buffer, buffer.length);

s.receive(messageIn);

System.out.println("Received:" + new String(messageIn.getData()));

}

s.leaveGroup(group);

}catch (SocketException e){System.out.println("Socket: " + e.getMessage());

}catch (IOException e){System.out.println("IO: " + e.getMessage());

}finally {if(s != null) s.close();}}}

DBConnector.java :

import java.sql.\*;

class DBConnector

{

public static Connection getDBConnection(String dsn) throws Exception{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

return DriverManager.getConnection("jdbc:odbc:"+dsn);

}

}

**Client.java :**

import java.io.\*;

import java.net.\*;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

import java.util.\*;

class Client extends JFrame implements ActionListener{

JButton b1,b2,b4,b5;

JPanel p1,p2;

JTextField t1;

JLabel l1;

ServerSocket ss;

Socket s;

DataOutputStream output;

DataInputStream input;

Connection con;

Statement stmt;

String serverMessage="Prepared";

static int port;

Client(){

b1=new JButton("Prepared");

b2=new JButton("NotPrepared");

b4=new JButton("Execute");

b5=new JButton("Exit");

t1=new JTextField("",35);

l1=new JLabel("SQL");

p1=new JPanel();

p2=new JPanel();

p1.setLayout(new FlowLayout());

p1.add(l1);

p1.add(t1);

p2.add(b1);

p2.add(b2);

p2.add(b4);

p2.add(b5);

add(p1);

add(p2,"South");

setSize(600,300);

setTitle("DNS Client");

setVisible(true);

b1.addActionListener(this);

b2.addActionListener(this);

b4.addActionListener(this);

b5.addActionListener(this);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

MulticastSocket ms =null;

InetAddress group ;

try{

s = new Socket("localhost",8088);

System.out.println("Client Connected");

output=new DataOutputStream(s.getOutputStream());

input=new DataInputStream(s.getInputStream());

con = DBConnector.getDBConnection("mydb");

stmt = con.createStatement();

con.setAutoCommit(false);

ms = new MulticastSocket(8899);

group=InetAddress.getByName("228.5.6.7");

ms.joinGroup(group);

byte[] buffer = new byte[1024];

while (true){

DatagramPacket serMsg= new DatagramPacket(buffer, buffer.length);

ms.receive(serMsg);

String commitMsg = new String (serMsg.getData()).trim();

if (commitMsg.equals("commit"))

{

System.out.println("Received "+commitMsg); con.commit();

System.out.println("Transactions Committed");

}}}

catch (Exception e){

e.printStackTrace();

}}

public void actionPerformed(ActionEvent ae){

try{

String str=ae.getActionCommand();

if(str.equals("Execute")){

String query = t1.getText(); System.out.println(stmt.executeUpdate(query));

t1.setText("Query Executed(NotPrepared)"); output.writeUTF("NotPrepared");

}

if(str.equals("Prepared")){

output.writeUTF("Prepared");

}

Query.txt :

update account set bal = 500 where accno = 1

1. Host a static website on Amazon AWS.

EXECUTE MODULE 3 GUIDED LAB

1. Deploy a Node.js Web Application using AWS services / EC2

EXECUTE MODULE 4 GUIDED LAB

1. Implement a distributed application on Hadoop framework to count word frequency with Map Reduce

**Source Code:**

package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static void main(String [] args) throws Exception

{

Configuration c=new Configuration();

String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path input=new Path(files[0]);

Path output=new Path(files[1]);

Job j=new Job(c,"wordcount");

j.setJarByClass(WordCount.class);

j.setMapperClass(MapForWordCount.class);

j.setReducerClass(ReduceForWordCount.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(j, input);

FileOutputFormat.setOutputPath(j, output);

System.exit(j.waitForCompletion(true)?0:1);

}

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String line = value.toString();

String[] words=line.split(",");

for(String word: words )

{

Text outputKey = new Text(word.toUpperCase().trim());

IntWritable outputValue = new IntWritable(1);

con.write(outputKey, outputValue);

}

}

}

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

con.write(word, new IntWritable(sum));

}

}

}

The above program consists of three classes:

* Driver class (Public, void, static, or main; this is the entry point).
* The Map class which **extends** the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT>  and implements the Map function.
* The Reduce class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.

1. Make  a jar file

Right Click on Project> Export> Select export destination as **Jar File** > next> Finish.

1. Take a text file and move it into HDFS format:

move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost~]$hadoop fs -putwordcountFilewordCountFile

Run the jar file:

[training@localhost~]$hadoop jar MRProgramsDemo.jar PackageDemo.WordCountwordCountFile MRDir1

**Input / Output:**

[training@localhost~]$hadoop fs -ls MRDir1

Found 3 items

-rw-r--r-- 1 training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_SUCCESS

drwxr-xr-x - training supergroup 0 2016-02-23 03:36 /user/training/MRDir1/\_logs

-rw-r--r-- 1 training supergroup 20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000

[training@localhost~]$hadoop fs -cat MRDir1/part-r-00000

BUS 7

CAR 4

TRAIN 6

1. Installation and deploying a PhP application on a Docker Container

**Description:**

Create a Machine Image of Ubuntu Bionic 18.04LTS or Xenial 16.04.

To install **Docker CE**, first, you need to remove older versions of **Docker** were called **docker**, **docker.io**, or **docker-engine** from the system using the following command.

$ sudo apt-get remove docker docker-engine docker.io containerd runc

Next, you need to set up the Docker repository to install and update Docker from the repository using following commands.

1. Update the apt package index

$ sudo apt-get update

1. Install packages to allow apt to use a repository over HTTPS

$ sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

1. Add Docker’s official GPG key

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

1. Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint

sudo apt-key fingerprint 0EBFCD88

pub rsa4096 2017-02-22 [SCEA]

9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid [ unknown] Docker Release (CE deb) <docker@docker.com>

sub rsa4096 2017-02-22 [S]

1. Use the following command to set up the stable repository

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

The lsb\_release -cs sub-command below returns the name of your Ubuntu distribution, such as xenial. Sometimes, in a distribution like Linux Mint, you might need to change $(lsb\_release -cs) to your parent Ubuntu distribution. For example, if you are using Linux Mint Tessa, you could use bionic. Docker does not offer any guarantees on untested and unsupported Ubuntu distributions.

1. Update the apt package index and install the latest version of **Docker CE** using following commands.

$ sudo apt-get update

1. Install the latest version of Docker Engine - Community and containerd, or go to the next step 8 to install a specific version

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

1. To install a specific version of Docker Engine - Community, list the available versions in the repo, then select and install: List the versions available in your repo:

$ apt-cache madison docker-ce

1. Install a specific version using the version string from the second column, for example, 5:18.09.1~3-0~ubuntu-xenial

$ sudo apt-get install docker-ce=<VERSION\_STRING> docker-ce-cli=<VERSION\_STRING> containerd.io

1. After successfully installing the **Docker CE** package, the service should be auto-started and auto-enabled to start at system boot, you can check its status using the following command.

$ sudo systemctl status docker

1. Press CTRL C to exit
2. Verify that Docker Engine - Community is installed correctly by running the hello-world image

$ sudo docker run hello-world

1. This command downloads a test image and runs it in a container. When the container runs, it prints the below informational message

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

1b930d010525: Pull complete

Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.

2. The Docker daemon pulled the "hello-world" image from the Docker Hub.

(amd64)

3. The Docker daemon created a new container from that image which runs the

executable that produces the output you are currently reading.

4. The Docker daemon streamed that output to the Docker client, which sent it

to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

For more examples and ideas, visit:

https://docs.docker.com/get-started/

**Dockerizing a Node.js web application**

1. Create a new folder namely nodejsapp
2. Make a package.json file as follows

{

"name": "docker\_web\_app",

"version": "1.0.0",

"description": "Node.js on Docker",

"author": "Sashi’s First Nodejs Application on Container <sashi.mamidanna@gmail.com>",

"main": "server.js",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"express": "^4.16.1"

}

}

1. Then create a file server.js to create a program that runs on the node. The idea is to enable the server.js file to run on the container at port no 8081

'use strict';

const express = require('express');

// Constants

const PORT = 8081;

const HOST = '0.0.0.0';

// App

const app = express();

app.get('/', (req, res) => {

res.send('Hello world\n');

});

app.listen(PORT, HOST);

console.log(`Running on http://${HOST}:${PORT}`);

1. Create a dockerfile now namely dockerfile in the same directory

$sudo nano dockerfile

1. Copy the source code into the dockerfile

FROM node:10

# Create app directory

WORKDIR /app

COPY . /app

RUN npm install

COPY . .

EXPOSE 8082

CMD [ "node", "server.js" ]

1. Now build the docker image with the node application on it

$sudo docker build -t nodejsapp .

1. Run the application by executing run command on docker

$sudo docker run -p 8082:8081 nodejsapp

1. The container engine will run the command node server.js that was initialized through the dockerfile. Now the server.js is listening to incoming requests on <http://localhost:8081> on the host operating system. But the application is running on port number 8082 on the docker engine.
2. Open a new ssh connection on the same VM and run the command to send an outgoing request to the application running on docker

$sudo curl <http://localhost:8082>

Hello World

1. This response is a result of the application running on node, devoted on the docker container, that's running on Docker engine available on the Ubuntu OS.
2. Run the bow command to check if the docker image is present in the list of images on Docker C

$sudo docker ps

1. To stop the docker container image

$sudo docker stop <docker image ID>

1. To remove the docker image
2. $sudo docker rmi <docker image ID>