Assignment 2:

**Step 1: Check for KVM Support**

1. Verify if your system supports hardware virtualization:

bash

Copy code

egrep -c '(vmx|svm)' /proc/cpuinfo

* + If the output is greater than 0, your CPU supports virtualization.
  + Ensure virtualization is enabled in the BIOS/UEFI settings.

1. Install required tools to confirm virtualization support:

bash

Copy code

sudo apt update

sudo apt install cpu-checker -y

kvm-ok

**Step 2: Install KVM and Virt-Manager**

**For Ubuntu/Debian-based systems:**

1. Install KVM and dependencies:

bash

Copy code

sudo apt install -y qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils

1. Add your user to the libvirt and kvm groups:

bash

Copy code

sudo usermod -aG libvirt,kvm $USER

1. Install virt-manager:

bash

Copy code

sudo apt install -y virt-manager

1. Start and enable the libvirtd service:

bash

Copy code

sudo systemctl enable --now libvirtd

**For Fedora/RHEL-based systems:**

1. Install KVM and dependencies:

bash

Copy code

sudo dnf install -y qemu-kvm libvirt virt-install bridge-utils

1. Add your user to the libvirt group:

bash

Copy code

sudo usermod -aG libvirt $USER

1. Install virt-manager:

bash

Copy code

sudo dnf install -y virt-manager

1. Start and enable the libvirtd service:

bash

Copy code

sudo systemctl enable --now libvirtd

**Step 3: Verify Installation**

1. Check if libvirtd is running:

bash

Copy code

sudo systemctl status libvirtd

1. Verify KVM is properly installed:

bash

Copy code

virsh list --all

**Step 4: Create and Run a Virtualized OS**

1. Launch virt-manager:

bash

Copy code

virt-manager

1. Create a new virtual machine:
   * Open virt-manager.
   * Click **File → New Virtual Machine**.
   * Select how you want to install the OS (e.g., ISO file, PXE, etc.).
   * Specify the resources (CPU, RAM, and storage).
   * Complete the configuration and start the VM.
2. Manage the VM through the virt-manager GUI.

**Troubleshooting**

* If you encounter permission issues:

bash

Copy code

sudo chmod 666 /dev/kvm

* Ensure virt-manager dependencies are installed properly:

bash

Copy code

sudo apt install -f

Once the setup is complete, you can run and manage virtual machines easily using virt-manager

Assignment 3-

**Task 1: Install Wine on Linux and Run a Windows Program**

Wine (Wine Is Not an Emulator) allows you to run Windows applications on Linux.

**Step 1: Install Wine**

1. **Update your system**:

bash

Copy code

sudo apt update

sudo apt upgrade -y

1. **Add the Wine repository and install Wine (Ubuntu/Debian)**:

bash

Copy code

sudo dpkg --add-architecture i386

sudo mkdir -pm755 /etc/apt/keyrings

sudo wget -O /etc/apt/keyrings/winehq-archive.key https://dl.winehq.org/wine-builds/winehq.key

sudo wget -NP /etc/apt/sources.list.d/ https://dl.winehq.org/wine-builds/ubuntu/dists/$(lsb\_release -cs)/winehq-$(lsb\_release -cs).sources

sudo apt update

sudo apt install --install-recommends winehq-stable -y

Replace $(lsb\_release -cs) with your Ubuntu version, e.g., focal for 20.04.

1. **Verify Wine installation**:

bash

Copy code

wine --version

**Step 2: Run a Windows Program**

1. Download a .exe file (e.g., Notepad++, Paint.NET).
2. Run the .exe file using Wine:

bash

Copy code

wine <program-name>.exe

Example:

bash

Copy code

wine notepad++.exe

1. Configure Wine if prompted (first run may require setup).

**Task 2: Install DOSBox and Run a Legacy DOS Application**

DOSBox is a DOS emulator that allows running legacy DOS apps and games.

**For Linux:**

1. **Install DOSBox**:

bash

Copy code

sudo apt install dosbox -y

1. **Run DOSBox**:

bash

Copy code

dosbox

1. **Run a DOS App or Game**:
   * Copy the DOS app/game to a directory, e.g., ~/dosapps/.
   * Mount the directory in DOSBox:

bash

Copy code

mount c ~/dosapps/

c:

* + Run the executable:

bash

Copy code

game.exe

**For Windows:**

1. **Download DOSBox**:
   * Download it from [DOSBox's official website](https://www.dosbox.com/).
   * Install and launch DOSBox.
2. **Run a DOS App or Game**:
   * Copy the DOS app/game to C:\dosapps.
   * Mount the directory:

bash

Copy code

mount c c:\dosapps

c:

* + Run the executable:

bash

Copy code

game.exe

**Run Mario in DOSBox**

1. Download a Mario DOS version or any other DOS game.
2. Place it in the folder you mounted in DOSBox.
3. Run the executable as described above:

bash

Copy code

mario.exe

By completing these tasks, you'll have explored application-level virtualization and emulation effectively!

Assignment 4-

Here is a step-by-step guide for **Assignment No. 4: Installation & Configuration of Bare Metal Hypervisor: Citrix Hypervisor**:

**Step 1: Download Citrix XenServer (Citrix Hypervisor) ISO**

1. Visit the official Citrix website or XenServer open-source community to download the ISO:
   * Citrix Hypervisor Download
   * Sign up if required to access the downloads.
2. Choose the open-source or free edition of Citrix Hypervisor.
3. Save the ISO file to your local system.

**Step 2: Create a Bootable USB**

Use a tool like **Rufus** (Windows), **Etcher** (cross-platform), or dd (Linux) to create a bootable USB.

**Using Rufus (Windows):**

1. Insert a USB drive (at least 8 GB) and launch Rufus.
2. Select the downloaded ISO file.
3. Choose "GPT" for UEFI or "MBR" for BIOS, depending on your system.
4. Click **Start** to create the bootable USB.

**Using Etcher (Linux/Windows/Mac):**

1. Download and install Etcher.
2. Select the ISO and the USB drive.
3. Click **Flash** to create the bootable USB.

**Using dd (Linux):**

1. Insert the USB and identify its name (e.g., /dev/sdb):

bash

Copy code

lsblk

1. Run the following command:

bash

Copy code

sudo dd if=<path-to-iso> of=/dev/sdX bs=4M status=progress

Replace <path-to-iso> with the full path to the ISO file and /dev/sdX with your USB drive.

**Step 3: Install Citrix Bare Metal Hypervisor**

1. **Boot from USB**:
   * Insert the bootable USB into the server machine.
   * Restart and boot into the USB (change boot order in BIOS/UEFI if necessary).
2. **Install Citrix Hypervisor**:
   * Follow the on-screen installation wizard.
   * Select the target disk for installation.
   * Configure basic settings such as keyboard layout, time zone, and root password.
   * Complete the installation and restart the system.

**Step 4: Configure the XenServer**

1. **Initial Configuration**:
   * After reboot, log in as root using the password set during installation.
   * Configure network settings (static IP, gateway, DNS):

bash

Copy code

xe pif-reconfigure-ip uuid=<uuid-of-interface> mode=static IP=<IP> netmask=<netmask> gateway=<gateway>

Replace <uuid-of-interface> with the UUID of the network interface.

1. **Create Users**:
   * Add users:

bash

Copy code

xe subject-add name-label=<username>

* + Assign roles:

bash

Copy code

xe role-add-actors role-name=<role> subject-identifier=<username>

**Step 5: Check XenServer Accessibility via LAN**

1. **Install XenCenter Client on Another Machine**:
   * Download Citrix XenCenter from the Citrix website.
   * Install it on a Windows/Linux machine in the same LAN.
2. **Connect to XenServer**:
   * Open XenCenter.
   * Add a new server connection:
     + Enter the XenServer's static IP address.
     + Provide the root credentials.
3. **Verify Connectivity**:
   * Check if the XenServer's dashboard is accessible.
   * Create a test VM to ensure everything is functioning correctly.

**Notes:**

* Ensure the server is connected to the same network as the client for XenCenter access.
* Use the xe command-line tool for advanced configurations if needed.
* If the LAN setup requires additional configuration, ensure firewalls or switches are correctly set up to allow communication between the XenServer and XenCenter client.

By completing these steps, you'll successfully install, configure, and test the Citrix XenServer bare-metal hypervisor.

Assignment 5-

**Step 1: Use the Citrix XenServer**

* Ensure the XenServer from **Assignment 4** is set up, running, and accessible via its static IP in the LAN.
* Log into the XenServer using SSH or a connected terminal for configuration.

**Step 2: Download an Operating System ISO File**

1. Download any OS ISO file (e.g., Ubuntu, CentOS, or Windows Server) on your local machine.
   * Visit official OS websites to get ISO files:
     + Ubuntu: <https://ubuntu.com/download>
     + CentOS: https://www.centos.org/download/
     + Windows Server: <https://www.microsoft.com/en-us/evalcenter/>
2. Save the ISO file on a system accessible in your LAN.

**Step 3: Create a Local ISO Repository on Citrix XenServer**

1. **Access XenServer**:
   * SSH into the XenServer or log in directly.
2. **Create a Repository Directory**:

bash

Copy code

mkdir -p /var/iso\_repository

1. **Copy the ISO File to the Repository**:
   * Transfer the ISO file from your local system to XenServer using SCP:

bash

Copy code

scp <path-to-iso> root@<xenserver-ip>:/var/iso\_repository/

1. **Create the ISO Storage Repository in XenServer**:
   * Run the following command to create an ISO repository:

bash

Copy code

xe sr-create name-label="Local ISO Repository" type=iso device-config:location=/var/iso\_repository content-type=iso

1. **Verify the Repository**:
   * Check if the repository is available:

bash

Copy code

xe sr-list

**Step 4: Access ISO Repository via Citrix XenCenter**

1. Open **XenCenter** on a client machine in the same LAN.
2. Add the XenServer instance (if not already added):
   * Enter the server's static IP and root credentials.
3. Navigate to **Storage** and verify that the "Local ISO Repository" appears under **Storage Repositories**.

**Step 5: Create a Virtual Machine**

1. **Launch VM Creation Wizard**:
   * In XenCenter, click **New VM**.
2. **Configure the VM**:
   * **Template**: Choose an appropriate template for the OS (e.g., Ubuntu, Windows).
   * **Name and Description**: Provide a name for the VM.
   * **ISO**: Select the ISO from the "Local ISO Repository."
   * **CPU, Memory, and Disk**: Allocate resources:
     + CPU: Assign virtual CPUs.
     + RAM: Specify the memory (e.g., 2 GB).
     + Disk: Allocate virtual disk space (e.g., 20 GB).
3. **Networking**:
   * Assign the VM to a virtual network on the XenServer.
4. **Complete the Wizard**:
   * Review the configuration and click **Finish** to create the VM.

**Step 6: Access the Virtual Machine and Observe Orchestration**

1. **Start the Virtual Machine**:
   * In XenCenter, select the VM and click **Start**.
2. **Open VM Console**:
   * Access the VM's console from the XenCenter client to observe the boot process.
3. **Observe Orchestration**:
   * Monitor resource allocation and usage from XenCenter (e.g., CPU, Memory, Disk I/O).
   * Use the VM to install and configure the OS, verifying that it operates as expected.

**Key Observations**

* **Resource Allocation**: XenCenter shows how CPU, memory, and storage are allocated to the VM.
* **Orchestration**: Observe the efficient resource sharing among multiple VMs (if created).
* **Networking**: Verify network connectivity for the VM within the LAN.

By following these steps, you'll create and manage a virtualized environment using Citrix XenServer, demonstrating full virtualization with an IaaS private cloud setup.

Assignment 6:

**Study of AWS EC2 - IaaS Public Cloud**

This guide walks through creating a virtual machine instance (EC2) using the free tier on AWS.

**Step 1: Create an Amazon AWS Account**

1. **Sign up for AWS**:
   * Go to [AWS Sign Up](https://aws.amazon.com/).
   * Provide your email, set a password, and create an account.
2. **Enter Billing Information**:
   * Add a credit/debit card (required for verification; no charges for free-tier usage).
3. **Verify Identity**:
   * Verify your phone number via SMS or call.
4. **Select a Plan**:
   * Choose the **Free Tier** plan to avoid charges.

**Step 2: Access the AWS Management Console**

1. Log in to the [AWS Management Console](https://aws.amazon.com/console/).
2. Search for **EC2** in the services search bar.

**Step 3: Create an EC2 Instance**

1. **Launch a Virtual Machine (Instance)**:
   * In the EC2 dashboard, click **Launch Instances**.
2. **Configure the Instance**:
   * **Name and Tags**: Name your instance (e.g., "MyFreeTierInstance").
   * **AMI (Amazon Machine Image)**: Choose a free-tier eligible image (e.g., Amazon Linux 2023, Ubuntu 22.04).
   * **Instance Type**: Select t2.micro (1 vCPU, 1 GB RAM, free tier eligible).
3. **Key Pair (SSH)**:
   * Create a new key pair or select an existing one:
     + If creating a new key pair, download the .pem file and keep it secure.
4. **Network Settings**:
   * Leave defaults for now (auto-assign public IP, default VPC and subnet).
5. **Storage**:
   * Use the default (8 GB general-purpose SSD, free tier eligible).
6. **Review and Launch**:
   * Click **Launch Instance**.

**Step 4: Access the EC2 Instance Console**

1. **Connect to the Instance**:
   * Go to the **Instances** page in the EC2 dashboard.
   * Select your instance, then click **Connect**.
2. **Connect Using Browser Console**:
   * Choose **Session Manager** or **EC2 Instance Connect (browser-based SSH)**.
   * Click **Connect** to access the instance.

**Step 5: Execute Commands on the Instance**

1. **Verify the Connection**:
   * Run the following to check the instance's OS and kernel:

bash

Copy code

uname -a

1. **Update the System**:
   * For Amazon Linux:

bash

Copy code

sudo yum update -y

* + For Ubuntu:

bash

Copy code

sudo apt update && sudo apt upgrade -y

1. **Install a Sample Package**:
   * For Amazon Linux:

bash

Copy code

sudo yum install nano -y

* + For Ubuntu:

bash

Copy code

sudo apt install nano -y

1. **Create a Test File**:

bash

Copy code

echo "Hello from AWS EC2 Free Tier" > testfile.txt

cat testfile.txt

**Step 6: Stop or Terminate the Instance**

1. To avoid incurring costs, stop or terminate the instance when not in use:
   * Select the instance in the EC2 dashboard.
   * Click **Instance State → Stop** (or **Terminate** if no longer needed).

**Important Notes**

* AWS Free Tier allows 750 hours per month for t2.micro instances.
* Monitor usage to avoid exceeding the free-tier limits.
* Store your key pair securely for future SSH connections.

By following these steps, you’ll have successfully created, accessed, and managed an EC2 instance using AWS’s free tier.

Assignment 7:

**Hosting a Website on Public Cloud using AWS EC2 and Elastic IP**

This guide walks through hosting a website on your EC2 instance, assigning a static public IP using **Amazon Elastic IP (EIP)**, and accessing it globally.

**Step 1: Use the EC2 Instance from Assignment No. 6**

1. **Access the EC2 Instance**:
   * Log into the AWS Management Console and navigate to EC2.
   * Select the EC2 instance created in Assignment 6 (from the Free Tier) and ensure it's running.

**Step 2: Install Apache HTTP Server (httpd) on the EC2 Instance**

1. **Connect to the EC2 Instance**:
   * Use the browser-based SSH method (EC2 Instance Connect) or your preferred SSH tool to connect to the instance.
2. **Install Apache HTTP Server (httpd)**:
   * **For Amazon Linux 2023** or **RHEL/CentOS-based instances**:

bash

Copy code

sudo yum update -y

sudo yum install httpd -y

* + **For Ubuntu-based instances**:

bash

Copy code

sudo apt update

sudo apt install apache2 -y

1. **Start Apache HTTP Server**:
   * **For Amazon Linux 2023** or **RHEL/CentOS**:

bash

Copy code

sudo systemctl start httpd

sudo systemctl enable httpd

* + **For Ubuntu**:

bash

Copy code

sudo systemctl start apache2

sudo systemctl enable apache2

1. **Check Apache Status**:
   * Run the following command to ensure the Apache server is running:

bash

Copy code

sudo systemctl status httpd # for Amazon Linux

sudo systemctl status apache2 # for Ubuntu

**Step 3: Host Your Web Pages in the Document Root**

1. **Upload Web Pages to Apache Document Root**:
   * The default document root for Apache is /var/www/html/.
   * Use the scp (secure copy) command or any file transfer method to upload your HTML files to this directory.
     + For example:

bash

Copy code

scp index.html root@<EC2-IP>:/var/www/html/

1. **Verify Web Pages**:
   * Navigate to the document root to ensure your files are in place:

bash

Copy code

cd /var/www/html/

ls -l

1. **Check the Web Server**:
   * Open the EC2 public IP in a browser to verify your website is live (this can be accessed temporarily via the public IP assigned to your EC2 instance).
   * You should see the webpage you uploaded.

**Step 4: Assign Static Public IP using Amazon Elastic IP**

1. **Allocate Elastic IP**:
   * Go to the **Elastic IPs** section under the **Network & Security** menu in the EC2 dashboard.
   * Click on **Allocate Elastic IP** and choose **Amazon pool**.
   * Click **Allocate** to get the Elastic IP.
2. **Associate Elastic IP with EC2 Instance**:
   * After allocating the Elastic IP, select **Actions → Associate Elastic IP**.
   * Choose the instance you want to associate the IP with.
   * Click **Associate**.
3. **Verify Public IP**:
   * The Elastic IP will now be assigned to your EC2 instance. You can check it in the EC2 dashboard under the **Instances** section.
4. **Update Security Group to Allow HTTP Access**:
   * Ensure your EC2 instance's security group allows HTTP traffic (port 80).
     + In the **Security Groups** section, ensure there is a rule for inbound traffic:
       - Type: **HTTP**
       - Protocol: **TCP**
       - Port: **80**
       - Source: **0.0.0.0/0** (or specify a range if needed)

**Step 5: Access the Website Globally using the Elastic IP**

1. **Test Access**:
   * Open any browser and enter the **Elastic IP** assigned to your EC2 instance. You should be able to see the website you've hosted.
2. **Test from Different Networks**:
   * Try accessing the website from different networks (e.g., your mobile data network, different Wi-Fi networks) to ensure it's publicly accessible.

**Final Checklist:**

* **Apache HTTP Server**: Make sure it is running on your EC2 instance.
* **Elastic IP**: The static Elastic IP should be associated with the EC2 instance.
* **Security Group**: Ensure port 80 is open for HTTP traffic.
* **Webpage**: Your custom webpage should be accessible via the public Elastic IP.

By following these steps, you'll successfully host a website on your EC2 instance and make it globally accessible using **Amazon Elastic IP**.

Assignment 8:

**Installation of Docker Desktop and Basic Docker Commands**

This guide will walk you through installing Docker Desktop on your machine, running the "hello-world" container, using Docker commands via the CLI, and exploring Docker GUI utilities. Additionally, you'll pull and run the Ubuntu image and access the container's bash terminal.

**Step 1: Install Docker Desktop on Your Machine**

1. **Download Docker Desktop**:
   * Go to the Docker Download Page and select the version that matches your operating system (Windows or macOS).
2. **Install Docker**:
   * **Windows**:
     + Run the installer and follow the prompts. Once installed, Docker Desktop will start automatically.
     + You may need to enable **Hyper-V** and **Containers** features (if not already enabled). Docker will prompt you to do so if necessary.
   * **macOS**:
     + Run the .dmg file you downloaded and follow the instructions to install Docker.
3. **Start Docker**:
   * After installation, launch Docker Desktop. You will see the Docker icon in your system tray (Windows) or the menu bar (macOS) indicating Docker is running.
4. **Verify Installation**:
   * Open a terminal or command prompt and type the following to check if Docker is installed and running:

bash

Copy code

docker --version

docker info

**Step 2: Use Docker CLI and Run the Hello-world Container**

1. **Pull the "hello-world" Image**:
   * In the terminal (CLI), run the following command to pull the Docker "hello-world" image from Docker Hub:

bash

Copy code

docker pull hello-world

1. **Run the Hello-world Container**:
   * After pulling the image, run the following command to start the "hello-world" container:

bash

Copy code

docker run hello-world

* + This will pull the image (if not already downloaded) and run a container, printing a message indicating that Docker is installed and working properly.

1. **Stop the Hello-world Container**:
   * Once the "hello-world" message is displayed, the container will stop automatically. You can list all containers (including the stopped ones) using:

bash

Copy code

docker ps -a

1. **Remove the Container** (Optional):
   * If you want to remove the container, use:

bash

Copy code

docker rm <container-id>

(Replace <container-id> with the actual ID of the container from the docker ps -a output.)

**Step 3: Pull the Docker Ubuntu Image and Access the Bash Terminal**

1. **Pull the Ubuntu Docker Image**:
   * Run the following command to pull the official Ubuntu image from Docker Hub:

bash

Copy code

docker pull ubuntu

1. **Run the Ubuntu Container**:
   * Start an Ubuntu container in interactive mode with a bash terminal:

bash

Copy code

docker run -it ubuntu bash

* + This will launch the Ubuntu container and provide you with access to the bash shell inside the container.

1. **Run Commands Inside the Ubuntu Container**:
   * Once inside the container, you can run any Ubuntu command (e.g., update the package list):

bash

Copy code

apt update

1. **Exit the Container**:
   * To exit the bash terminal, type:

bash

Copy code

exit

1. **View Running Containers**:
   * To check if your container is still running, use:

bash

Copy code

docker ps

1. **Stop and Remove the Ubuntu Container**:
   * If you need to stop the container, use:

bash

Copy code

docker stop <container-id>

* + To remove it:

bash

Copy code

docker rm <container-id>

**Step 4: Explore Docker GUI Utilities**

Docker Desktop also includes a graphical interface for managing containers. Here’s how you can use it:

1. **Open Docker Desktop**:
   * Launch Docker Desktop from your applications or system tray.
2. **View Containers and Images**:
   * The **Dashboard** shows a list of running containers, images, and volumes.
   * You can interact with containers by stopping, starting, or removing them directly from the GUI.
3. **Run Containers from GUI**:
   * In the **Containers/Apps** section, you can click on the **Run** button to start a container from an image (e.g., Ubuntu or hello-world).
   * You can also access the container’s logs and statistics from the GUI.
4. **Manage Images**:
   * In the **Images** tab, you can view, pull, or remove Docker images.

**Step 5: Additional Docker Commands to Try**

Here are some additional Docker commands you may find useful:

* **List all running containers**:

bash

Copy code

docker ps

* **List all containers (running and stopped)**:

bash

Copy code

docker ps -a

* **List all Docker images**:

bash

Copy code

docker images

* **Remove an image**:

bash

Copy code

docker rmi <image-id>

* **List Docker volumes**:

bash

Copy code

docker volume ls

* **Remove a volume**:

bash

Copy code

docker volume rm <volume-name>

By following these steps, you'll successfully install Docker Desktop, use basic Docker commands, run your first container with the "hello-world" image, and interact with a container using the Ubuntu image's bash terminal.

Assignment 9:

**Docker Networking and Volumes: Pulling and Running the httpd Docker Image**

This guide will walk you through pulling the httpd image from Docker Hub, running the container, observing Docker internal network configurations, and mapping container ports with host ports. Additionally, you'll map a volume from your desktop to the container to serve web pages stored on your desktop.

**Step 1: Pull the httpd Docker Image**

1. **Open Docker Desktop** and make sure Docker is running.
2. **Open Terminal or Command Prompt** and run the following command to pull the official Apache httpd image from Docker Hub:

bash

Copy code

docker pull httpd

**Step 2: Run the httpd Container**

1. **Run the container**:
   * Run the Apache container and map port 80 inside the container to port 6543 on your host machine:

bash

Copy code

docker run -d -p 6543:80 --name my-apache httpd

* + Here’s what each option does:
    - -d: Runs the container in detached mode (in the background).
    - -p 6543:80: Maps port 80 inside the container (Apache’s default port) to port 6543 on the host machine.
    - --name my-apache: Names the container as my-apache.

1. **Check if the container is running**:
   * Run the following command to verify that the container is running:

bash

Copy code

docker ps

* + You should see a running container with the name my-apache.

1. **Verify Webpage Access**:
   * Open a browser and navigate to http://localhost:6543 (or http://<host-ip>:6543 if accessing from a different machine in the LAN).
   * You should see the default Apache "It works!" webpage.

**Step 3: Observe Docker Internal Network Configurations**

1. **Inspect the container’s network settings**:
   * Use the docker inspect command to view detailed information about the running container, including its network configuration:

bash

Copy code

docker inspect my-apache

* + Look for the "NetworkSettings" section to see details like IP addresses, ports, and network modes.

1. **View the container’s internal IP address**:
   * In the docker inspect output, locate the section for Networks and find the IPAddress to see the internal IP address of the container.
     + Example:

json

Copy code

"Networks": {

"bridge": {

"IPAddress": "172.17.0.2"

}

}

* + This is the internal IP address that Docker uses to communicate between containers.

**Step 4: Create Sample Webpages and Map Desktop Path with Docker Volume**

1. **Create Sample Webpages**:
   * On your host machine (your desktop or any folder), create a folder where you’ll store the webpages:
     + For example, create a folder C:\docker-webpages on Windows or /home/user/docker-webpages on Linux.
   * Inside this folder, create an HTML file, e.g., index.html with your sample webpage content:

html

Copy code

<html>

<head><title>Sample Webpage</title></head>

<body>

<h1>Welcome to My Webpage Hosted on Docker!</h1>

</body>

</html>

1. **Stop and Remove the Running Container** (Optional):
   * If you want to map the volume and re-run the container, stop and remove the existing container:

bash

Copy code

docker stop my-apache

docker rm my-apache

1. **Run the Apache Container with Volume Mapping**:
   * Use the -v flag to mount the folder from your desktop to the container:

bash

Copy code

docker run -d -p 6543:80 -v /path/to/your/docker-webpages:/usr/local/apache2/htdocs/ --name my-apache httpd

* + **Important**:
    - Replace /path/to/your/docker-webpages with the actual path to the folder you created on your desktop (e.g., /home/user/docker-webpages or C:\docker-webpages on Windows).
    - The /usr/local/apache2/htdocs/ path inside the container is where Apache serves files by default.

1. **Access Your Sample Webpage**:
   * Now, when you navigate to http://localhost:6543 (or http://<host-ip>:6543), you should see the "Sample Webpage" that you created on your desktop.

**Step 5: Verify the Volume Mapping**

1. **Check if the volume is properly mapped**:
   * You can verify the volume mapping by logging into the container and checking the content of the /usr/local/apache2/htdocs/ directory:

bash

Copy code

docker exec -it my-apache bash

cd /usr/local/apache2/htdocs/

ls

* + You should see your index.html file and any other files you placed in the folder on your desktop.

**Step 6: Clean Up**

1. **Stop and Remove the Container**:
   * To stop the container when you're done:

bash

Copy code

docker stop my-apache

* + To remove the container:

bash

Copy code

docker rm my-apache

1. **Remove the Image** (Optional):
   * If you no longer need the httpd image, you can remove it:

bash

Copy code

docker rmi httpd

**Summary**

* You have successfully pulled and run the httpd container, mapped ports between the container and your host machine, and served a webpage stored on your desktop using Docker volumes.
* The -v option in Docker is key to mapping local directories (like your desktop folder) to directories inside containers, allowing you to serve dynamic content or persist data across container restarts.