**Docker & Kubernetes Series - Session 1: Docker Fundamentals**

**GitHub Repository:** [Link] (Contribute for updates)

**Index**

1. **Containers vs Virtual Machines (VMs)**
2. **What is a Container?**
3. **Docker Architecture**
4. **Basic Docker Commands**
5. **Networking in Docker**
6. **Port Forwarding**
7. **Logs & Container Inspection**
8. **Storage Management in Docker**

**1. Containers vs Virtual Machines (VMs)**

**Virtual Machines (VMs)**

* **Components:**
  + **Infrastructure (Data Center)** → **Hypervisor (VMware, Hyper-V)** → **Guest OS (Each with its own Kernel)** → **Applications**
  + Each VM has its own OS and kernel.
* **Disadvantages:**
  + **Resource-Heavy:** Each VM consumes CPU, RAM, and storage independently.
  + **Environment Inconsistency:** An app working in Dev (Ubuntu) may fail in QA (Windows) or Prod (Mac).

**Containers**

* **Components:**
  + **Infrastructure (AWS/Azure)** → **Host OS** → **Container Engine (Docker, ContainerD, CRI-O)** → **Containers**
  + **No Guest OS/Kernel:** Containers share the host OS kernel.
* **Advantages:**
  + **Lightweight:** No separate OS per container.
  + **Consistent Environments:** Works across Dev, QA, Prod.

**2. What is a Container?**

* A **lightweight, isolated process** running on a host machine.
* Shares the host OS kernel but has its own filesystem, CPU, memory, and network.
* **Example:**
  + A notepad.exe file interacts with the OS kernel → kernel translates request to hardware.
  + Similarly, a containerized app interacts with the Docker engine.

**3. Docker Architecture**

**Components:**

1. **Docker Client (**docker**CLI commands)**
2. **Docker Daemon (Background process managing containers)**
3. **Docker Registry (Stores images, e.g., Docker Hub)**

**Flow:**

1. User runs docker run nginx (Client sends HTTP API call to Daemon).
2. Daemon checks local cache → If not found, pulls image from Registry.
3. Runs container from the image.

**Key Notes:**

* **Docker in Production:**
  + Developers use **Docker Desktop** locally.
  + Production uses **ContainerD** (not Docker Engine).

**4. Basic Docker Commands**

| **Command** | **Description** |
| --- | --- |
| docker --version | Check Docker version |
| docker run -d --name app1 nginx | Run container in detached mode |
| docker ps | List running containers |
| docker ps -a | List all containers (including stopped) |
| docker stop <container> | Stop a container |
| docker rm <container> | Remove a container |
| docker rmi <image> | Remove an image |
| docker logs -f <container> | Stream container logs |
| docker inspect <container> | Inspect container details |

**Bulk Operations:**

bash

Copy

Download

*# Stop all containers*

docker stop $(docker ps -aq)

*# Remove all containers*

docker rm $(docker ps -aq)

*# Remove all images*

docker rmi $(docker images -q)

**5. Networking in Docker**

* **Default Network:** docker0 (Bridge network, IP: 172.17.0.1)
* **How Containers Communicate:**
  + Containers → docker0 → Host’s eth0 → Internet.
* **View Networks:**

bash

Copy

Download

docker network ls

ifconfig (Check docker0 interface)

**6. Port Forwarding**

* Maps host port to container port.
* **Example:**

bash

Copy

Download

docker run -d --name frontend -p 8000:80 nginx

docker run -d --name backend -p 8001:80 nginx

* + Access:
    - http://<host-ip>:8000 → frontend (Port 80 inside container).
    - http://<host-ip>:8001 → backend.

**7. Logs & Container Inspection**

**View Logs:**

bash

Copy

Download

*# Method 1 (Basic)*

docker logs -f <container>

*# Method 2 (Advanced, using JQ for pretty logs)*

cat /var/lib/docker/containers/<container-id>/<container-id>-json.log | jq

**Inspect Container:**

bash

Copy

Download

docker inspect <container>

docker inspect <container> | grep IPAddress *# Get container IP*

**8. Storage Management**

* **Default Storage:** /var/lib/docker (Avoid filling root volume).
* **Best Practice:**
  + Use **EBS Volume** for Docker data in production.
  + Monitor disk usage:

bash

Copy

Download

df -h

docker system df *# Check Docker disk usage*

**Mind Map**

Copy

Download

Docker Fundamentals

├── Containers vs VMs

├── Docker Architecture (Client-Daemon-Registry)

├── Basic Commands (run, ps, stop, rm, logs)

├── Networking (docker0, Port Forwarding)

├── Logs & Inspection

└── Storage Management (EBS for /var/lib/docker)