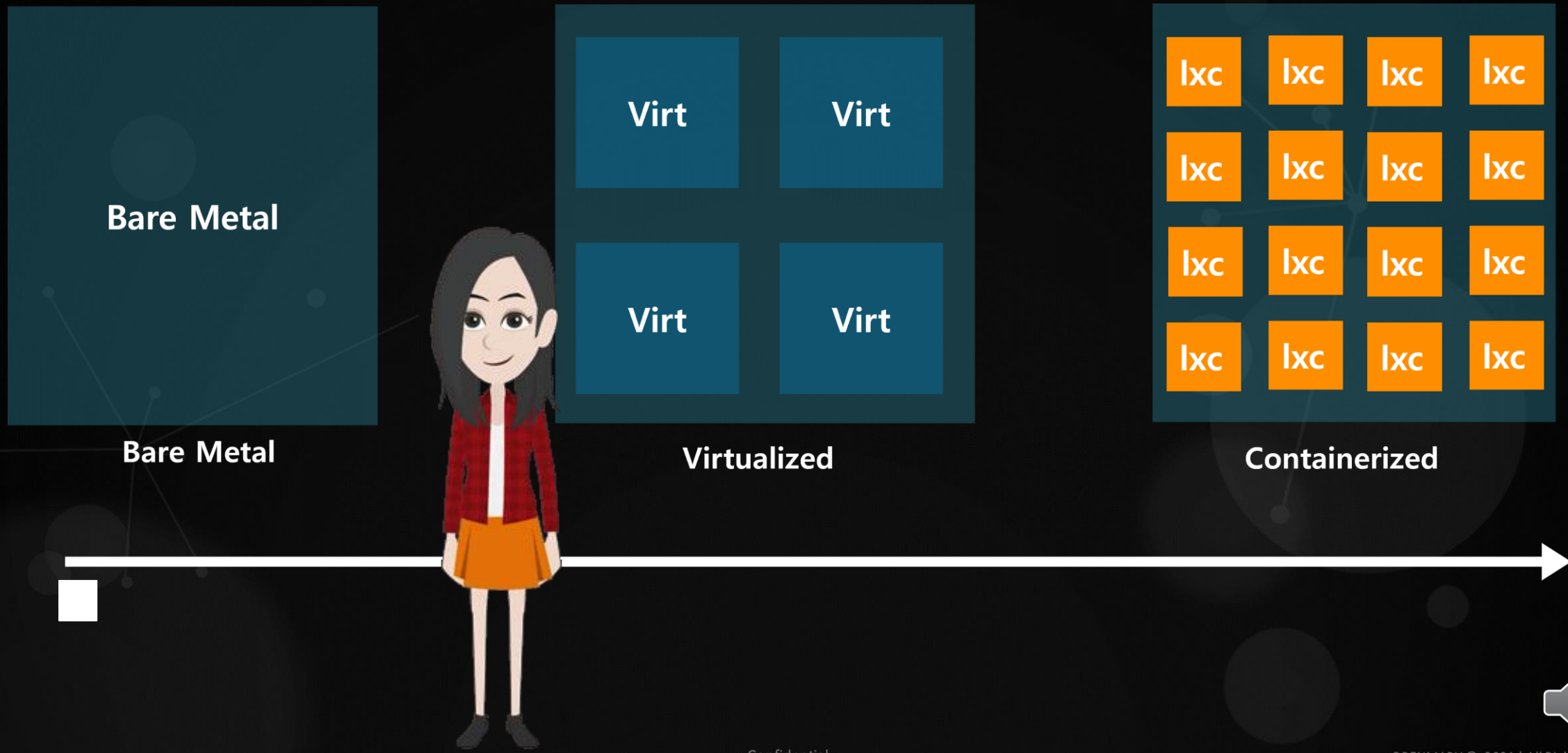
가상화 기술과 컨테이너 기술의 차이점

Container vs. Virtualization

Evolution of Infrastructure Architectures



openmaru

가상 머신

1개 애플리케이션을
위한 유휴 자원

VM

VM

VM

VM

게스트OS

컨테이너

여유 자원

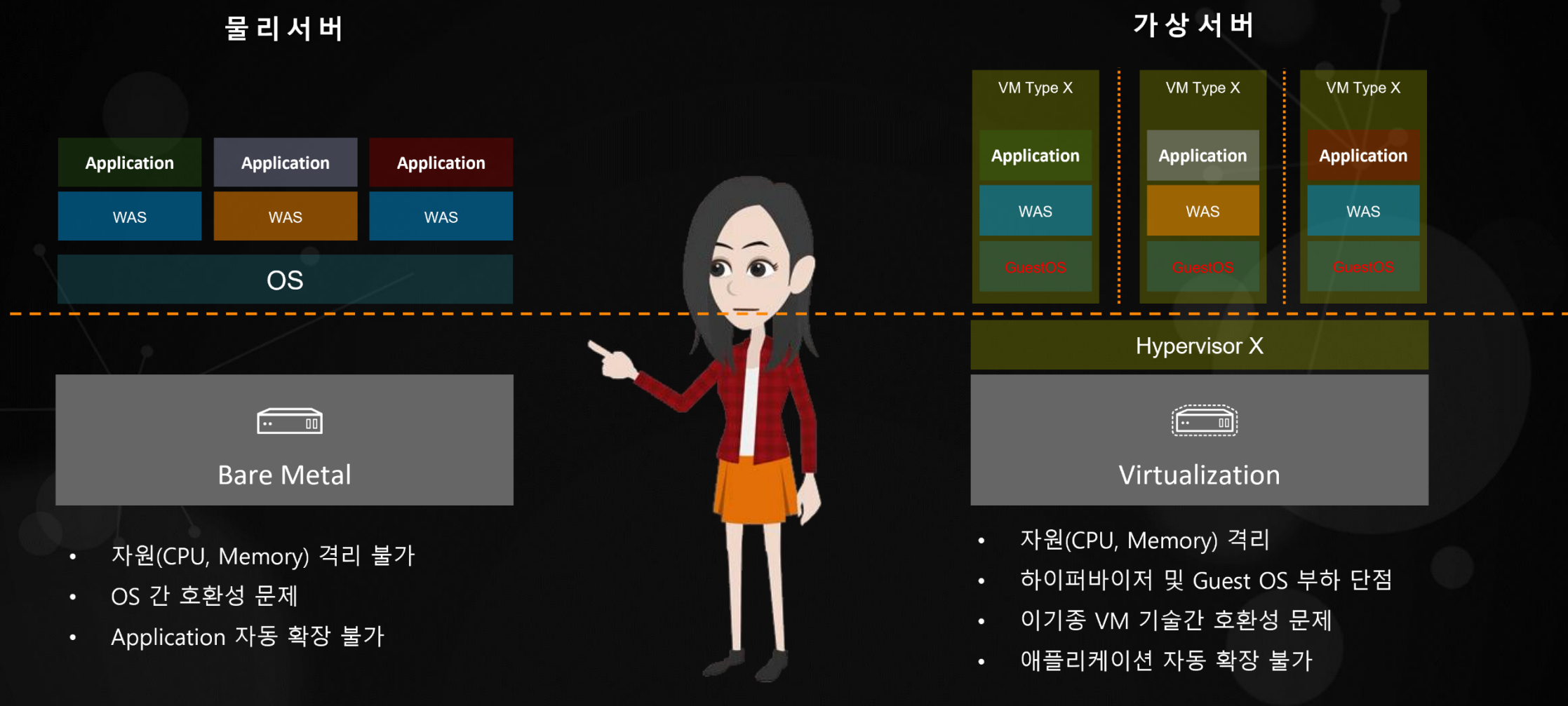
C C C

C C C C

동일한 H/W

물리서버와 가상서버 비교 - 물리서버

- 자원 효율성, 자원 격리, 호환성, Auto Scaling, DevOps, MSA, 관리 편의성에서 비교

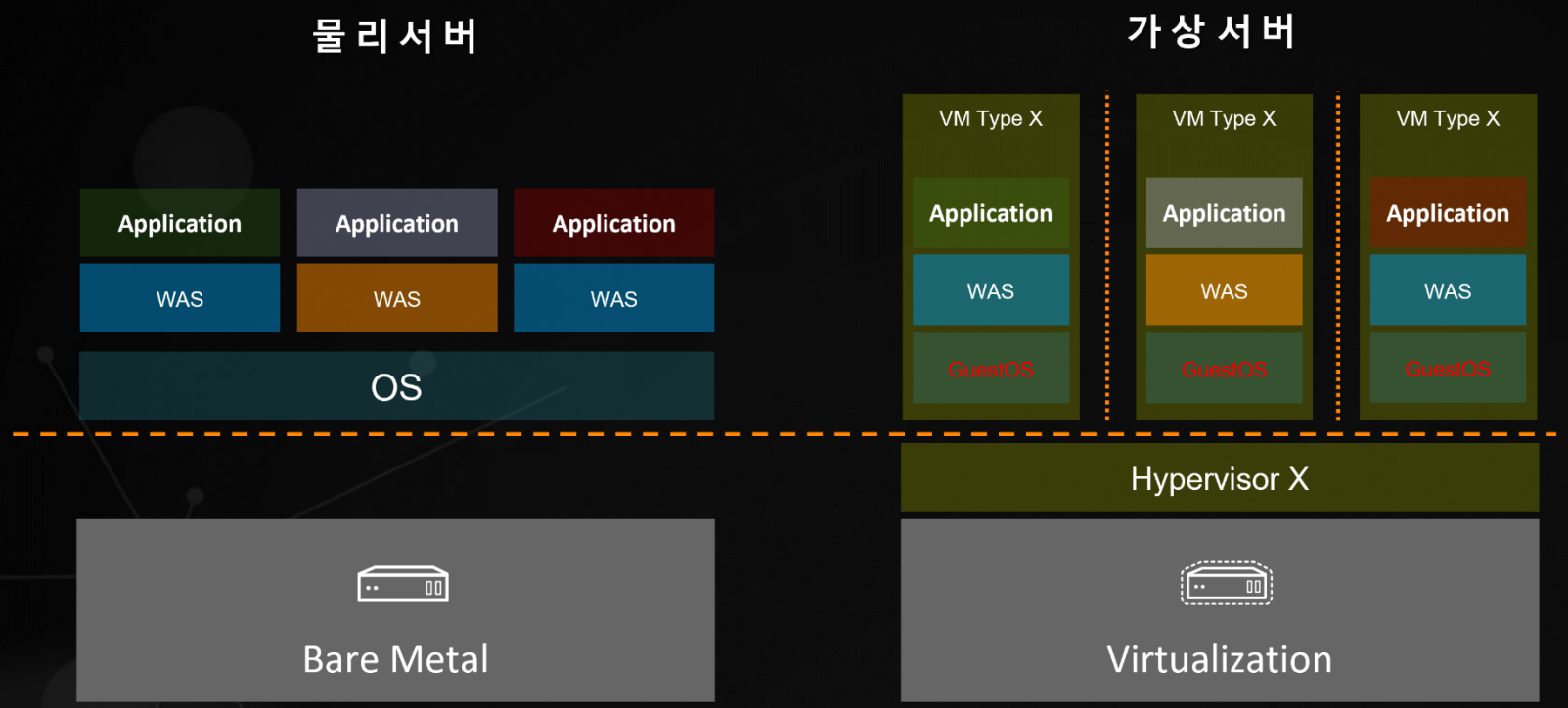


- 자원(CPU, Memory) 격리 불가
- OS 간 호환성 문제
- Application 자동 확장 불가

- 자원(CPU, Memory) 격리
- 하이퍼바이저 및 Guest OS 부하 단점
- 이기종 VM 기술간 호환성 문제
- 애플리케이션 자동 확장 불가

물리서버와 가상서버 비교 - 가상서버

- 자원 효율성, 자원 격리, 호환성, Auto Scaling, DevOps, MSA, 관리 편의성에서 비교

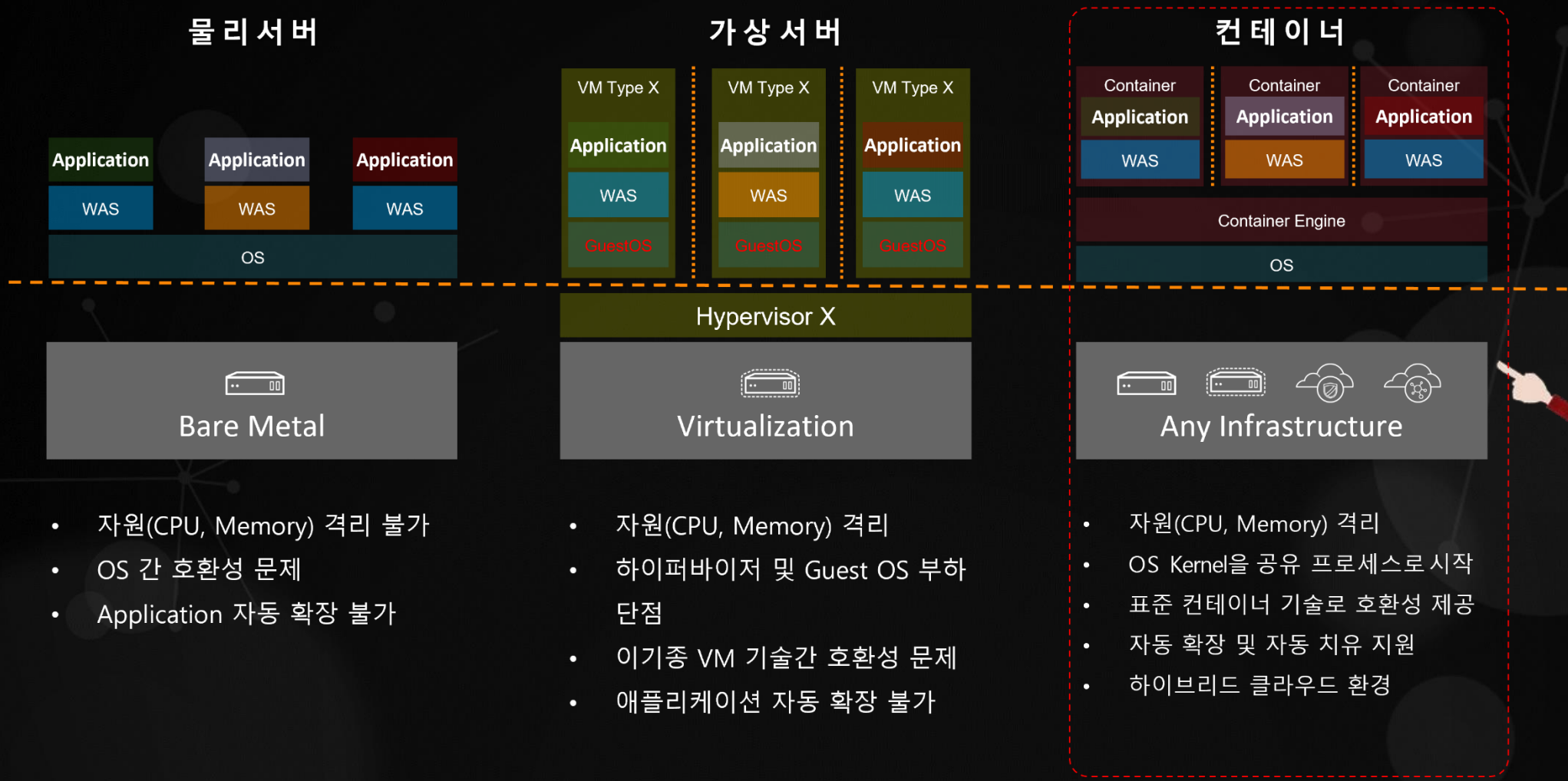


- 자원(CPU, Memory) 격리 불가
- OS 간 호환성 문제
- Application 자동 확장 불가

- 자원(CPU, Memory) 격리
- 하이퍼바이저 및 Guest OS 부하 단점
- 이기종 VM 기술간 호환성 문제
- 애플리케이션 자동 확장 불가

물리서버 vs 가상서버 vs 컨테이너 비교

- 자원 효율성, 자원 격리, 호환성, Auto Scaling, DevOps, MSA, 관리 편의성에서 비교



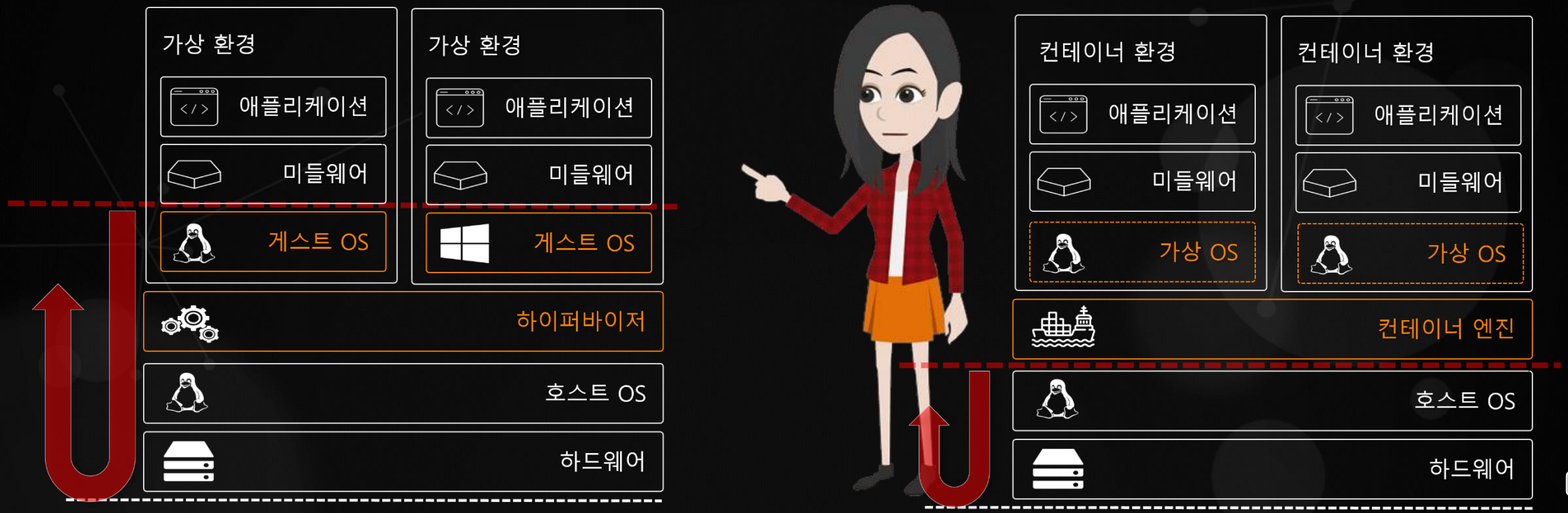
- 자원(CPU, Memory) 격리 불가
- OS 간 호환성 문제
- Application 자동 확장 불가

- 자원(CPU, Memory) 격리
- 하이퍼바이저 및 Guest OS 부하 단점
- 이기종 VM 기술간 호환성 문제
- 애플리케이션 자동 확장 불가

- 자원(CPU, Memory) 격리
- OS Kernel을 공유 프로세스로 시작
- 표준 컨테이너 기술로 호환성 제공
- 자동 확장 및 자동 치유 지원
- 하이브리드 클라우드 환경

시작 시간 - Containers vs. VMs

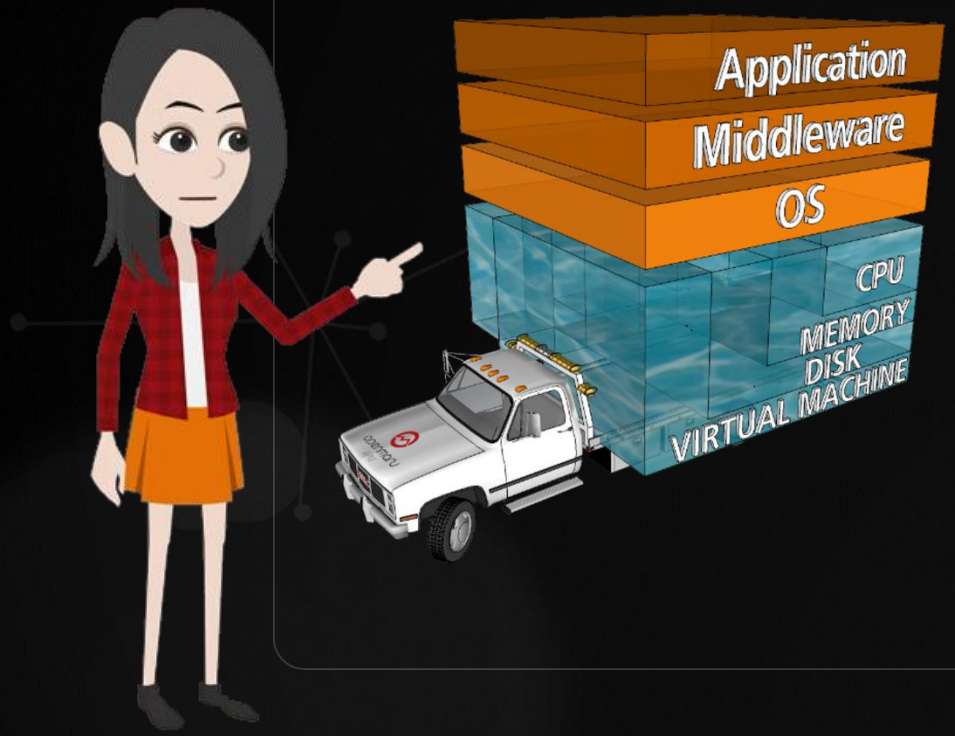
- 하드웨어 가상화는 CPU, 메모리, 하드 디스크 등의 하드웨어를 가상화하고 있기 때문에 하드웨어와 OS 부팅이 필요 (부팅에 분 단위 시간 소요)
- 컨테이너 형 가상화에서는 컨테이너 부팅 시 OS가 이미 시작된 상태 (프로세스 시작에 필요한 초 단위 시간 소요)



가상서버의 고질적인 문제점

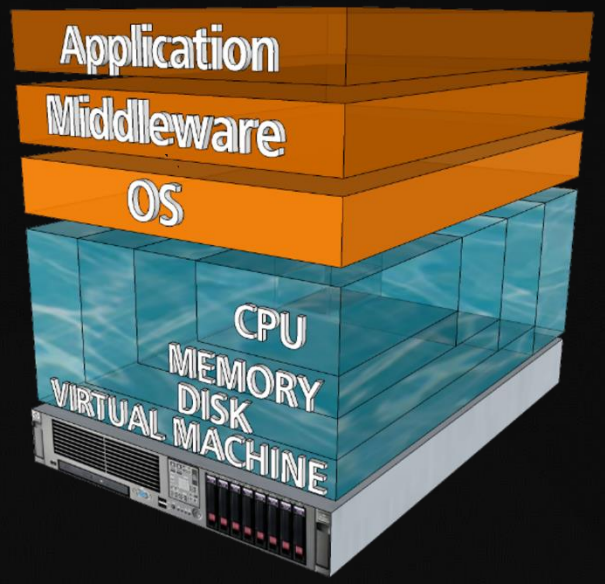
거대한 이미지 사이즈

- VM 을 템플릿 관리는 하지만 사이즈가 커서 재사용성을 높이기 어려움



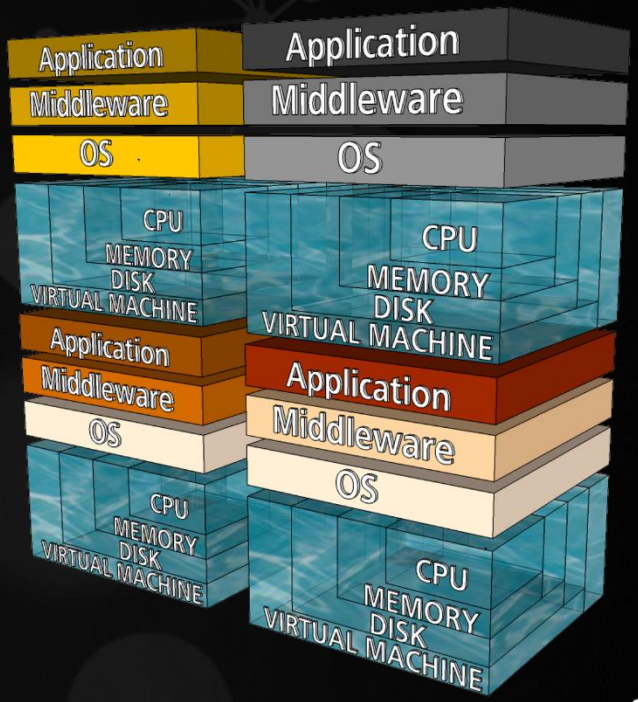
느린 시작 시간

- 부팅 시 Hypervisor – OS- 미들웨어 – 애플리케이션까지 실행 되어야함



VM 간의 환경 불일치

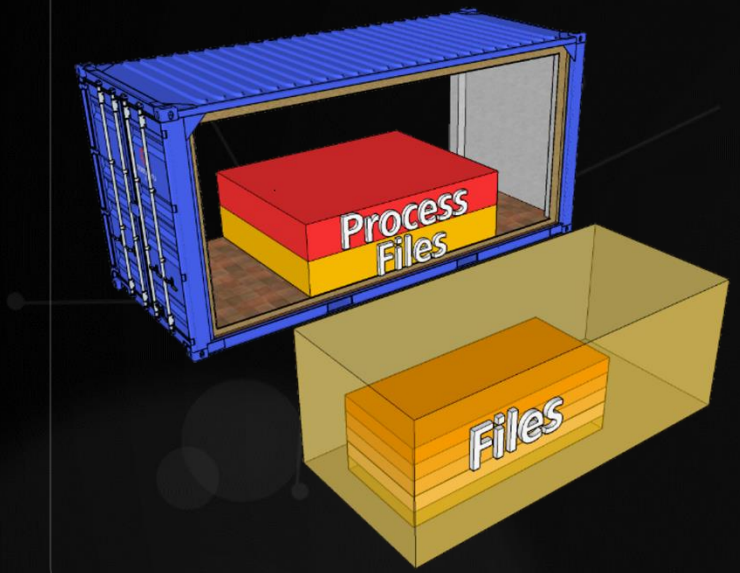
- VM 생성 후 개별로 변경 사항을 관리하기 때문에 VM 간 구성이나 환경이 불일치



컨테이너를 통한 가상서버 문제점 해결

작은 이미지 사이즈

- 컨테이너는 레이어 개념으로 이미지에 파일을 추가/삭제하여 관리함
- 레이어 사이즈를 최적화하여 이미지 사이즈를 최소화



빠른 시작 시간

- 컨테이너는 분리된 프로세스 형식으로 OS 부팅이 필요 없기 때문에 부팅 시간을 최소화 할 수 있음

Container = Process

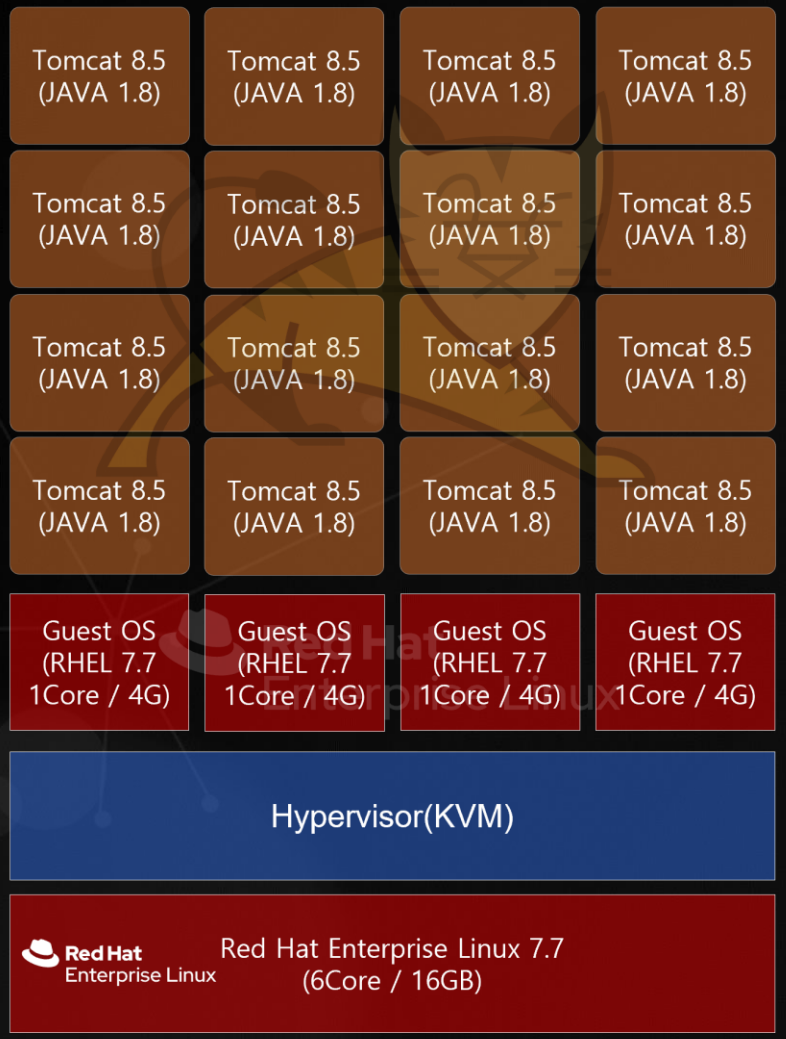
높은 이동성(Portability)

- 애플리케이션에 필요한 라이브러리나 의존 파일들을 이미지에 포함하기 때문에 환경에 의한 발행되는 문제가 거의 없음

가상화와 컨테이너 집적도 비교

openmaru

가상화 - 16개 Tomcat 인스턴스



컨테이너 - 40개 Tomcat 인스턴스