**Namespaces and their types in Docker**  
  
  
In Docker, namespaces are **used to isolate processes and resources**, so that **each container can have its own view of the system**. Docker uses multiple types of namespaces to achieve this isolation. Here are some of the namespaces used by Docker:

1. **PID Namespace:** This namespace **provides isolation of the process ID number space**. Each container has its own set of process IDs, which are separate from those of the host system and other containers.
2. **Network Namespace:** This namespace **provides isolation of the network resources**, such as **network interfaces, routing tables, and firewall rules**. Each container has its own network namespace, which means that it has **its own network stack and its own IP address.**
3. **Mount Namespace**: This namespace provides **isolation of the filesystem mount points**. Each container has its own mount namespace, which means that it has its own root filesystem and cannot see or modify the host system's filesystem.
4. **IPC Namespace:** This namespace provides isolation of the **IPC (interprocess communication) resources**, such as **message queues and shared memory**. Each container has its own IPC namespace, which means that it cannot see or access the IPC resources of other containers or the host system.
5. **UTS Namespace:** This namespace **provides isolation of the hostname and domain name of the container.** Each container has its own UTS namespace, which means that it has its own hostname and domain name, separate from those of the host system and other containers.
6. **User Namespace:** This namespace **provides isolation of the user and group IDs**. Each container has its own user namespace, which means that it has its own set of user and group IDs, separate from those of the host system and other containers. This can help to mitigate some security risks associated with running containers as root.

By **using** these different **types of namespaces**, **Docker** is able to **provide strong isolation between containers**, while still allowing them to share the same underlying host system resources.