# Docker Namespaces – Brief Explanation with Simple Examples

Docker uses Linux namespaces to isolate containers from each other. Each namespace provides a container with its own view of a specific resource. Below are brief explanations of each namespace type along with simple examples.

## 1. PID Namespace (pid)

Isolates process IDs.  
Each container has its own set of PIDs.

Example:  
Run a container and check process IDs inside and from host:  
Inside container: `ps` shows PID 1 for the main process.  
On host: `docker top <container\_id>` shows a different PID.

## 2. Network Namespace (net)

Isolates network interfaces, IP addresses, and ports.  
Each container has its own network stack.

Example:  
Run: `docker run -it alpine ip a`  
Shows container’s own network interface (e.g., eth0) and IP.

## 3. Mount Namespace (mnt)

Isolates filesystem mount points.  
Each container has its own root filesystem.

Example:  
Run: `docker run -it alpine ls /`  
Shows a different filesystem view than the host.

## 4. UTS Namespace (uts)

Isolates hostname and domain name.  
Each container can have a different hostname.

Example:  
Run: `docker run -it --hostname=mycontainer alpine hostname`  
Displays: `mycontainer`.

## 5. IPC Namespace (ipc)

Isolates inter-process communication mechanisms.  
Processes in one container can’t access IPC of another.

Example:  
Run two containers and try using shared memory (e.g., `/dev/shm`).  
They do not share IPC unless explicitly configured.

## 6. User Namespace (user)

Isolates user and group IDs.  
A root user inside a container can map to a non-root user on host.

Example:  
Enable user namespace remapping in Docker daemon config.  
Inside container: `id` shows UID 0 (root), but on host it's mapped to an unprivileged user.