# **🚀 DOCKER CHEAT SHEET — MASTER LEVEL SUMMARY**

## **🌟 START POINT — ON YOUR WINDOWS PC:**

You have:

* Windows OS
* 1 TB Disk (ROM)
* 16 GB RAM

## **🐳 STEP 1 → INSTALL DOCKER DESKTOP**

When you install **Docker Desktop**, it sets up:

✅ Docker Daemon → dockerd process  
 ✅ A **Linux Virtual Machine** (via:

* Hyper-V
* OR WSL2)

👉 WHY? → Because:

* Linux containers need a **Linux Kernel**.
* Windows Kernel cannot run Linux containers.
* So **Hyper-V/WSL2** runs a small **Linux VM**, which gives a **Linux Kernel**.

## **🎮 STEP 2 → DOCKER DAEMON**

When you **start Docker Desktop** → it starts:

→ dockerd = background process → manages containers → listens for your docker CLI commands.

## **📦 STEP 3 → BUILD DOCKER IMAGE**

You write a **Dockerfile**, example:

Dockerfile

CopyEdit

FROM openjdk:17-jdk-slim

COPY target/myapp.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

You run:

bash

CopyEdit

docker build -t myapp .

→ This creates a **Docker Image** → stored on disk (ROM) → in:

bash

CopyEdit

/var/lib/docker

## **🚦 WHAT IS IN DOCKER IMAGE?**

Contains:

✅ Minimal Linux user-space libs (ex: Debian-slim → ~25 MB)  
 ✅ OpenJDK 17 → JVM (~150 MB)  
 ✅ Your app.jar (~50 MB)

## **🏃 STEP 4 → RUN DOCKER CONTAINER**

You run:

bash

CopyEdit

docker run -m 1g -p 8080:8080 myapp

What happens:

→ Docker Daemon (dockerd) asks **Linux Kernel** inside the VM:

"Please start a container with 1 GB RAM, mapped to port 8080."

## **🔍 WHAT IS A CONTAINER?**

* It is just an **isolated Linux process**.
* Created by host Linux Kernel.
* Isolation done using:  
  + **namespaces** → hides other processes / networks.
  + **cgroups** → limits RAM/CPU.

## **🧠 HOW JVM PROCESS RUNS?**

Inside container:

bash

CopyEdit

java -jar app.jar

* The java binary is from **inside the image** (not host JDK).
* The JVM process runs on:  
  + **Host CPU** (your laptop CPU).
  + **Host RAM** (your 16 GB RAM).

## **💾 HOW DISK (ROM) WORKS?**

* The Docker image is stored on host disk (ROM).
* Container can also read/write files → mapped to host disk (volumes).

## **📌 FINAL VISUAL → END TO END**

lua

CopyEdit

|===========================|

| Windows OS + Kernel |

|===========================|

| Hyper-V OR WSL2 |

|---------------------------|

| Linux Kernel (in VM) |

|---------------------------|

| Docker Daemon (dockerd) |

|---------------------------|

| Container → runs JVM |

| - Linux libs (25 MB) |

| - JVM (150 MB) |

| - app.jar |

|---------------------------|

| Process: java -jar app.jar|

|---------------------------|

| RAM used → Host 16 GB RAM |

| CPU used → Host CPU |

| Disk used → Host 1TB Disk |

|===========================|

# **🎁 SUMMARY SENTENCE:**

👉 When you run docker run, **Docker Daemon** tells **Linux Kernel** to start an isolated **JVM process** inside a container → which uses **host CPU and RAM**, with JVM binary from inside **Docker image**, and Linux Kernel for process management.

A linux vm has something called namespace and we can run any number of processes(containers) in linux vm’s

And basically when we download and run the docker…it creates a process that starts the docker daemon, that is responsible for creating a linux vm using hyper -v or others to have an linux kernel and listens for docker commands from cli or (other places)...  
Now the responsibility of creating a container is achieved by having the already mentioned linux kernel by creating a linux vm inside your windows as windows can't have linux kernel  
Now these kernels look for docker commands and creates a process for the command in case  
now when u dockerize a spring app…the kernel takes care of creating a java process using image(lightweight linux libs+jdk+app)..now the best thing here is somehow the linux vm’s namespace takes care of the java process…it is an isolated process and does not depend on host machine…means the container exposes a port ..not the host machine

ONE Linux VM → one Kernel → can run **many containers**.  
Each container is an isolated process inside that one Kernel.

Docker Desktop just manages this.  
But the container would need the memory from the host machine though..which we will specify in case of ec2 and for fargate..its get taken care by aws  
And also docker would manage all these container created