Virtual Machine (VM)

VM은 하드웨어 수준에서 격리된 가상 환경을 제공함. 이는 실제 컴퓨터의 하드웨어 자원을 가상화 하여 여러 개의 독립된 운영체제를 실행할 수 있게 해줌.

구성 요소

호스트 머신(Host Machine): VM을 실행하는 실제 물리적 컴퓨터.

Hypervisor: VM을 관리하는 소프트웨어 레이어. Hypervisor는 물리적 자원을 가상 자원으로 변환하여 각 VM에 할당함. 크게 두 종류

Type 1 Hypervisor (Bare-metal): 하드웨어 위에서 직접 실행되며, VM을 구동.

VMware ESXi, Microsoft Hyper-V, Xen 등

Type 2 Hypervisor (Hosted): 기존의 운영체제 위에서 실행되며, 그 위에 VM을 구동함.

VMware Workstation, Oracle VirtualBox 등

가상 머신: Hypervisor 위에서 실행되는 가상화 된 환경. VM은 CPU, 메모리, 디스크 등의 자원을 할당 받아 독립적으로 운영체제와 애플리케이션을 구동함.

작동 원리

Hypervisor는 물리적 하드웨어의 자원을 추상화하여 가상 자원으로 변환함.

이 가상 자원은 각 VM에 독립적으로 할당되며, VM은 실제 하드웨어에 직접 접근하는 것처럼 동작함.

각 VM은 독립된 운영체제를 설치하고 실행할 수 있음. 예를 들어, 하나의 물리적 서버에서 Windows, Linux, macOS 등의 다양한 운영체제를 동시에 실행할 수 있음.

Hypervisor는 VM 간의 자원 사용을 관리하고, 격리를 보장함. 이를 통해 각 VM은 다른 VM에 영향을 주지 않고 독립적으로 동작할 수 있음.

장점

격리성: 각 VM은 완전히 독립된 환경에서 실행되므로 보안과 안정성이 높음.

다양한 운영체제 지원: 하나의 물리적 서버에서 다양한 운영체제를 동시에 실행할수 있음.

기존 인프라와의 호환성: 전통적인 서버 환경과 유사한 방식이라 기존 인프라와 쉽게 통합 가능.

단점

자원 소모: 운영체제를 포함하여 전체 시스템을 가상화하므로 각 VM이 상당한 자원을 소모함.

부팅 시간: 운영체제를 포함하고 있어 부팅 시간이 길고 무거움.

복잡서: VM환경을 관리하고 유지하는 데 있어 높은 수준의 기술적 지식이 필요함.

컨테이너(Container)

컨테이너는 운영체제 수준에서 애플리케이션을 격리된 환경에서 실행할 수 있게 해주는 기술이다.

컨테이너는 애플리케이션과 그 종속성을 패키징하여, 어디서든 일관된 방식으로 실행될 수 있게함.

구성 요소

호스트 운영체제: 컨테이너가 실행되는 물리적 또는 가상 서버의 운영체제.

컨테이너 엔진: 컨테이너를 관리하고 실행하는 소프트웨어. 대표적으로 Docker가 있음.

컨테이너 이미지: 컨테이너 실행에 필요한 애플리케이션, 라이브러리, 설정 파일 등을 포함한 패키지.

컨테이너 이미지는 불변하며, 이를 기반으로 컨테이너가 생성됨

컨테이너: 실행중인 컨테이너 이미지. 컨테이너는 격리된 프로세스, 네트워크, 파일 시스템을 갖고있음.

작동 원리

컨테이너는 호스트 운영체제의 커널을 공유하면서, 격리된 환경에서 애플리케이션을 실행함.

이는 리눅스의 네임스페이스와 croups라는 커널 기능을 사용하여 이루어짐.

네임스페이스: 프로세스, 네트워크, 사용자, 파일 시스템 등 다양한 시스템 자원을 격리.

Cgroups: CPU, 메모리, 디스크 I/O 등의 자원 사용을 제한하고 관리.

컨테이너 엔진은 컨테이너 이미지를 바탕으로 컨테이너를 생성하고, 실행함. 각 컨테이너는 독립적인 환경에서 애플리케이션을 실행하며, 필요에 따라 빠르게 시작하고 종료할 수 있음.

장점

경량성: 운영체제를 포함하지 않고, 애플리케이션과 그 종속성만을 포함하므로 자원 소모가 적음.

이식성: 컨테이너는 운영체제 커널을 공유하기 때문에, 어느 환경에서든 일관된 방식으로 실행 가능.

확장성: 쉽게복제되고, 빠르게 스케일링할 수 있음. 따라서 클라우드 환경에서 자동화된 배포/확장 용이

DevOps 친화적: CI/CD 파이프라인에서 중요한 역할. 개발, 테스트, 배포과정을 자동화/일관된환경 유지

단점

커널 공유: 커널에 문제가 발생하면 모든 컨테이너가 영향을 받을 수 있음.

운영체제 호환성 제한: 컨테이너는 동일한 OS 커널을 공유하므로, 다른 운영체제를 실행할 수 없음.

보안: 컨테이너의 경량성으로 인해, 격리 수준이 VM보다 낮을 수 있음. 특히 다중 테넌트 환경에서

VM과 컨테이너의 주요 차이점

격리 수준	VM: 하드웨어 수준에서 완전한 격리, 자체 운영체제. 컨테이너: OS 수준에서 격리, 호스트의 커널 공유.
자원 사용 효율	VM: 더 많은 자원을 필요, 각 VM마다 운영체제가 구동되어 오버헤드가 큼. 컨테이너: 경량화되어 더 적은 자원을 사용, 더 빠르게 실행.
부팅 및 시작 시간	VM: 운영체제를 포함하고 있어 부팅 시간이 김. 컨테이너: 애플리케이션과 종속성만 포함하기 때문에 매우 빠르게 시작 가능.

사용 사례

VM: 멀티 OS환경이 필요하거나 높은 수준의 격리와 보안이 요구되는 상황에서 사용됨.

컨테이너: 애플리케이션의 빠른 배포, 확장, 마이크로서비스 아키텍처와 같은 상황에서 주로 사용됨.