🚀Docker, systemd & PID 1 — Deep Notes (No Shortcuts)

> **How to use:** Har heading ke niche explanation + examples + WHY/WHAT/WHEN diya gaya hai. Isko VC me slide/script ki tarah follow karo.

PART 1 — FOUNDATIONS (Base banate hain)

Computer, Program & Process

Concept: Computer khud decision nahi leta; OS instructions ko execute karwata hai.

Program (Static): Disk par padi binary/script. Jab tak run nahi hota, CPU/RAM consume nahi karta. - Example: `/usr/sbin/nginx`, `node`, `python`

Process (Dynamic): Program jab RAM me load hota hai aur CPU time leta hai. - Example: `nginx` running, `node app.js`

WHY important: Docker/OS hamesha *process* ko manage karta hai, program ko nahi.

Process lifecycle

Stages: `Create → Run → Wait → Exit` - **Create:** OS memory allocate karta hai, PID assign karta hai. - **Run:** Scheduler CPU time deta hai. - **Wait:** I/O ya child-process ka wait. - **Exit:** Process khatam, parent ko status milta hai.

WHY: Cleanup ka responsibility parent (akhir me PID 1) ki hoti hai.

PID kya hota hai

PID (Process ID): Har running process ka unique number. - PID reuse ho sakta hai (process exit ke baad).

Commands:

```
ps aux
ps -p <pid> -o pid,ppid,cmd
```

WHY: Debugging, signal sending (`kill`), monitoring ke liye PID zaroori.

PID 1 — system ka asli boss

Role: - Sab processes ka *ultimate parent* - Zombie cleanup - Signal handling (SIGTERM/SIGKILL) - Shutdown orchestration

Rule: PID 1 galat ho ⇒ leaks, hangs, dirty shutdowns.

PART 2 — LINUX INTERNALS (Andar tak)

Linux process tree

Hierarchy: Har process ka parent hota hai; root parent = PID 1.

```
PID 1 (systemd)
 ├─ sshd
 │   └─ bash
 └─ nginx
```

WHY: Parent signal forward kare ya na kare—child behavior decide hota hai.

Zombie process

Zombie: Child exit ho gaya, parent ne `wait()` nahi kiya. - Entry process table me rehti hai.

Cleanup: PID 1 orphan zombies ko reap karta hai.

WHY Docker me issue: Bash PID 1 zombies clean nahi karta.

Linux signals

Signals = OS messages - `SIGTERM` → Graceful stop (cleanup chance) - `SIGKILL` → Immediate kill (no cleanup)

Commands:

```
kill -TERM <pid>
kill -KILL <pid>
```

PART 3 — BOOT PROCESS & SYSTEMD

Linux boot flow

1. BIOS/UEFI
2. Bootloader (GRUB)
3. Kernel
4. init
5. **systemd (PID 1)**

WHY: systemd tabhi possible jab full OS boot ho.

---

systemd kya hai

systemd = init + service manager + dependency resolver + logger

Features: - Parallel service start - Dependency graph - Auto-restart - Resource limits

---

🚗systemd = PID 1

- First user-space process
- Services ka parent
- Clean shutdown guarantee

WHY containers me nahi: Containers OS boot nahi karte.

---

systemd components

- **systemctl:** Control CLI
- **journald:** Central logging
- **logind:** Sessions/users
- **timers:** cron replacement
- **targets:** runlevel replacement

Unit files & lifecycle

Locations: `/etc/systemd/system`, `/lib/systemd/system`

Example:

```
[Service]
ExecStart=/usr/sbin/nginx
Restart=always
```

Lifecycle: start → running → stop → restart

PART 4 — UBUNTU + NGINX REALITY

`apt install` ke peeche kya hota hai

- Package download
- Unit file register
- `systemctl start` (post-install)
- Enable on boot

Ubuntu VM me nginx auto-start kyun

Reason: systemd pehle se PID 1 hota hai; services allowed to auto-start.

PART 5 — DOCKER INTERNALS (Most Important)

Docker asal me kya hai

Docker = process isolation - Namespaces + cgroups - Kernel shared

---

Docker ≠ VM

- VM: full OS + kernel
- Docker: single process

Mental model: Docker *process runner* hai.

Container lifecycle

`create → start → run → stop → delete`

Key: PID 1 decide hota hai *start time* par.

Docker me PID 1 kaise decide hota hai

Rule: `CMD/ENTRYPOINT` jo run hota hai wahi PID 1.

---

systemctl Docker me kyun fail hota hai

- systemd absent
- OS boot nahi hota

PART 6 — NGINX + PID 1 (CORE)

nginx background vs foreground

- Default: daemon (background)
- Docker: foreground chahiye

`daemon off` ka matlab

```
nginx -g 'daemon off;'
```

- Background disable - Foreground me run - Signals receive

---

docker stop vs docker kill

- `stop`: SIGTERM → grace
- `kill`: SIGKILL → no grace

Bash PID 1 kyun dangerous

- Signals forward nahi
- Zombie cleanup nahi

---

nginx PID 1 kyun best

- Proper signal handling
- Graceful shutdown
- Production safe

PART 7 — NODE + NGINX (Real World)

Node image ka role

- Build-time tool
- Runtime ke liye nginx better

Galat approach

```
docker run node bash
```

- PID 1 = bash ❌

Sahi approach

```
CMD ["nginx", "-g", "daemon off;"]
```

- PID 1 = nginx ✅

Multi-stage build philosophy

- Build stage: node
- Run stage: nginx
- Smaller, safer image

PART 8 — WINDOWS & KUBERNETES

Windows Service Control Manager

- `services.exe`
- systemd equivalent

Kubernetes me PID 1 ka role

- SIGTERM first
- Grace period
- Restart on exit

PART 9 — FINAL MINDSET

Common misconceptions

- Docker = VM ❌
- systemctl in container ❌

Interview traps

Wrong: Docker me systemd hota hai **Correct:** Docker runs processes, not OS

Golden rules

1. Docker OS nahi
2. PID 1 matters
3. Foreground process mandatory

Final mental model

systemd = OS brain Docker = process runner

End Goal: Is doc ko explain kar paoge ⇒ Docker/Linux tumhare control me.