**Lesson 07 Demo 02**

**Configuring Docker Networking for a Microservices Environment**

**Objective:** To configure Docker networking for a microservices environment for seamless communication and scalability among microservices within Docker containers  
  
**Tools required:** Ubuntu

**Prerequisites:** None

Steps to be followed:

1. Set up a simple HTTP server with Docker
2. Set up a client script with Docker
3. Configure Docker networking for microservices

**Step 1: Set up a simple HTTP server with Docker**

1. Run the following command to switch to the root user: **sudo su**
2. Create a directory using the following command:  
   **mkdir server**
3. Navigate inside the directory using the following command:  
   **cd server/**
4. Create a Python fileusing the following command:  
   **vi server.py**
5. Enter the following script inside the **server.py** file: **import http.server**

**import socketserver**

**handler = http.server.SimpleHTTPRequestHandler**

**with socketserver.TCPServer(("", 8090), handler) as httpd:**

**httpd.serve\_forever()**A screenshot of a computer

Description automatically generated

This script creates a simple HTTP server that listens on port 8090 and serves requests indefinitely using the built-in **http.server** module.

1. Create a file using the following command:  
   **vi index.html**  
     
   
2. Enter the data inside the **index.html** file as shown in the screenshot below: **We are learning Microservices in docker**A screenshot of a computer

   Description automatically generated
3. Create a Dockerfile using the following command:  
   **vi Dockerfile**
4. Enter the below script inside the **Dockerfile** file: **FROM python:latest**

**ADD server.py /server/**

**ADD index.html /server/**

**WORKDIR /server/**A screenshot of a computer

Description automatically generated

This **Dockerfile** sets up a container using the latest Python image, adds **server.py** and **index.html** to the **/server/** directory, and sets **/server/** as the working directory.

1. Exit from the directory using the following command: **cd ..**

**Step 2: Set up a client script with Docker**

1. Create a **client** directory using the following command**:  
   mkdir client**
2. Navigate inside the directory using the following command:  
   **cd client**  
     
   
3. Create a Python fileusing the following command: **vi client.py**  
     
   
4. Enter the below script inside the **client.py** file: **import urllib.request**

**fp = urllib.request.urlopen("http://localhost:8090/")**

**encodedContent = fp.read()**

**decodedContent = encodedContent.decode("utf8")**

**print(decodedContent)**

**fp.close()**

A screenshot of a computer

Description automatically generated

This script fetches the content from a local server at **http://localhost:8090/**, reads and decodes it, prints the decoded content, and then closes the connection.

1. Create a Dockerfileusing the following command: **vi Dockerfile**
2. Enter the below script inside the **Dockerfile** file: **FROM python:latest**

**ADD client.py /client/**

**WORKDIR /client/**  
A screenshot of a computer

Description automatically generated

This **Dockerfile** sets up a container using the latest Python image, adds **client.py** to the **/client/** directory, and sets **/client/** as the working directory.

1. Exit from the directory using the following command: **cd ..**  
     
   

**Step 3: Configure Docker networking for microservices**

1. Create a Docker compose fileusing the following command: **vi docker-compose.yml**  
     
   
2. Enter the below script inside the **docker-compose.yml** file:  
   **version: "3"**

**services:**

**server:**

**build: server/**

**command: python ./server.py**

**ports:**

**- 8090:8090**

**client:**

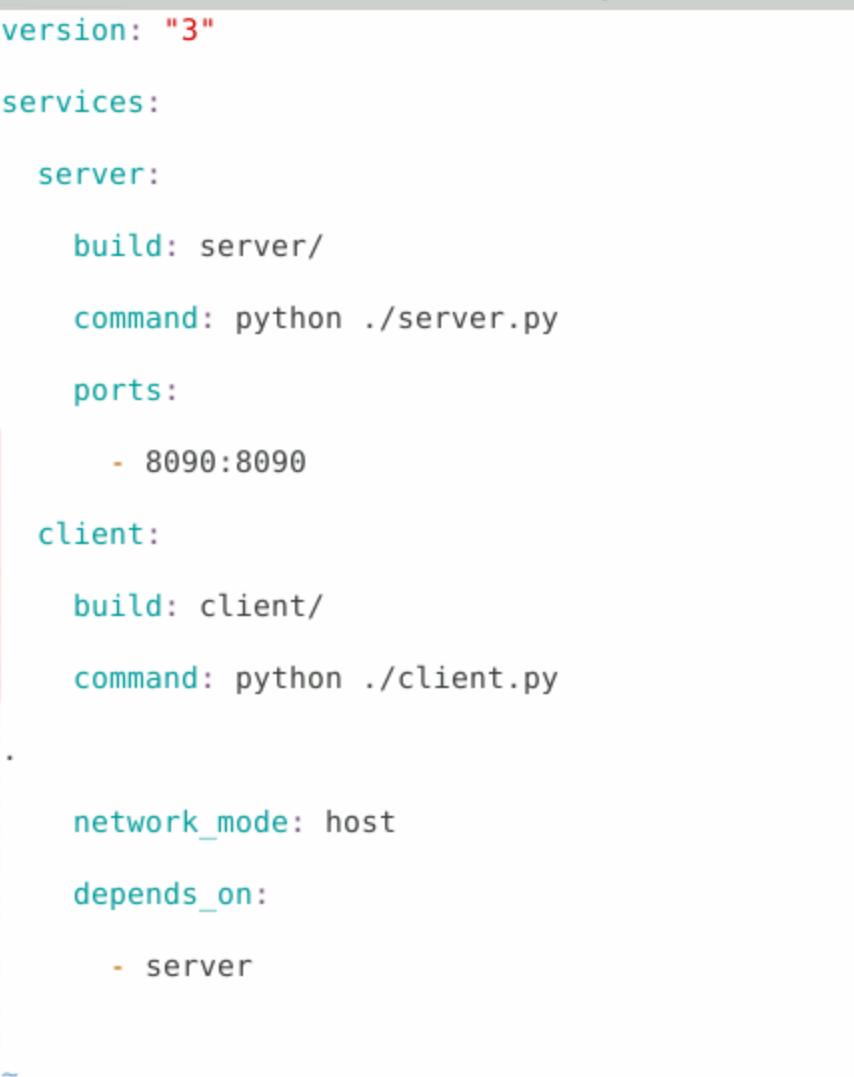
**build: client/**

**command: python ./client.py**

**network\_mode: host**

**depends\_on:**

**- server**



This Docker Compose file version 3 defines services for building and running a server from **server/** on port 8090 and a client from **client/**, where the client depends on the server and shares the host network.

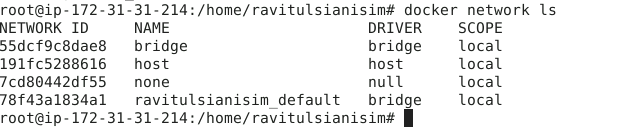
1. Run the following command to install Docker Compose in the system:  
   **install Docker-compose**  
     
   A screenshot of a computer error

   Description automatically generated
2. Build the Docker Compose using the following command: **docker-compose build**  
     
   A close up of a text

   Description automatically generated
3. List all the Docker images that are currently stored on a system:  
   **docker images**  
     
   A black text on a white background

   Description automatically generated
4. Run the following command to install the Docker Compose:  
   **apt install docker-compose -y**  
   A screenshot of a computer

   Description automatically generated
5. Start the Docker containers defined in a Docker Compose file:  
   **docker-compose up -d  
     
   A screenshot of a computer code

   Description automatically generated**
6. Check the network in which the microservice is running using the following command: **docker network ls**  
     
   ****
7. Inspect the network using the following command:  
   **docker inspect bridge**  
     
   **A screenshot of a computer

   Description automatically generated**  
     
   By following these steps, you have successfully configured Docker networking for a microservices environment for seamless communication and scalability among microservices within Docker containers.