한국컴퓨터정보학회 하계학술대회 논문집 제31권 제2호 (2023. 7)

컨테이너 오케스트레이션 비교 및 분석

오지훈O
O제주대학교 컴퓨터공학과
e-mail: thrp4170@naver.comO

Container Orchestration Comparison and Analysis

Ji-hun OhO
ODept. of Computer Engineering, Jeju University

● 요 약 ●

본 논문에서는 컨테이너 오케스트레이션 플랫폼에 대하여 분석하고자 한다. 공공 클라우드 전환 로드맵 검토에 따라 클라우드 네이티브 전환을 위한 기술로 컨테이너, 마이크로서비스, 컨테이너 오케스트레이션의 중요성이 강조되고 있다. 대표적인 컨테이너 오케스트레이션 도구인 Kubernetes, Docker Swarm, Mesos를 비교하며, 이들의 초기 설치 용이성, 볼륨 관리, 애플리케이션 배포, 장애 관리 등에 대해 분석하고, 이를 통해 각 도구의 장단점과 적용 상황에 따른 고려사항을 파악함으로써, 클라우드 네이티브 전환 로드맵 수립에 도움을 제공하고자 한다.

키워드: Cloud Native, Migration, Container Orchestration

I. 서론

공공 클라우드 전환 로드맵 전면 재검토에 따라 2023년 5월 3일 행정안전부 주관하에 '클라우드 네이티브 전환 로드맵 수립 설명회'가 진행되었다. 2024년부터 2030년까지 7개년 동안 범정부 정보자원 등록·관리시스템에 등록된 모든 시스템의 클라우드 네이티브 전환이 목표이며, 이에 따라 부처별로 클라우드 네이티브 전환 로드맵을 수립, 자체 추진하도록 안내했다[1]. 클라우드 네이티브 전환이 본격화 됨에 따라, 마이크로 서비스 아키텍처(MSA, MicroServiceArchitecture), 컨테이너(Container) 기술 등과 같은 클라우드 기술의 중요성이 높아지고, 그중 컨테이너 기술로 사실상 표준화된 도커를 효율적으로 관리하기 위한 컨테이너 오케스트레이션(Container Orchestration) 또한 중요성이 높아지고 있다. 컨테이너 오케스트레이션은 컨테이너의 배포, 관리, 확장, 네트워킹을 자동화하는 기능을 가지고 있으며, 마이크로서비스에서 애플리케이션의 여러 부분들을 독립적으로 실행시키고 라이프사이클을 더욱 효과적으로 제어할 수 있다. 오케스트레이션을 통해 컨테이너 라이프사이클을 관리하면 CI/CD 워크플로우에 이를 통합하는 DevOps 팀을 지원할 수도 있다. 컨테이너화된 마이크로서비스는 애플리케이션 프로그래밍 인터페이스(API) 및 DevOps 팀과 함께 클라우드 네이티브 애플리케이션의 기반을 이룬다[2]. 본 논문에서는 대표적으로 사용되는 컨테이너 오케스트레이션인 Kubernetes, Docker Swarm, Mesos 도입 시 고려해야 할 점들에 대해 살펴보고자 한다.

II. 오케스트레이션 플랫폼 비교 및 분석

1. 초기 설치의 용이성과 운영 복잡성

Kubernetes는 기능이 많은 만큼 초기 설치 시에 여러 Resource 설정과 운영 시 난이도 존재한다. 이로인해, 사용자가 시스템을 이해하는데 많은 시간과 노력이 필요하다[3]. Mesos는 분산시스템을 위한 IDC 및 복잡한 대규모 작업 관리가 필요한 클라우드 환경에 특화되어 있고, Marathon과 같은 별도의 프레임워크를 운용해야 하기에, 난이도와 운영 복잡성이 높은 편이다. 반면에 Doker Swarm은 설치와 사용이 간단하며, Doker CLI를 기반으로 작동하기 때문에, 설정 및 운용 방법이 비교적 낮은 진입장벽을 가진다[5].

2. 볼륨 관리

Kubernetes에서는 Persistent Volume Claim과 Persistent Volume와 같은 Resource를 제공하여 복잡한 볼륨 관리를 지원한다. Mesos에서는 Third party를 통한 별도의 설정을 거쳐 외부 스토리지 서비스나 외부 볼륨 지원 등을 사용해야 한다[4]. Doker Swarm에서는 Doker의 볼륨 기능을 기본적으로 사용하게 되는데, 동적으로 볼륨을 조절할 수 있는 등, 추가적인 고급 기능은 제공되지 않는다.

3. 애플리케이션 배포

Doker Swarm은 무중단 배포 기법의 하나인 Rolling updates를 지원한다. Kubernetes는 Rolling updates뿐만 아니라 Blue-green deployments와 Canary deployments 등 다양한 배포 전략을 지원한다. Mesos에서는 추가적인 도구나 설정을 이용해 다양한 배포 전략을 구현할 수 있다.

4. 장애 관리

Docker Swarm은 서비스의 상태를 지속적으로 모니터링하고, 실패한 서비스를 자동으로 복구한다. 또한, 복제된 서비스를 여러 노드에 분산하여 단일 노드 장애가 전체 시스템에 영향을 미치지 않도록 한다. 그러나 Docker Swarm의 장애 복구 기능은 기본적인 레벨에 그치며, 복잡한 복구 시나리오에 대한 지원은 제한적이다. Kubernetes는 자체 회복 메커니즘을 통해 장애 복구를 지원한다. Pod가 실패하거나 삭제되면, Kubernetes는 ReplicaSet을 통해 자동으로 새 Pod를 생성하여 원하는 상태를 유지하며, Node가 다운되면, 해당 Node에서 실행되던 Pod들은 다른 Node에서 재스케줄링 된다. Mesos는 서드파티 프레임워크를 통해 장애 복구를 지원한다. 예시로 Third party Marathon은 애플리케이션의 인스턴스를 모니터링하고, 실패한 인스턴스를 자동으로 재시작하여 애플리케이션의 가용성을 유지한다[6]. 또한, Mesos는 분산 시스템의 복잡성을 추상화하여, 애플리케이션을 여러 노드에 분산시키고 장애 시에도 작업을 계속 진행할 수 있도록 지원한다. 그러나 Mesos의 장애 복구 기능은 프레임워크에 따라 달라지며, 복잡한 복구 시나리오를 다루기 위해서는 추가적인 설정과 관리가 필요하다.

III. 결론

세 플랫폼 모두 장점과 단점이 있지만, 각 플랫폼의 선택은 특정 운영환경, 사용자의 기술적 역량 등 다양한 요소를 고려해야 한다. Kubernetes는 복잡하지만, 가장 다양한 기능을 제공하며, Kubernetes를 활용하면 복잡한 클러스터 환경을 더욱 효과적으로 관리하고 확장할 수 있다. Docker Swarm은 간단한 인터페이스와 설치 과정을 제공하지만, Kubernetes나 Mesos에 비해 제공하는 기능이 제한적이다. Mesos는 대규모 분산 시스템에 적합하며, Marathon과 같은 별도의 Third party 프레임워크를 사용하여 커스텀 기능을 구현할 수 있다.

따라서, 컨테이너 오케스트레이션 플랫폼은 단순히 기술적인 선택이 아니라, 다양한 요소를 고려해서 전략적으로 선택이 되어야 한다. 오케스트레이션 플랫폼은 클라우드 네이티브로 전환하는 데 필요한 기능, 성능, 안정성을 제공하는 도구이기 때문이다. 알맞은 오케스트레이션의 선택은 공공 클라우드 전환 로드맵을 수립하는 데 중요한 기준이 될 수 있을 것이다.

REFERENCES

[1] https://www.etnews.com/20230503000233
[2] https://www.redhat.com/ko/topics/containers/what-is-container-orchestration
[3] https://kubernetes.io/
[4] https://mesos.apache.org/
[5] https://docs.docker.com/engine/swarm/
[6] https://mesosphere.github.io/marathon/