Introduction to Containers: Concept, Pros and Cons, Orchestration, Docker, and Other Alternatives

Container-based virtualization is one of the hottest technologies in cloud computing today. It fuels innovation and changes how applications are developed and operated today. It is difficult to name any of the bigger tech companies who are not investing in nor using container technology in one way or another. The idea of containers is not new, Linux-based operating systems had the technology available since the early 2000's. However, it needed Docker's appearance on the scene to really make it into the mainstream of application development. In this article, we provide a simple introduction to container-based virtualisation technology by discussing, the concept, benefits, shortcomings, orchestration and some of the key players.

What Are Containers ?

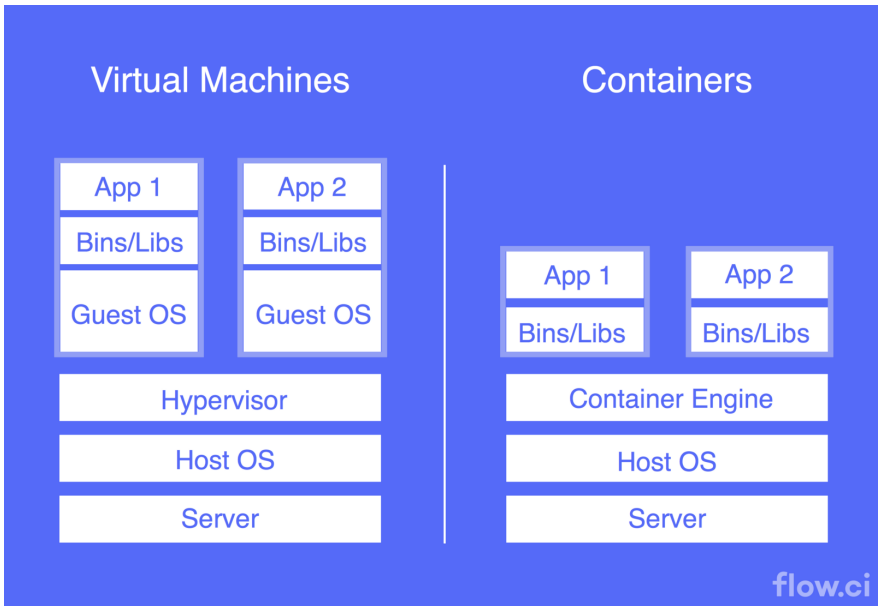
Containers, or otherwise known as operating-system-level virtualization, are a lightweight approach to virtualization that only provides the bare minimum that an application requires to run and function as intended. In a way, they can be considered as super minimalist virtual machines that are not running on a hypervisor. Items usually bundled into a container include:

- Application
- Dependencies
- Libraries
- Binaries
- Configuration files

Containerizing an application enables it to run reliably in different environments by abstracting away the operating system and the physical infrastructure. Containerized applications are sharing the kernel of the host operating system with other containers and the shared part of the OS is read only. Inside a container, there is often a single executable service or microservice.

The size of the containers is usually measured in tens of megabytes and it only takes 1–2 seconds to provision one. When the load is increasing new containers can be created and when the load drops containers can be destroyed. When containers need to be updated you only need to modify the configuration file and then create new containers and destroy the old ones.

What Is the Difference Between Virtual Machines and Containers?



Simplified model of virtual machines and containers

Even though virtual machines and containers share some characteristics there are significant differences setting them apart. If you take a look at the picture above it easy to point out them:

- Virtual machines contain a complete operating system and applications.
  Hypervisor-based virtualization is resource intensive, a VM can take up several GB depends on the guest the OS.
- Virtual machines use hypervisors to share and manage hardware while containers share the kernel of the host OS to access the hardware.
- Virtual machine have their own kernel and they don't use and share the kernel of the host OS, hence they isolated from each other at a deep level.
- Virtual machines residing on the same server can run different operating systems. One VM can run Windows while the VM next door might be running Ubuntu.
- Containers are bound by the host operating system, containers on the same server use the same OS.
- Containers are virtualizing the underlying operating system while virtual machines are virtualizing the underlying hardware.

What Are the Advantages of Using Containers?

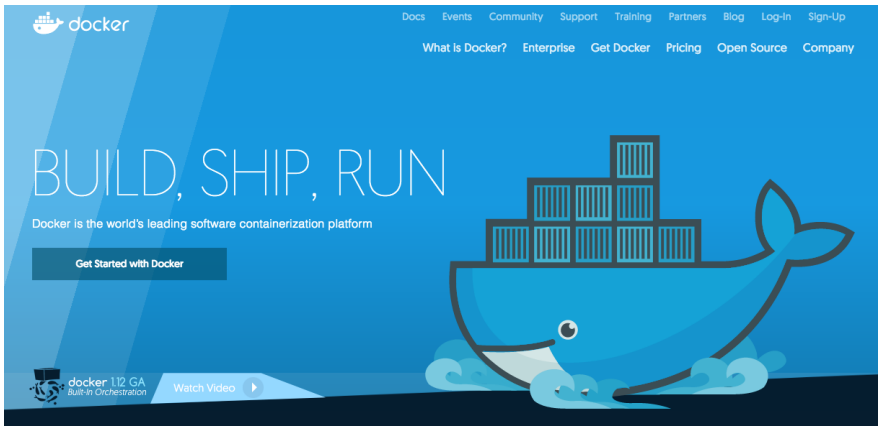
- The average container size is within the range of tens of MB while VMs can take up several gigabytes. Therefore a server can host significantly more containers than virtual machines.
- Running containers is less resource intensive then running VMs so you can add more computing workload onto the same server.
- Provisioning containers only take a few seconds or less, therefore, the data center can react quickly to a spike in user activity.
- Containers can enable you to easily allocate resources to processes and to run your application in various environments.
- Using containers can decrease the time needed for development, testing, and deployment of applications and services.
- Testing and bug tracking also become less complicated since you there is no difference between running your application locally, on a test server, or in production.
- Containers are a very cost effective solution. They can potentially help you to decrease your operating cost (less servers, less staff) and your development cost (develop for one consistent runtime environment).
- Container-based virtualization are a great option for microservices, DevOps, and continuous deployment.

What Are the Disadvantages of Using Containers ?

- One of the main disadvantages of container-based virtualisation compared to traditional virtual machines is security. Containers share the kernel, other components of the host operating system, and they have root access. This means that containers are less isolated from each other than virtual machines, and if there is a vulnerability in the kernel it can jeopardize the security of the other containers as well.
  Virtual Machines only share the hypervisor which has less functionality and less prone to attacks than the shared kernels of the containers. The system hardware is presented to the VMs in a virtualized form so intrusions, viruses, and other malicious activities cannot spread over to other VM.

- Less flexibility in operating systems. You need to start a new server to be able to run containers with different operating systems. While virtual machines with any kind of OS can live next to each other on the same server. This might not be a problem for hosting providers, but for complex enterprise application this can be a serious constrain.
- Another challenge is networking. Deploying containers in a sufficiently isolated way while maintaining an adequate network connection can be tricky. There are solutions that are addressing this issue such as Weave Net 1.5 but as it looks like at the moment there is still room to improve.

Docker and Other Containers Alternatives



Screenshot of docker.com

Docker is by far the most popular containerisation platform. Based on previously available open source technologies Docker created a standard way to deploy Linux applications into containers which then can run in different environments as intended. Docker has several open source projects serving as the basis for the Docker platform. The company was founded in 2010 by Solomon Hykes and in 6 rounds in they raised $180.8 million. They built partnerships with technology giants including Microsoft and Google and in Cluster HQ's Container Market Adoption Survey 2016 94% of the respondents said that they use Docker.

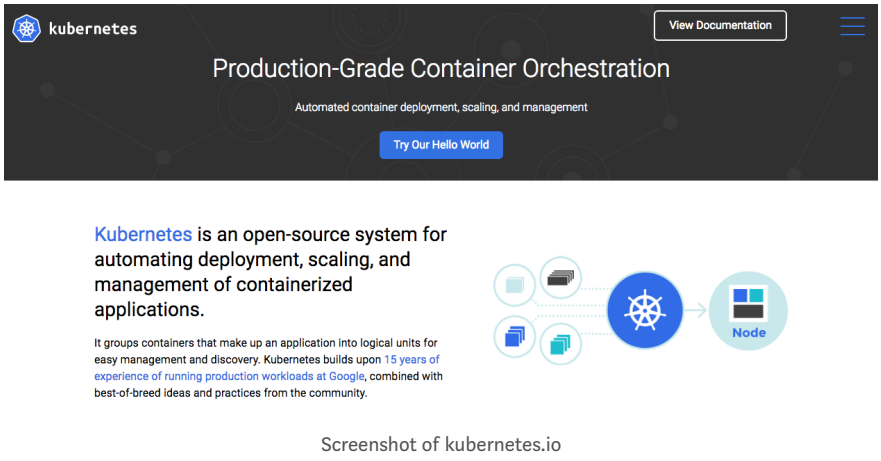
Although Docker is the most popular container technology, but there many other container solutions out there:

- LXC
- LXD
- Solaris Zones
- RKT
- BSD Jails

Are There NonLinux-based Container Solutions?

Yes. Microsoft provides two type of container solutions Windows Server Containers and Hyper-V containers. The main difference between two that Windows Server Containers, just like Docker, share the kernel with the container host and the other containers while Hyper-V Containers do not.

Container Orchestration



Screenshot of kubernetes.io

Container orchestration platforms empower users to easily deploy, manage, and scale multi-container based applications in large clusters without having to worry about which server will host a particular container. Container cluster orchestration is also a very competitive space. Some of the most popular vendors are:

- Kubernetes
- Docker Swarm
- Amazon ECS

- Azure Container Service
- Marathon
- CoreOS Fleet
- Open Stack Magnum
- Diego
- Hashicorp Nomad

Summary

Container-based virtualization is a disruptive technology that is being adopted at a remarkable pace by small and medium size businesses. Enterprises show great interest as well but they are approaching the adoption in production environment very carefully. Google Search, Google App Engine, and Twitter are successful examples of using containers on a grand scale. At flow.ci we also use Docker containers, which help us scale easily while staying lean.

Virtual machines are still considered as a more mature technology with a higher level of security and many teams are more used to working with them. Virtual machines are generally more suitable for monolithic applications and for scenarios where security concerns are outweighing the needs for a lightweight solution.

Container-based virtualization is a much better solution for microservices architectural style, where features of the application are divided into small well-defined distinctive services. Containers and VMs are not excluding each other, they can be viewed as complementary solutions. Excellent example for this is the Netflix cloud where containers are running inside virtual machines.

. . .

flow.ci is a hosted continuous integration and delivery service, designed for teams who need a flexible and scalable solution but prefer not to maintain their own infrastructure. In flow.ci, development pipelines or automation workflows are simply called flows. In a flow, every step is a plugin that can be added by two clicks. You can add as many steps to your flow as you need, and there is no time limit on builds.