▧ Docker Demo, Dew Computing and Serverless Computing – Revision Notes

◇ **Docker Demo – MySQL & PHPMyAdmin**

- **MySQL**:
  - Open-source relational database
  - Stores data in structured format using tables
- **PHPMyAdmin**:
  - Web-based GUI for managing MySQL databases
  - Enables easy CRUD operations

Without Docker (Standalone Setup):

- Requires separate installation of MySQL, Apache, PHP, PHPMyAdmin
- Hard to transfer across machines
- Backup/restore must be done manually

With Docker (Containerized Setup) – Benefits:

- **Separation of Responsibility**: Developers focus on code, IT manages deployment
- **Portability**: Run on any OS – Windows, Linux, Mac
- **Isolation**: Each container is logically separated
- **Lightweight**: Faster and uses less memory than VMs

◇ **Containers vs Virtual Machines**

Feature	Containers	Virtual Machines
Virtualization Level	OS level	Hardware level
Performance	Lightweight, fast startup	Heavy, slow startup
Kernel	Shared	Separate for each VM
Resource Usage	Low	High

◇ **Dew Computing (DC)**

- **Definition**:
  - Combines **cloud computing** with **end device capabilities**
  - Supports offline work with auto-sync when online
  - First in **IoT–Fog–Cloud** continuum

Key Features:

- **Independence**: Works offline
- **Collaboration**: Syncs with cloud when available
- **Micro-service Based**: No need for centralized servers
- **Located close to the user (at the "ground" level)**

Example: Dropbox

- Offline file access + cloud sync = Dew Computing

Dew Computing Architecture Goals:

1. **Data Replication**
2. **Data Distribution**
3. **Synchronization**

Dew Service Models:

- **Infrastructure as Dew**: iCloud
- **Software as Dew**: Play Store / App Store
- **Platform as Dew**: GitHub
- **Storage in Dew**: Dropbox, Google Drive
- **Web in Dew**: Pocket

Application Areas:

- **WiD**: Offline web copy + sync
- **SiD**: Local + cloud file storage
- **DBiD**: Redundant databases (local & cloud)
- **PiD**: Development tools + synced settings (e.g., GitHub)

Challenges:

- Power management
- Processor utilization
- OS viability
- Programming principles
- **Database security**

◇ **Serverless Computing**

- **Definition**:
  - Backend services provisioned on-demand
  - Developers only write logic – no server setup needed

Advantages:

- No server management
- Auto-scaling
- Faster deployment
- Event-driven model

Two Types:

1. **BaaS (Backend-as-a-Service)**:
   - Provides backend services like auth, DB, storage
   - Example: Firebase, AWS Amplify
2. **FaaS (Function-as-a-Service)**:
   - Executes small, event-triggered functions
   - Example: AWS Lambda, Azure Functions

FaaS Features:

- Event-driven
- Short execution time
- Auto-scaling
- Stateless

Serverless Challenges:

- **Asynchronous calls**
- **Function chaining**
- **Code sharing** between functions
- **Library overload**
- Managing many small functions

🔁 **Quick Memory Tips:**

Topic	**Quick Tip**
Dew Computing	"Dropbox works offline"
Containers	"Lightweight, OS-level"
Serverless	"Write code, forget infra"
BaaS	"Pre-built backend services (e.g. Firebase)"
FaaS	"Small code blocks triggered by events"