Geographic Information System (GIS) and Docker – Revision Notes

☑ **Geographic Information Systems (GIS)**
- **Definition**: A computer system for capturing, storing, querying, analyzing, and displaying **geospatial data**.
- **Use**: Converts spatial data into actionable information for decision-making.

☑ **Components of GIS**
- **Hardware**
- **Software**
- **Data Management & Analysis Procedures**
- **Spatial Data**
- **People** (operators/users

☑ **Challenges in GIS**
- **Data Intensive** & **Computation Intensive**
- **Variable Load** on GIS server → Requires **dynamic scaling**
- High **reliability** and **performance** needs
- **Network-intensive** due to heavy web service usage

☑ **Heterogeneity Issue in GIS**
- Caused by **diverse software systems** in departments
- **Barriers in data sharing**
- **Solutions**:
  - Homogeneous data description
  - Standard data encoding
  - Common data sharing mechanisms

☑ **Spatial Data Infrastructure (SDI)**
- **Definition**: Coordinated policy & infrastructure for spatial data discovery and usage
- **Users**: Government, commercial, academia, non-profits, citizens

☑ **Need for Geospatial Cloud**
- Large volume of data & metadata
- Demand for **services** and **service orchestration**
- Requires **scaling**, **policy evolution**, and **multi-tenancy**

☑ **Actors in Geospatial Cloud**
- **Brokers**
- **Customers**
- **Negotiators**
- **SLA Manager / Security Auditor**

☑ **Challenges in Geospatial Cloud**
- **Spatial DB implementation & scaling**
- **Multi-tenancy** and **policy management**
- **Geo-situated backups**
- **Data security**

☑ **Interoperability in Geospatial Systems**
1. **Data Level** → Consume data
2. **Service Level** → Exchange & access data
3. **Security Level** → Trustworthy & reliable access
   ☑ **Solution**: Use **OGC standards**

Introduction to Docker & Containerization

☑ **Docker**
- A **container management service** (since March 2013)
- Lets you **develop, ship & run apps anywhere**
- **Lightweight** alternative to Virtual Machines

☑ **Docker Features**
- **Smaller OS footprint**
- **Easier collaboration** among Dev, QA, and Ops
- **Cross-platform deployment**
- **Highly scalable**

☑ **Docker Components**
- **Docker for Mac / Linux / Windows**
- **Docker Engine** – builds & runs containers

- **Docker Hub** – public registry of images
- **Docker Compose** – multi-container app config

☑ **Docker Architecture**

- **Server**: Hosts containers
- **Host OS**: Base system (Linux/Windows)
- **Docker Engine**: Manages containers
- **Apps run inside containers**, not VMs

☑ **Containers**

- Packages app code + dependencies
- Share OS kernel → faster & lighter than VMs
- Run as **isolated user-space processes**
- **Image**: Standalone package (code + runtime + config)
- **Container**: Running instance of an image

☑ **VMs vs Containers**

Feature	Virtual Machine	Docker Container
OS	Includes Guest OS	Shares Host OS kernel
Size	Heavy (GBs)	Lightweight (MBs)
Boot Time	Minutes	Seconds
Isolation	Full OS Isolation	Process-level isolation
Portability	Less portable	Highly portable
Resource Usage	High	Low

☑ **Docker Hub**

- **Public Docker image registry**
- Offers **automated builds** from GitHub Dockerfiles
- URL: https://hub.docker.com

☑ **Docker Use Cases**

- Fixes **"works on my machine"** problems
- Run **apps side-by-side** in containers
- Create **CI/CD pipelines** for agile deployment