Docker

Container Architecture

Container Architecture (Docker)

Docker is a client-server application where both the daemon and client *can* be run on the same system or you can connect a Docker client with a remote Docker daemon.

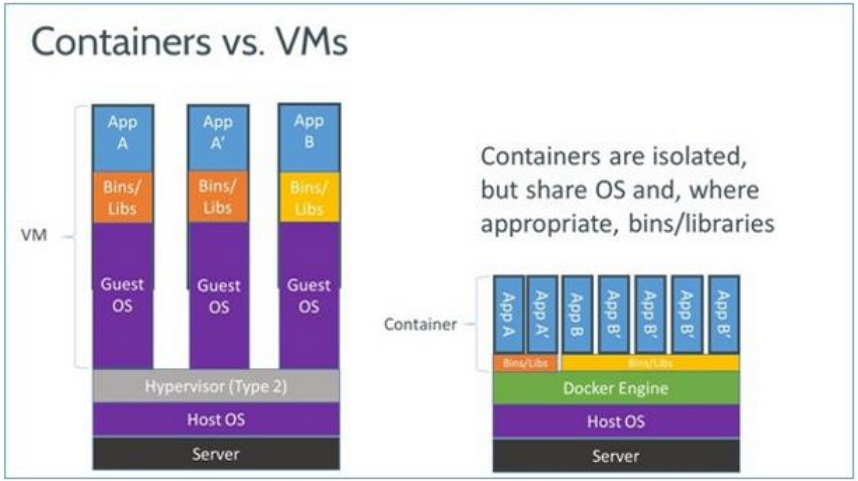
Docker clients and daemons communicate via sockets or through a RESTful API *(Representational State Transfer – it is a stateless transfer over HTTP of a web page containing an XML file that describes and includes the desired content).*

The main components of Docker are:

- Daemon
- Client
- Docker.io Registry

Container Architecture

Instead of virtualizing hardware, containers rest on top of a single Linux instance. This allows Docker (or generic LXC) to leave behind a lot of the bloat associated with a full hardware hypervisor. Here is a look at the architecture of each:



Don't mistake the *Docker Engine (or the LXC process)* as the equivalent of a hypervisor in a more traditional VM, it is simply the encapsulating process on the underlying system.

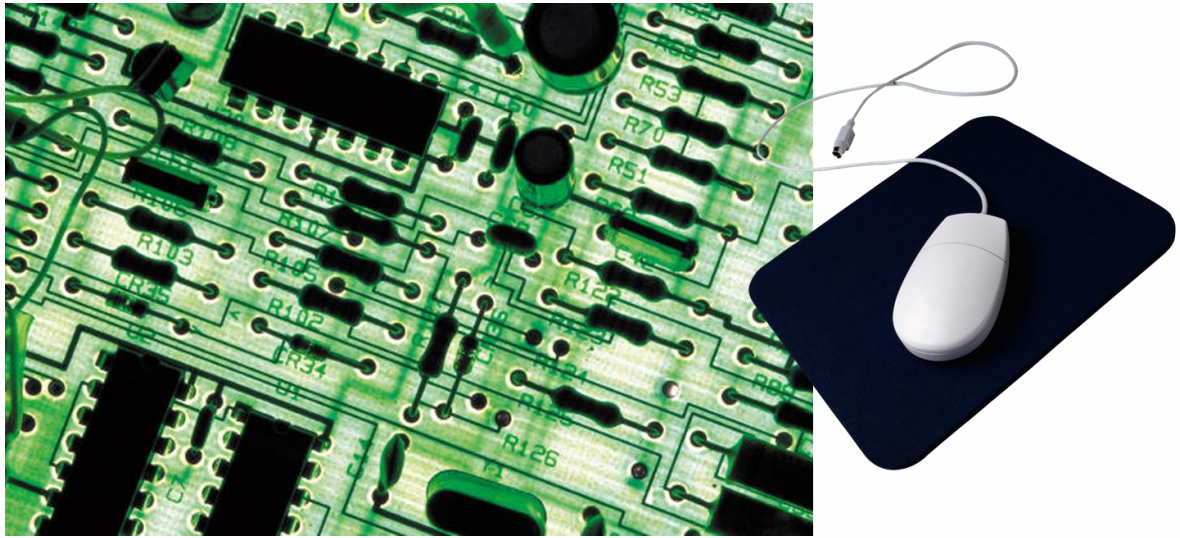
Hasn't This Already Been Done?

Quite simply, yes. Containers are not a new concept in technology, it just appears that Docker has captured the buzz (right place, right time). If you look around, you will find that a number of companies and projects have been working on the concept of application virtualization for some time:

- FreeBSD – Jails
- Sun (and now Oracle) Solaris – Zones
- Google – lmctfy (Let Me Contain That For You)
- OpenVZ

We appear to have reached the point in time where software has caught up to hardware virtualization.

Summary



The architecture of Docker and the containers that it relies on are not new concepts, having been around since the early part of this century. However, hardware virtualization performance has now become almost indistinguishable from bare metal so that further virtualization on the technology stack can be realized.