**Live Session III**

Tips/tricks, commands…

bandwith testing …

docker

----

[ Tips/tricks, commands… ]

Grep Recursively Without a Pipe

Very often, we will use find's **exec** option to grep for something. This is a basic way of emulating a "find in files" task:

find . -exec grep 'hello' {} \;

But this command is lengthy, this command is much more convenient and easier to learn:

**grep -r 'hello'**

Switch Between Two Directories Instantly

Each time you run "**cd -**" it will flip between the previous two directories you navigated to, making it very fast and easy to move back and forth.

"cd -" will print the path of the directory it changes to, even if you have the current directory in your prompt.

Copy Directories With Maximum Fidelity

cp -R docs backup-docs

This command lets you copy entire directory structures, producing a full copy of a directory and all its contents, including files and other directories. It's a great way of backing up files or creating a copy of a larger project to work on.

However, you may notice that the copies aren't exactly the same as the originals. Their modification times, owners, and permissions may all be different.

These times only differ by a minute, but the problem can be much greater. The fix is simple: use **-a** (for "archive") instead of -R:

**cp -a docs backup-docs**

[ bandwith testing … ]

Real Time Bandwith manager!

sudo apt-get update

sudo apt-get -y install vnstat

A screenshot of a computer program

Description automatically generated

sudo vnstat -l -i wlan0 (ex. listen in on wlan0 socket for ex.)

nmcli connection show to check socket names if nec.

Surf the web in browser and watch for bit transfer rates

**Ctrl^C** when complete for summary

[ docker ]

Docker (for httpd/php/mysql services, etc.)

Docker simplifies the process of application deployment by packaging software and its dependencies into containers. These containers are lightweight, portable, and can run consistently across *different* computing environments, **ensuring** that applications behave the same way regardless of where they are deployed. This capability is particularly valuable in modern software development, where applications are often distributed across various platforms and infrastructures.

Containers vs. Virtual Machines

Containers are often compared to virtual machines (VMs), but they operate differently. While VMs virtualize the entire hardware stack, containers share the host operating system's kernel and utilize its resources more efficiently. This results in faster startup times and lower overhead, allowing multiple containers to run on a single host without the resource constraints typically associated with VMs

Components of Docker

Docker consists of several key components:

* **Docker Engine**: The core service that runs and manages containers.
* **Docker Images**: Read-only templates used to create containers. They include everything needed to run an application, such as code, libraries, and environment variables.
* **Docker Containers**: Instances of Docker images that run the application in an isolated environment.
* [**Docker Hub**](https://hub.docker.com/): A cloud repository where users can share and access Docker images

Benefits of Using Docker

1. **Portability**: Docker containers can run on any system that supports Docker, making it easy to move applications between development, testing, and production environments.
2. **Efficiency**: Containers share the host OS kernel, reducing resource consumption compared to traditional VMs. This allows for higher density of applications on a single host.
3. **Speed**: Containers can be started and stopped quickly, facilitating rapid deployment and scaling of applications.
4. **Isolation**: Each container runs in its own environment, which enhances security and stability by preventing applications from interfering with one another

Let’s do it!

Install docker

# Update the apt package index and install packages to allow apt to use a repository over HTTPS:

**sudo apt-get update**

**sudo apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release**

Add Docker’s official GPG key:

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg**

Set up the stable repository

**echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**

Update the apt package index, and install the latest version of Docker Engine and containerd

**sudo apt-get update**

**sudo apt-get install -y docker-ce docker-ce-cli containerd.io**

Start Docker and enable it to run at boot:

**sudo systemctl start docker**

**sudo systemctl enable docker**

Create a Docker Container with httpd, PHP, and MySQL

You can create a Docker Compose file to define your services. Create a file named   
docker-compose.yml:

Open editor to compose .yml file

Create file and enter script info as follows:

**sudo nano docker-compose.yml**

version: '3.8'

services:

web:

image: nginx:latest

ports:

- "80:80"

db:

image: mysql:latest

environment:

MYSQL\_ROOT\_PASSWORD: rootpassword

MYSQL\_DATABASE: mydatabase

MYSQL\_USER: user

MYSQL\_PASSWORD: userpassword

volumes:

- db\_data:/var/lib/mysql

volumes:

db\_data:

**\*Note port 80 above installs webpages that will run on the default listed 80 port. Can be also another port like 8080:80**

Create a Simple PHP Page

Create a directory named **html**in your current working directory and add a file named index.php with the following content:

**mkdir html**

**cd html**

**sudo nano index.php**

Add the following contents

<?php

phpinfo();

?>

Save contents and exit editor.

Run Docker Compose

Run the following commands to start the containers:

First download docker compose as follows

**sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose**

**sudo chmod +x /usr/local/bin/docker-compose**

Second apply permissions to docker-compose binary

**sudo chmod +x /usr/local/bin/docker-compose**

Then execute the following command

**sudo docker-compose up -d**

**View web page in Firefox, type**

[**http://localhost:8080**](http://localhost:8080)

or do a curl request command

**curl localhost:8080**

Create a Database and Table in MySQL

You can access the MySQL container and create a database and table. Use the following commands:

Run status and notate container id number for MySQL container

**sudo docker ps**

# Access the MySQL container

**sudo docker exec -it <your\_container\_id\_of\_db> mysql -u root -p**

Enter **root password** per your .yml file you created above.

# Inside the MySQL shell, execute the following:

**CREATE DATABASE testdb;**

**USE testdb;**

**CREATE TABLE users (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), email VARCHAR(255));**

**INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com'), ('Jane Doe', 'jane@example.com');**

SELECT \* FROM users;

Refs

<https://www.zdnet.com/article/kde-neon-shows-that-the-plasma-6-linux-distro-is-something-truly-special/>

<https://www.bing.com/videos/riverview/relatedvideo?q=kde%20Desktop%20Cube&mid=4A279441455BF03A49D04A279441455BF03A49D0&ajaxhist=0>

<https://www.bing.com/videos/riverview/relatedvideo?q=kde%20Desktop%20Cube&mid=4F9520E7EC29DE6B05A64F9520E7EC29DE6B05A6&ajaxhist=0>

<https://www.bing.com/videos/riverview/relatedvideo?q=kde%20Desktop%20Cube&mid=3ED909A7878B6BB8AC8F3ED909A7878B6BB8AC8F&ajaxhist=0>

Alt distros

NIXOS

<https://www.zdnet.com/article/sparky-linux-is-a-blazing-fast-distro-that-can-keep-your-older-machines-running-for-years/>

<https://www.zdnet.com/article/how-to-install-ubuntu-linux/>

Kali Purple