문서뷰어의 모놀리식에서 쿠버네티스 기반 MSA 전환과 그 효과

이철순*, 김양중**
한국공학대학교 소프트웨어 융합공학과
{*samwalto, **zeroplus}@tukorea.ac.kr

Transitioning from Monolithic to Kubernetes-based MSA in Document Viewer and Its Impact

Chulsoon Lee, Yangjung Kim*
Tech University of Korea

요 약

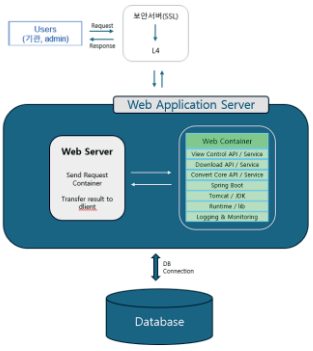
문서뷰어 시스템을 기존 모놀리식 아키텍처에서 쿠버네티스 기반 MSA 로 전환하여 확장성과 유연성을 확보하고 자동화된 배포 및 장애 격리를 가능하게 하였다. 이를 위해 서비스 단위를 기능별로 분리하고 HPA, Istio, Jenkins 를 활용한 인프라를 구축하여 운영 효율성을 높였다. 그러나 서비스 증가로 운영 복잡성, 문서 변환과 이미지 스트리밍 성능 개선의 한계, 서비스 종속성 문제가 나타났다. 향후 서버리스 아키텍처와 AI 기반 자동화를 도입하여 운영 효율성을 더욱 개선할 계획이다.

Ⅰ. 서 론

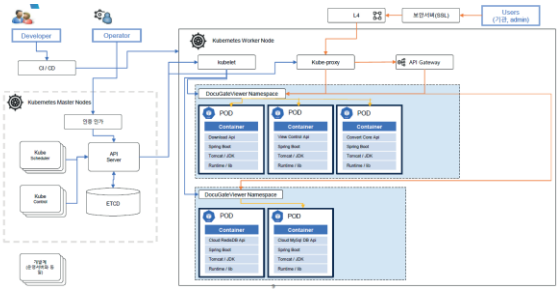
기존 문서뷰어 시스템은 모놀리식 아키텍처로 개발되어 확장성 부족, 장애 전파, 배포 지연 등의 문제를 겪었다. 단일 데이터베이스와 결합된 구조는 신규 기능 추가 시 전체 시스템의 테스트와 배포를 필요로 하며, 트래픽 폭주 시 특정 모듈의 장애가 전체 서비스 중단으로 이어졌다. 본 논문은 이러한 문제를 해결하기 위해 쿠버네티스[1] 기반 MSA[2]로의 전환을 수행한 과정과 그 효과를 분석한다.

Ⅱ. 본 론

1. 전환전략 및 구현



[그림 1]뷰어 기존구조, 모놀리식



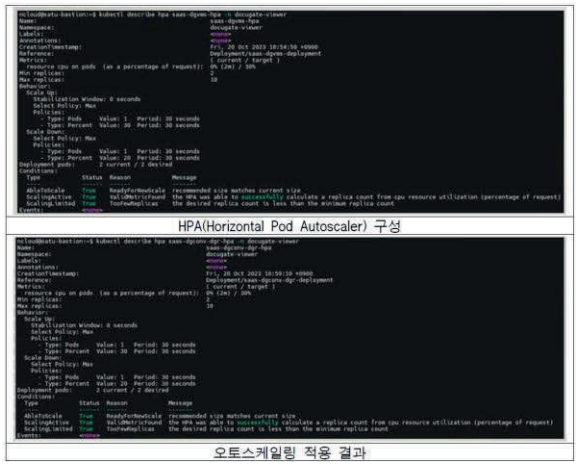
[그림 2]MSA 를 적용한 뷰어 구조

[그림 1] 모놀리식 구조 → [그림 2] 쿠버네티스 기반 MSA 구조로의 전환이 시도되었다. 이러한 변화를 통해 높은 확장성과 유연성을 가지게 되고 장애 격리가 가능해짐을 알 수 있다. 또한 DevOps 및 자동화에 최적화되어 개발 및 배포가 효율적으로 이루어질 수 있게 되었다.

1.1 아키텍처 분해전략

뷰어의 서비스 단위를 원본파일 다운로드, 문서변환, 뷰어, 이미지 스트리밍, 사용자 권한 관리 등 기능별로 분리하였다.

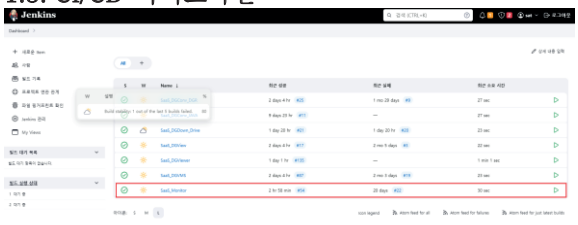
1.2. 쿠버네티스 기반 인프라 구축



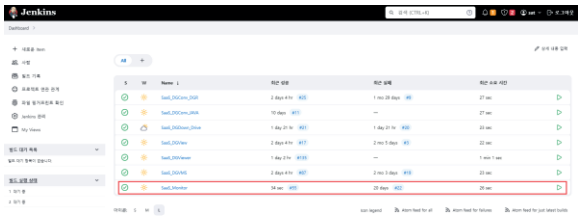
[그림 3]HPA 설정

HPA(Horizontal Pod Autoscaler) 설정을 통해 CPU 사용률 30% 초과 시 자동으로 파드 수를 2→10 개로 확장해 트래픽 폭주에 대응하였고 서비스 메시(Istio) 에서 Istio Ingress Gateway 를 활용하여 외부 트래픽(예: 웹 브라우저, API 요청)을 관리하고 TLS 암호화를 적용하였다.

1.3. CI/CD 파이프라인



[그림 4]Jenkins 배포 전



[그림 5]Jenkins 배포 후

Jenkins 를 활용하여 클라우드 서비스 배포, Kubernetes 의 Pod 배포, 그리고 롤백까지 자동화하였다.

1.4. 데이터 관리

서비스별 데이터 격리, API 기반 데이터 교환, 이벤트 기반 데이터 동기화, 분산 트랜잭션 관리 등에 집중하였으며 CQRS 패턴[3]을 참고하여 서비스간 데이터 변경에 대응하였다.

Cloud for MySQL 을 이용해 변환요청일시, 변환성공여부, 문서유형 등 문서이력정보를 관리하고 원본문서, 변환결과물을 blob 으로 저장하였다.

2. 전환 효과

2.1. 운영 효율성 향상

각각의 서비스에 대해 1 분 이내(23sec ~ 61sec) 자동 배포 자동 롤백이 가능하게 되었다. 각 서비스별 별도 대응이 가능해짐에 따라 장애에 좀 더 효율적으로 대응할 수 있게 되었고 장애 대응을 위한 개발시간 또한 단축할 수 있을 것이라 기대된다.

2.2. 확장성 확보

모놀리식 서비스에 비해 동시사용자 확장이 더 용이해졌다. 문서뷰어의 특성 상 문서 포맷의 종류, 문서 용량, 문서에 포함된 object, page 에 따라 리소스 사용량이 달라지긴 하지만 HPA 와 클라우드 자원 동적할당을 통해 좀 더 유연하게 대처할 수 있게 되었다.

3. 도전 과제 및 해결 방안

3.1. 운영 복잡성 증가

서비스가 많아질수록 모니터링, 로깅, 배포 등 운영 관리가 복잡해져, 적절한 인프라와 도구가 갖춰지지 않으면 오히려 성능 저하로 이어질 수 있고 비용 또한 증가할 수 있다는 점에서 주의가 필요하다.

3.2. 종속성 문제

MSA 구조에서 각 서비스는 독립적으로 운영되고 특정 서비스의 장애가 다른 서비스에 영향을 미치지 않아야 한다. 그러나 실시간으로 문서변환이 이루어지는 문서뷰어의 특성 상 이러한 장애 격리에 어려움이 있다. 예를 들어 원본문서 다운로드에 실패한 경우 문서변환이 이루어질 수 없다.

서비스는 물리적으로 구분되어 있지만 종속성으로 인한 문제가 발생한다.

이를 해결하기 위해 회복탄력성(Resilience) 패턴[4], 비동기 처리 아키텍처[5]와 같은 대응 전략에 대한 구현 및 테스트가 필요하고 뷰어 자체적인 정책(단순 실패처리 할 것인지, 일정시간 장애대응 대기 후 처리할 것인지 등)을 마련할 필요가 있다.

3.3. 성능향상

MSA 를 적용하면서 문서변환, 이미지 스트리밍 등의 속도개선을 기대하였으나 효과적이지 않은 것으로 분석되었다. MSA 는 작은크기의 빈번한요청에 최적화되어 있으나 대용량 이미지 또는 파일에 대해서는 전송지연이 발생할 수 있다.

위와 같은 사항을 해결하기 위해 하이브리드 아키텍처(이미지 스트리밍은 모놀리식 유지, 나머지는 MSA 적용)와 같은 사항들이 논의되었으나 개발기간 내에 적용하는 것에 어려움이 있었다.

3.4. 초기 전환 비용

개발기간(6 개월) 동안 개발 리소스 3 배 이상 증가하였다. 이를 해결하기 위해서는 단계적 전환 로드맵 수립 및 DevOps 팀 협업 강화가 필수적이라고 생각된다.

초기 투입금액이 모놀리식에 비해 크다는 점도 고려되어야 한다. 리소스 자동 증감 등이 구현되기 위해서는 클라우드 기반 서비스를 사용하거나 그에 준하는 하드웨어가 준비되어야 하기 때문이다.

Ⅲ. 결 론

문서뷰어의 MSA 전환은 확장성·성능·유연성 측면에서 뚜렷한 성과를 거두었으나, 운영 복잡성 관리와 조직 문화 변화가 성공의 핵심 요소였다. 향후 서버리스 아키텍처, 하이브리드 아키텍처 도입과 AI 기반 자동화를 통해 운영 효율성을 더욱 개선할 계획이다.

ACKNOWLEDGMENT

본 연구는 고용노동부 및 한국산업인력공단 “2025 년 고숙련 마이스터 사업”과 (주)에스에이티정보(www.satu.co.kr)의 지원을 받음

참 고 문 헌

[1]https://kubernetes.io/ko/docs/concepts/overview/

[2]https://learn.microsoft.com/ko-kr/azure/architecture/guide/architecture-styles/microservices

[3]https://docs.aws.amazon.com/ko_kr/prescriptive-guidance/latest/modernization-data-persistence/cqrs-pattern.html

[4]https://joylucky7.tistory.com/50#google_vignette

[5]https://f-lab.kr/insight/understanding-async-queues-and-event-driven-architecture