



# Kubernetes 플랫폼 옵저버빌리티 모범 사례 가이드

플랫폼 엔지니어링 시 대규모 Kubernetes 모니터링,  
보안, 최적화를 간소화하는 방법

# 서론

조직들이 기술 스택을 지속적으로 현대화해나가면서 Kubernetes는 컨테이너화된 환경을 대규모로 관리하기 위한 사실상의 표준으로 부상했습니다. 그 결과, 많은 조직들이 내부 개발 플랫폼(IDP)을 구축하는 데 플랫폼으로 활용하고 있기도 합니다.

모든 조직 중 **3분의 2(66%)**가 기업의 운영 현대화를 지원할 효과적인 솔루션으로서 Kubernetes를 프로덕션 환경에 도입했습니다.

- [CNCF 2023 연례 설문조사](#)

하지만 복잡하고 동적이며 일시적인 Kubernetes의 특성으로 인해, 애플리케이션과 멀티클라우드 인프라의 옵저버빌리티를 확보하기가 특히 어렵습니다. 복잡한 Kubernetes 환경에서 하점을 모니터링하면 팀의 생산성과 협업을 크게 저해할 수 있다는 점에서, 엔드 투 엔드 옵저버빌리티는 여러 부문으로 이루어진 팀을 조율하고 유용한 인사이트를 얻는 데 열쇠가 되고 있습니다.

전체 기술 스택과 수많은 구성 요소에 대한 전반적인 인벤토리를 수집하기는 쉽지 않습니다. 하지만 이러한 구성 요소가 최적의 상태로 안전하게 작동하는지 여부를 판단하는 것은 완전히 차원이 다른 문제입니다.



**76%**의 기술 리더들은 Kubernetes의 동적 특성으로 인해 이 아키텍처에 대한 가시성을 유지하는 것이 기존 기술 스택에 비해 더 어렵다고 답했습니다.

- [Dynatrace 2024년 옵저버빌리티 현황 보고서](#)

# Kubernetes의 과제

Kubernetes를 대규모로 구현하려고 할 때 플랫폼 엔지니어링 및 개발 팀은 다음과 같은 과제에 직면하게 됩니다.

## 1. 복잡성과 클러스터 불안정성

Kubernetes 플랫폼의 복잡성을 감안할 때, 다양한 원인으로 클러스터 불안정성과 보다 광범위한 애플리케이션 문제가 발생할 수 있습니다. 이 플랫폼의 복잡한 아키텍처와 수많은 구성 요소로 인해 구성하고 모니터링하기가 특히 어려울 수 있습니다.

## 2. 리소스 및 비용 관리

Kubernetes 환경은 효율적으로 운영되지 않을 경우 비용과 소요되는 리소스가 빠르게 증가할 수 있습니다. Kubernetes 환경과 관련한 성능 저하와 추가 비용을 방지하려면 적절한 리소스 할당 기능과 확장 기능이 필요합니다.

## 3. 보안

Kubernetes 인프라를 다루는 데 있어 보안은 중요한 과제 중 하나입니다. 고도로 동적인 환경, 수동 구성, 일원화되지 않은 클러스터 소유권으로 인해 변수가 많아져 취약성이 나타날 가능성이 더 커집니다. 설계상 Kubernetes는 내장된 보안 기능이 부족하기 때문에, 잠재적으로 악의적인 행위자가 포드와 기타 Kubernetes 고유 구성 요소 간의 무제한 통신을 악용할 가능성이 있습니다.

## 4. 한정적인 옵저버빌리티

모니터링 및 옵저버빌리티에 대한 전통적인 접근 방식은 도구의 무질서한 증가를 초래하고 Kubernetes가 대규모로, 비효율적으로 사용되게 만들 수 있습니다. 이러한 접근 방식을 사용할 경우 매우 동적인 환경에서 모니터링에 상당한 허점이 발생할 수 있으며, 그 결과 Kubernetes의 상태와 성능에 대한 가시성이 부족해질 수 있습니다.

## 5. 엔지니어링 팀의 부담 가중

단순한 모놀리식 애플리케이션의 차원을 넘어 동적 애플리케이션 컨테이너화로 전환하면 엔지니어링 팀의 인지 부하가 증가하여 일시적으로 생산성이 저하될 수 있습니다. 플랫폼 엔지니어링은 이 같은 복잡성을 해결할 솔루션으로 등장했지만, 대규모 환경을 설계하고 관리하는 데 요구되는 Kubernetes 숙련도가 부족한 팀이 많습니다.

하지만 잘 구현된 Kubernetes 환경의 이점은 그 어떤 복잡성도 상쇄하고 남을 만큼 큽니다.

# Kubernetes를 사용해야 하는 이유

Kubernetes를 사용하면 단일 기본 플랫폼 내에서 컨테이너형 애플리케이션을 매우 유연하고 확장 가능한 방식으로 실행, 관리 및 조정할 수 있습니다. Kubernetes가 인기를 끌게 된 주된 이유 중 하나는 사용자가 선언적 구성을 사용하여 설정하는, 정상 상태를 유지해 주는 자가 복구 기능입니다.

아울러 Kubernetes는 사용자가 플랫폼의 경험과 기능을 고유한 니즈와 원하는 설정에 따라 조정할 수 있는 풍부한 확장성과 오픈 소스 호환성을 제공합니다. 여러 노드에 포드를 배포하여 고가용성을 보장함으로써 Kubernetes는 DevOps, 플랫폼 엔지니어링, 보안 방식 모두에서 운영 효율성과 민첩성을 제공하는 인기 플랫폼이 되었습니다.

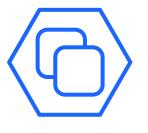
이러한 각 방식에서 Kubernetes의 이점을 얻을 수 있지만, 이 플랫폼에 대한 의존도는 각기 다릅니다. 그 차이점을 더 잘 이해하려면 먼저 이러한 방식들이 어떻게 상호 작용하고 최적의 Kubernetes 기능에 어떻게 좌우되는지 알아보면 도움이 됩니다.



**DevOps** 방식에서는 IT 인프라 및 애플리케이션의 계획, 개발, 테스트, 배포 및 전반적인 상태를 다룹니다.



**보안** 방식에서는 이러한 인프라 및 애플리케이션 내의 취약성, 위협 및 공격을 관리하고 완화합니다. 보안은 DevOps 프로세스 ("DevSecOps")에 기본적으로 포함되어 소프트웨어 개발 라이프사이클(SDLC) 초기에 취약성을 찾아내는 데 도움을 주기도 합니다.



**플랫폼 엔지니어링**은 개발, 운영 및 보안 팀의 운영 효율성을 높이기 위해 내부 개발 플랫폼(IDP)을 통해 표준화된 도구와 방식을 프로비저닝하고 유지 관리할 목적으로 DevOps 분야에서 등장한 기본 방식입니다. 이 셀프 서비스 기반 접근 방식은 개발자의 인지 부하를 줄여 개발자 경험(DevEx)을 개선합니다.

# 셀프 서비스 배포 모델

"플랫폼 구축용 플랫폼"인 Kubernetes는 셀프 서비스 배포 모델을 사용하여 대규모 개발 작업을 지원하면서 IDP의 기본 구성 요소 역할을 하는 경우가 많습니다. 셀프 서비스 모델은 골든 패스 템플릿, 수작업을 줄이기 위한 워크플로 자동화, 거버넌스와 규정 준수를 보장하는 디지털 가드레일로 구성됩니다. 팀은 셀프 서비스 템플릿, 애플리케이션 컨테이너화, 코드형 구성(CaC), 개발 프로세스를 간소화하고 강화하는 기타 기능 등의 주요 기능을 잘 갖춘 IDP를 사용하여 이러한 방식을 구현할 수 있습니다. 오른쪽 다이어그램은 가장 기본적인 수준에서 IDP에 어떤 기능이 포함되어야 하는지 간략하게 보여 줍니다.

이러한 특성을 가진 IDP를 구현하고 강력한 셀프 서비스 모델을 따름으로써, 팀은 생산성을 최적화하면서 긍정적이고 만족스러운 개발자 경험을 증진할 수 있습니다. Kubernetes를 IDP 스택의 중요한 구성 요소로 구현할 경우, 이러한 컨테이너화된 환경이 안전하고 정상적이며 최적화된 상태로 실행되어야 성능 저하를 방지할 수 있습니다. IDP가 기능하는 데 있어서 기반이 되는 Kubernetes의 역할을 고려할 때, 문제가 있거나 정상적이지 않은 클러스터는 여러 다른 구성 요소에 도미노 효과를 일으켜 SDLC의 모든 단계를 저해할 수 있습니다.

이 같은 고기능 Kubernetes 환경의 중요성을 감안하면, 설계에서부터 옵저버빌리티와 보안을 플랫폼에 내장하는 것이 무엇보다 중요합니다. 옵저버빌리티는 시스템에 부차적이거나 부가적인 기능으로서가 아니라, Kubernetes의 설계와 유지보수 차원에서 반영되어야 하는 기능입니다. 옵저버빌리티는 Kubernetes의 정상적인 작동, 최적화, 보안을 보장하기 위한 포괄적인 가시성과 지침을 제공할 수 있기 때문입니다.

Gartner®는 "2026년까지 대규모 소프트웨어 엔지니어링 조직의 80%가 재사용 가능한 서비스, 구성 요소 및 애플리케이션 딜리버리 도구의 내부 공급자로서 플랫폼 엔지니어링 팀을 설치할 것으로 전망하며, 이는 2022년의 45%에서 대폭 증가한 수치입니다."

— Gartner, 2024년 주요 전략 기술 동향: 플랫폼 엔지니어링, 2023년 10월. Gartner는 미국 및 기타 국가에서 Gartner 및/또는 해당 계열사의 등록 상표이자 서비스 마크이며, 허가를 받아 본 문서에 사용되었습니다. All rights reserved.

제품 및 애플리케이션 팀

## 플랫폼 인터페이스

- |          |                     |
|----------|---------------------|
| 설명서      | 그래픽 사용자 인터페이스(웹 포털) |
| 프로젝트 템플릿 | API 및 CLI           |

## 플랫폼 기능

- |                      |                          |
|----------------------|--------------------------|
| 환경 및 리소스 제공          | 사용자 및 서비스를 식별하고 권한 부여    |
| 인프라: 컴퓨터, 네트워크, 스토리지 | 아티팩트 스캔 및 정책 적용          |
| 데이터: 데이터베이스, 캐시, 버킷  | 구축, 테스트 및 딜리버리 자동화       |
| 메시징: 대기열, 브로커        | 보안 정보를 통해 서비스를 워크로드에 바인딩 |
|                      | 레지스트리 및 리포지토리에 아티팩트 저장   |
|                      | 워크로드 관찰                  |

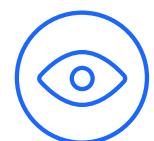
인프라 공급자  
플랫폼 기능 공급자

CNCF 2023 연례 설문조사를 바탕으로 작성

# 클라우드 네이티브 환경의 복잡성을 해결하기 위한 통합 옵저버빌리티 및 보안

개발 팀이 수많은 클라우드 문제를 해결하는 데 사용할 수 있는 도구는 무수히 많습니다. 하지만 옵저버빌리티에 대한 전체론적 접근 방식은 조직이 Kubernetes와 같은 클라우드 네이티브 기술이 제시하는 중요한 가치를 실현하는 데 도움이 될 수 있습니다.

통합 옵저버빌리티 플랫폼은 팀에 다음과 같은 이점을 제공합니다.



## 포괄적인 엔드 투 엔드 옵저버빌리티

기본적으로 분산되어 있는 Kubernetes와 무수한 관련 마이크로서비스의 특성으로 인한 복잡성을 극복하기 위해서는 Kubernetes 환경, 그 기반 위에 구축된 IDP, 플랫폼에 배포된 애플리케이션에 대한 포괄적인 엔드 투 엔드 옵저버빌리티를 반드시 확보해야 합니다. 포괄적인 옵저버빌리티를 통해 팀은 Kubernetes 환경의 모든 부분에서 데이터와 인사이트를 수집하여 모든 시스템 이벤트 또는 구성 요소가 빠짐없이 모니터링되도록 보장할 수 있습니다. 아울러 이 기능은 팀이 워크로드를 추적하고 문제의 근본 원인을 신속하게 파악하는 데에도 유용합니다.



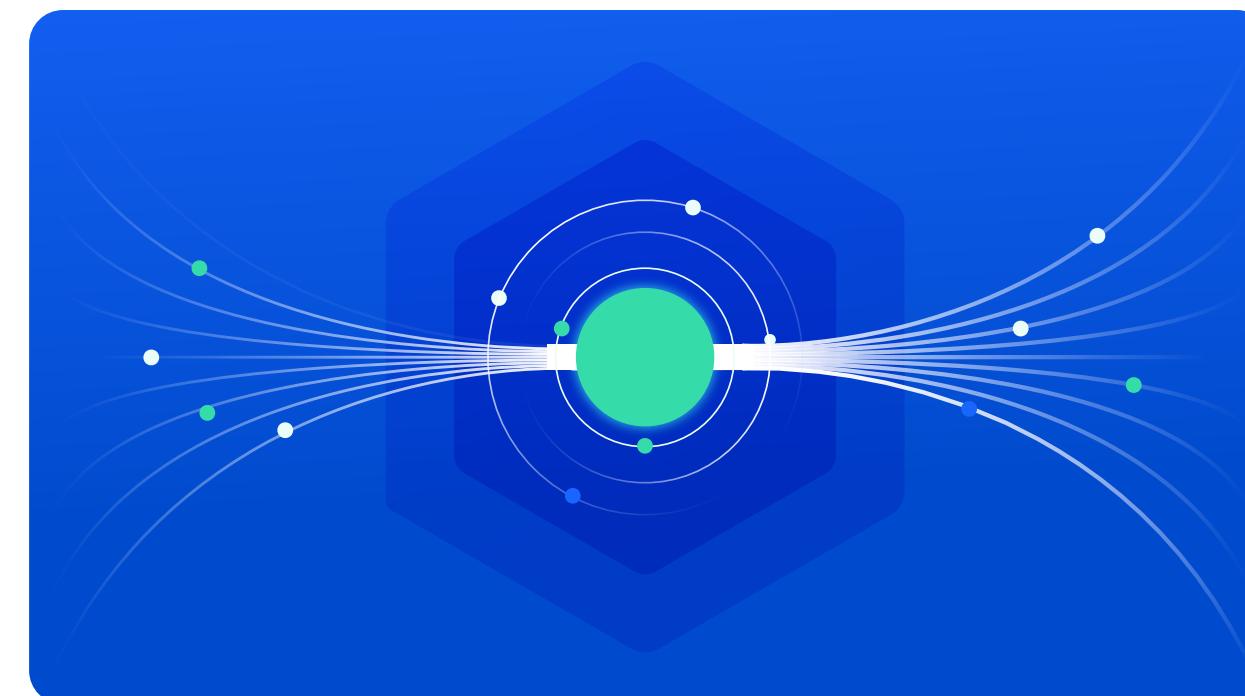
## 플랫폼 상태 관리

최적의 Kubernetes 방식을 위해서는 모든 구성 요소가 일관되게 정상적으로 작동해야 합니다. Kubernetes 플랫폼 상태 관리 솔루션은 클러스터, 노드, 네임스페이스, 워크로드, 기타 관련 구성 요소 등, 모든 중요한 Kubernetes 객체의 상태를 손쉽게 표시하고 전달할 수 있는 접근성 높은 인터페이스를 제공해야 합니다. 솔루션은 성능 저하를 실시간으로 포착하고 표시하며, 적절한 사람들에게 자동화된 알림을 보내야 합니다. 또한 플랫폼 상태 관리 솔루션은 문제를 신속하고 능동적으로 포착할 수 있도록 일관된 자동 테스트를 배포해야 합니다. 나아가 알아보기 쉬운 플랫폼 상태 솔루션은 Kubernetes 관련 지식이 부족한 사람도 문제 없이 사용할 수 있도록 하는 데 도움이 되며, 과도한 복잡성을 야기하지 않으면서 풍부한 정보와 함께 작동 상태를 자세히 보여 줄 수 있습니다.



## 상황에 맞는 중앙 집중식 데이터

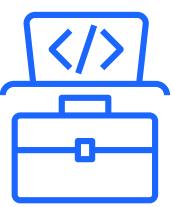
클러스터를 최적의 상태로 운영하려면 Kubernetes 플랫폼 운영 및 애플리케이션 팀원들이 일관된 정보를 잘 제공받아야 합니다. 이 같은 정보 확보를 보장하려면 중앙 집중식 데이터 소스가 중요합니다. 모든 데이터가 단일 위치에 저장되므로, 애플리케이션 및 플랫폼 운영 팀이 항상 동일한 정보 세트를 작업에 사용하게 됩니다. CPU, 메모리 사용량 및 기타 Kubernetes 시스템 요구 사항을 중앙에서 파악하면 팀이 리소스 할당 및 확장 방법을 더 잘 조율할 수 있게 됩니다. 중앙 집중식 데이터 세트를 사용하여 팀은 다양한 구성 요소나 서비스가 소비하는 리소스의 양을 직접 비교하여 최적화할 여지가 가장 큰 영역을 식별할 수 있습니다.





## Kubernetes 보안

내장된 보안 기능의 부족, 일반화된 수동 구성 방식, Kubernetes 플랫폼 자체의 지속적인 업데이트로 인해, 구식 Kubernetes 버전을 실행하는 동안 발생할 수 있는 최신 취약성을 추적하기가 어렵습니다. 다양한 클러스터와 이를 기반으로 실행되는 서비스를 안전하게 보호하려면 운영상의 취약성을 포착할 수 있는 통합 엔드 투 엔드 옵저버빌리티 및 보안 솔루션이 필요합니다. 이러한 솔루션은 런타임에 취약성을 식별함으로써 실시간으로 공격을 감지하고 차단하며, 팀이 복구 워크플로 문제 해결의 우선순위를 정하고 원활하게 해결할 수 있도록 지원해야 합니다.



## 우수한 개발자 경험

개발자 만족도와 생산성을 저해할 수 있는 Kubernetes의 복잡성을 해결하기 위해서는 Kubernetes와 전체 CI/CD 도구 스택의 개발자 경험(DevEx)을 간소화하고 자동화하는 솔루션이 필요합니다. IDP를 통해 제공되는 셀프 서비스로서, 자동화는 수작업만으로 대량의 워크로드와 포드를 유지관리하는 것이 사실상 불가능한 Kubernetes 환경에서 특히 유용합니다. 조직의 Kubernetes 환경에 특별히 최적화된 지능형 코드 제안 및 워크플로를 제공하는 AI는 Kubernetes의 복잡성과 관련한 인지 부하를 줄여줍니다.

이러한 Kubernetes 관리의 기본 요소를 도입하면 플랫폼 팀이 모든 Kubernetes 객체와 그 성능을 전반적으로 파악할 수 있게 됩니다. 전체적인 관점에서 팀은 이러한 객체가 정상적으로 안전하게 작동하고 있는지 확인하여 대규모 환경이 최적의 상태로 실행되도록 보장할 수 있습니다. 아울러, 통합 옵저버빌리티 플랫폼은 위에서 언급한 기능을 제공함으로써 Kubernetes의 중앙 집중식 관리 콘솔 역할도 합니다. 중앙 집중식 관리 콘솔을 사용하면 여러 부문의 팀이 Kubernetes 스택 전반의 모든 구성 요소를 일관된 단일 뷰로 볼 수 있습니다.

중앙 집중식 관리 콘솔의 도입만으로는 Kubernetes의 운영 우수성을 보장할 수 없지만, 모든 조직이 Kubernetes 환경 내에서 성공을 거두기 위해 실천할 수 있는 몇 가지 모범 사례가 있습니다.



# Kubernetes 관리 및 최적화를 위한 모범 사례

시작 시에, Kubernetes 환경의 복잡성을 해결하고 이를 최적화하기 위해 요구되는 작업을 수행하는 것은 플랫폼 구축 담당자와 개발자 모두에게 큰 부담이 될 수 있습니다. 이 같은 복잡성을 해결하고 올바른 솔루션을 찾기 위해서는 핵심 사용 사례를 두 가지 범주로 나누어 생각해 본다면 도움이 될 수 있습니다.

## Kubernetes 플랫폼의 상태 및 성능

Kubernetes 관리 솔루션을 사용하면 다음 질문에 답할 수 있습니다.

- **포괄적인 Kubernetes 인벤토리:** 나는 어떤 데이터가 어디에 있으며 그 관리 책임이 누구에게 있는지 알고 있는가?
- **Kubernetes 플랫폼 상태:** 플랫폼이 정상적, 안정적으로, 안전하게 작동하고 있는가?
- **리소스 사용:** 플랫폼이 최적의 상태로 효율적으로 실행되고 있는가?

## 개발자를 지원하는 플랫폼 딜리버리 및 옵저버빌리티

플랫폼이 정상적으로 작동하고 최적화되었으며 안전하다고 확신할 수 있다면, 개발 팀의 작업 효율을 높이는 솔루션을 찾았다고 할 수 있습니다.

- **옵저버빌리티 중심 개발(ODD).** SDLC의 모든 단계에서 가시성을 보장하여 아티팩트의 상태와 성능에 관한 즉각적인 피드백을 얻고 평균 옵저버빌리티 확보 시간을 단축합니다.
- **DevEx 개선.** 개발자의 니즈에 적합한 플랫폼을 설계하여, 동급 최고의 개발자 경험을 제공하고 민첩성과 유연성을 실현합니다. 특정 시나리오에서는 백스테이지와 같은 개발자 포털을 구축하기 위한 오픈 소스 프레임워크와 통합하여 이를 달성할 수 있습니다. 하지만 개발자를 염두에 두고 플랫폼을 설계하는 것이 긍정적인 DevEx를 보장하는 최선의 방법입니다.

---

다음 모범 사례는 플랫폼 구축 담당자와 개발자가 옵저버빌리티 기반 접근 방식을 사용하여 이러한 Kubernetes 목표를 달성하는 방법을 설명합니다.

---

# 모범 사례 1: 전체 Kubernetes 자산에 대한 엔드 투 엔드 옵저버빌리티 확보

Kubernetes 자산에 대한 포괄적인 옵저버빌리티는 복잡한 Kubernetes 환경을 탐색하고 모니터링하는 데 필수적입니다. 엔드 투 엔드 옵저버빌리티를 달성하는 가장 간단하고 효율적인 방법은 포괄적이고 완벽한 기능을 갖춘 플랫폼 솔루션을 도입하는 것입니다. 이 솔루션은 모든 클러스터에 걸친 전체 환경에 대한 옵저버빌리티를 제공하고, 모든 노드, 포드, 서비스 및 기타 Kubernetes 객체와 관련한 데이터와 지능적 인사이트를 제공해야 합니다. 또한 이 옵저버빌리티 솔루션은 IDP에서 제공하는 개발자 도구 세트나 프레임워크에 대한 가시성을 제공해야 합니다. 핵심은 서로 단절된 여러 데이터 포인트를 수집하고 분석하는 것이 아니라, Kubernetes 환경이 어떻게 작동하는지 전체적으로 파악할 수 있도록 전체 자산에 걸쳐 데이터를 수집하는 것입니다.

## 옵저버빌리티 배포 자동화

개발자 작업을 간소화하고 수동 구성과 관련한 복잡성을 줄이려면 모든 Kubernetes 클러스터에 옵저버빌리티가 자동으로 적용되어야 합니다. 코드 구성 템플릿을 사용하여 개발 팀이 모니터링, 옵저버빌리티 및 보안 정책을 효율적으로 설정할 수 있도록 함으로써 맞춤형 솔루션을 구축할 필요성을 최소화할 수 있습니다.

## 옵저버빌리티의 이점 확대

플랫폼 팀은 DevOps 파이프라인에 대한 주요 인사이트를 제공하고 애플리케이션 성능 모니터링(APM) 및 트레이싱과 같은 기능을 포함하는 엔드 투 엔드 옵저버빌리티를 활용함으로써 큰 이점을 얻을 수 있습니다.

APM은 Kubernetes 컨텍스트 안팎에서 애플리케이션 상태와 기능에 대한 포괄적인 가시성을 제공하여, 애플리케이션과 상호 작용하는 Kubernetes 구성 요소 및 서비스의 잠재적 병목 현상과 최적화 방법을 찾아냅니다.

분산 트레이싱은 전체 환경에서의 요청 흐름에 따라 요청에 대한 엔드 투 엔드 가시성을 지원하는 또 다른 필수 자산입니다.

## 옵저버빌리티 도구 체인 간소화

통합 솔루션을 도입하면 팀이 참조할 수 있는 단일 정보 소스를 제공하여 방대한 양의 데이터를 생성하는 복잡한 Kubernetes 환경을 다룰 때 절실히 요구되는 커뮤니케이션, 효율성 및 협업을 지원할 수 있습니다.

포괄적인 옵저버빌리티를 확보하기 위해 Kubernetes 환경에서 Dynatrace를 구현하는 방법을 자세히 알아보려면 [Kubernetes 환경에서 Dynatrace 설정](#)이라는 주제의 Dynatrace 문서를 참조하시기 바랍니다.

# 모범 사례 2: 클러스터 상태 및 보안 보장

어떠한 IDP든, 복원력, 보안, 가용성을 보장하려면 지원하는 최종 사용자 애플리케이션이 그러한 특성을 갖추어야 합니다. 개발 팀은 고객 애플리케이션을 개발하는 플랫폼을 신뢰할 수 있어야 합니다. 이 같은 신뢰는 사용하는 모든 Kubernetes 구성 요소와 서비스가 지속적으로 정상적인 상태를 유지할 것이라는 신뢰를 바탕으로 구축됩니다.

## 다중 클러스터 및 단일 클러스터 분석 중앙 집중화

Kubernetes 작업을 수행할 때 상태와 안정성에 영향을 미치는 다양한 문제가 발생할 수 있습니다. 이러한 문제는 워크로드 및 노드, 네트워킹 또는 Kubernetes 핵심 객체의 문제에 기인할 수 있습니다. 일반적인 문제 이벤트로는 노드의 리소스 부족, 컨테이너의 빈번한 재시작, 잘못된 구성이나 리소스 누락으로 인해 포드가 대기 중인 상태로 고착되는 경우 등이 있습니다. 효과적인 문제 해결을 위해서는 팀이 다중 클러스터 및 단일 클러스터 분석을 중앙 집중화하여 Kubernetes 구성 요소와 핵심 객체의 정상 작동 여부와 상태를 쉽게 확인할 수 있도록 해야 합니다.

## 사용률 세부 정보, 로그 및 트레이스 분석

Kubernetes API를 통해 노출된 데이터를 수집하는 옵저버빌리티 백엔드는 객체의 인벤토리, 그리고 단계 및 조건 등의 리소스 사용에 관한 세부 정보를 제공합니다.

옵저버빌리티 솔루션은 워크로드의 로그 및 이벤트에 대한 세부 정보를 제공해야 합니다. 이벤트에 대한 액세스는 문제를 해결하는 데 매우 중요하며, 로그는 문제가 특정 애플리케이션과 관련이 있는지 파악하는 데 중요합니다. Kubectl에서 명령줄을 실행하는 시험착오 과정 없이 주어진 상황을 명확하게 파악하려면, 로그와 이벤트 외에 인벤토리와 사용량 데이터도 필요합니다.

트레이싱은 옵저버빌리티 솔루션의 또 다른 핵심 기능입니다. 상관관계가 손상되어 워크로드에 문제가 발생한 경우, 이러한 문제의 영향, 환경 전반에 걸친 발생률 및 경로, 특정 사용자에게 미치는 영향을 파악하는 것이 중요합니다.

## AI 및 자동화를 사용하여 정확성과 효율성 증대

옵저버빌리티 데이터에서 유용하고 자동화 가능한 인사이트를 얻으려면, 문제의 근본 원인을 식별하고 심각도에 따라 우선순위를 정하는 데 효과적인 AI 기반 분석 기능이 필요합니다. 이 솔루션은 담당 팀에 보낼 알림을 트리거하고 적절한 경우 자동화된 워크플로를 시작해야 합니다.

이러한 AI 기반 운영은 Kubernetes 관리를 크게 간소화합니다. 이를 통해 플랫폼 및 개발 팀의 비 Kubernetes 전문가가 문제와 취약성을 신속하게 식별하고 해결하는 동시에, 근본 원인을 파악하여 향후 문제의 재발을 방지할 수 있습니다.

Kubernetes의 클러스터 상태 및 보안을 보장하는 방법에 대해 자세히 알아보려면 [클러스터 상태 평가 및 문제 해결](#)이라는 Dynatrace 문서를 참조하시기 바랍니다.

# 모범 사례 3: 런타임 취약성 탐지, 예방 및 해결을 통해 Kubernetes 보안 보장

업종을 불문하고 모든 조직에는 안전한 컨테이너화 환경이 필요하지만, Kubernetes는 즉시 적용 가능한 보안 보호 기능을 제공하지 않습니다. 최신 애플리케이션은 고도로 분산된 마이크로서비스로 구성되어 있기 때문에, 수백, 심지어 수천 개의 취약성이 내포된 경우가 많습니다. 이로 인해 시스템은 서비스 거부(DoS), 사이트 간 스크립팅(XSS), 원격 코드 실행(RCE) 등의 사이버 공격에 취약해집니다. 안전한 클러스터를 구축하고 배포하려면, 팀이 SDLC 전반에 걸쳐 다양한 조치를 취해야 합니다.

## 런타임에 취약성 자동 감지

최신 애플리케이션에서 방대한 양의 취약성을 신속하게 평가하기 위해서는 팀이 취약성과 공격을 실시간으로 자동 감지하고 우선순위를 정할 수 있는 기능이 필요합니다. 이 자동 실시간 감지 기능에는 서드 파티 라이브러리 및 애플리케이션 코드의 결함으로 인해 나타나는 여러 유형의 취약성이 반영되어야 합니다.

통합 옵저버빌리티 솔루션은 런타임에 이 같은 감지 기능을 수행할 수 있어야 하며, 취약성이 확대되어 시스템이 2차적인 악용 위험에 노출되기 전에, 팀이 영향을 받은 Kubernetes 객체의 취약성을 식별하고 우선순위를 정하고 해결할 수 있도록 보장해야 합니다.

## 우선순위 설정 및 자동 해결

취약성에 따라 초래하는 위험의 정도가 다를 수 있습니다. 일례로, 자주 사용되지 않는 서비스에서 발생한 심각한 취약성은 트래픽이 많은 서비스에서 노출된 중간 정도의 취약성보다 해결의 우선순위가 낮을 수 있습니다.

이러한 이유로, 옵저버빌리티 솔루션은 AI를 사용하여 취약성이 어떻게 노출되는지 자동으로 감지할 수 있어야 합니다. 또한 옵저버빌리티 솔루션은 구체적인 위험 수준, 문제 설명, 영향을 받는 구성 요소, 해결 권장 사항 등의 식별 및 해결 정보를 취약성에 태깅해야 합니다. 이 데이터를 통해 팀은 비즈니스에 있어 가장 중요한 문제에 집중하고, 대응 및 해결 워크플로를 자동화할 수 있습니다.

이러한 구체적인 Kubernetes 보안 모범 사례와 다른 여러 방식을 구현하는 방법을 자세히 알아보려면 [K8s 클러스터를 보호하는 방법](#)에 대한 이 *Is It Observable* 에피소드를 시청하시기 바랍니다. [취약성 우선순위 지정](#) 및 [해결 관리](#)라는 주제의 Dynatrace 문서에서도 자세한 정보를 참조할 수 있습니다.

# 모범 사례 4: 리소스 사용 최적화

팀이 비용을 관리하고 리소스를 효과적으로 활용하려면, Kubernetes 리소스를 자동으로 적절한 크기로 조정하고 최적화할 수 있는 수단을 갖추어야 합니다.

## 자동으로 적절한 크기로 Kubernetes 리소스 조정

비용을 절감하고 효율성을 높이는 방법 중 하나는 요청한 리소스를 충분히 활용하지 못하는 워크로드를 찾아내는 것입니다. 워크플로 자동화를 구현하여 수작업 없이 예상 소비 니즈에 따라 리소스를 확장하거나 축소할 수도 있습니다. 이를 위해서는 최적화할 여지가 있는 클러스터를 식별하고, 과거 데이터를 바탕으로 분석하여 우선순위를 정한 다음, 리소스가 보다 효과적으로 활용되도록 최적화해야 합니다.

## 워크로드 리소스 최적화

장기적으로 워크로드 사용량이 어떻게 변화하는지, 어떤 워크로드를 우선적으로 최적화해야 하는지에 대한 지식을 바탕으로, 사용량이 급증하더라도 일관된 성능을 보장하기에 충분할 정도까지만 요청 값을 낮춤으로써 워크로드의 요청과 한도를 최적화할 수 있습니다. CPU 스로틀링이나 메모리 부족으로 인한 중단을 방지하려면, 한도 수량을 실행 가능한 최저 용량으로 바로 줄여서는 안 됩니다. 그보다는, 한도 수치가 약간 줄어들었을 때 워크로드가 어떻게 작동하는지 살펴본 다음, 그 결과에 따라 조정해야 합니다.

## 디스크 공간 최적화

Kubernetes 환경의 수많은 서비스를 관리하기 위한 또 다른 모범 사례로서, 디스크 공간을 최적화할 수 있습니다. 리소스의 부적절한 할당은 용량 제한을 초래하여 서비스 속도 저하와 기타 여러 가지 문제를 초래할 수 있습니다. 수동으로 디스크 크기를 조정하려면 번거롭고 시간이 많이 걸립니다. 반면 워크플로 자동화를 통해 디스크 크기 조정 프로세스를 간소화하고 개발 및 플랫폼 팀에 충분한 시간을 확보해 줄 수 있습니다.

Kubernetes의 리소스 활용을 수동 및 자동으로 최적화하는 방법에 대해 자세히 알아보려면 [워크로드 리소스 사용 최적화](#) 및 [예측 Kubernetes 작업](#)이라는 주제의 Dynatrace 문서를 참조하시기 바랍니다.

# 모범 사례 5: 동급 최고의 DevEx 지원

고성능 소프트웨어를 제공하고 혁신을 실현하는데 있어 중추적인 역할을 하는 만큼, 조직에서 우수한 개발자 경험(DevEx)을 제공하는 것은 매우 중요합니다. 긍정적인 DevEx를 통해 생산성, 효율성, 만족도를 높이는 리소스와 방식으로 개발자의 역량을 강화할 수 있습니다. DevEx를 개선하기 위한 다음 모범 사례를 고려해 보시기 바랍니다.

## 옵저버빌리티 중심 개발 실천

확산되기 전에 선제적으로 문제를 해결하는 것은 Kubernetes와 같은 동적이고 컨테이너화된 클라우드 네이티브 환경에서 작업할 때 매우 중요합니다. 옵저버빌리티 중심 개발(ODD)은 SDLC의 모든 단계에서 개발 중인 사안에 대한 전체적인 가시성을 제공하여 일관성과 거버넌스를 지원합니다.

또한 ODD는 개발자에게 간단하고 빠르며 보다 안전한 애플리케이션 배포 방법을 제공함으로써 개발자의 작업 부담을 덜어 줍니다. 사전에 최소한의 정보만 있으면 되므로, 개발자가 성능이 뛰어난 애플리케이션을 개발하는 데 필요한 필수 옵저버빌리티 데이터를 즉시 이용할 수 있습니다.

## 기술 및 프로세스 표준화

여러 팀 간에 기술과 프로세스를 표준화하면 평균 옵저버빌리티 확보 시간을 크게 단축할 수 있습니다. 이러한 표준화를 실현하기 위해 팀은 다음 모범 사례를 통해 설정 없이 즉시 사용 가능한 옵저버빌리티 및 투명성 기능을 구현할 수 있습니다.

- 개발 및 SRE 팀이 효과적인 문제 해결 및 성능 병목 현상을 식별하기 위해 워크로드에 필요한 인사이트를 제공하는 새로운 서비스 및 애플리케이션을 개발하고 배포하기 위한 "골든 패스"를 설정합니다.
- 조직의 고유한 기술 조합, 성숙도 및 중요도 요구 사항을 바탕으로, 모니터링, 보안 및 SLO 규칙이 총족되도록 가드레일을 설정합니다.
- DevOps, 개발자, 보안, SRE와 같은 주요 팀의 이해관계자 요구 사항에 따라, SDLC의 각 단계에서 개발 팀에 필요한 최소한의 정보와 메타데이터 세트를 정의합니다. 이는 옵저버빌리티 및 보안 기준이 자동으로 일관되게 적용되도록 보장합니다.
- Monaco 또는 Terraform 같은 코드형 구성 도구를 사용하여 사전 정의된 규칙이 적용된 전용 템플릿을 준비합니다. 이 방식을 통해 처음부터 옵저버빌리티를 내장하여, 온보딩 프로세스에서 개발자의 수작업을 줄이고 DevEx를 개선할 수 있습니다.

개발자를 위해 구축된 확장 가능하고 유연한 골든 패스 템플릿에 중요한 옵저버빌리티 데이터를 포함하여, 인지 부하를 줄이고 안전하고 규정에 부합하는 애플리케이션 개발을 보장함으로써 데이 제로 옵저버빌리티가 실현됩니다.

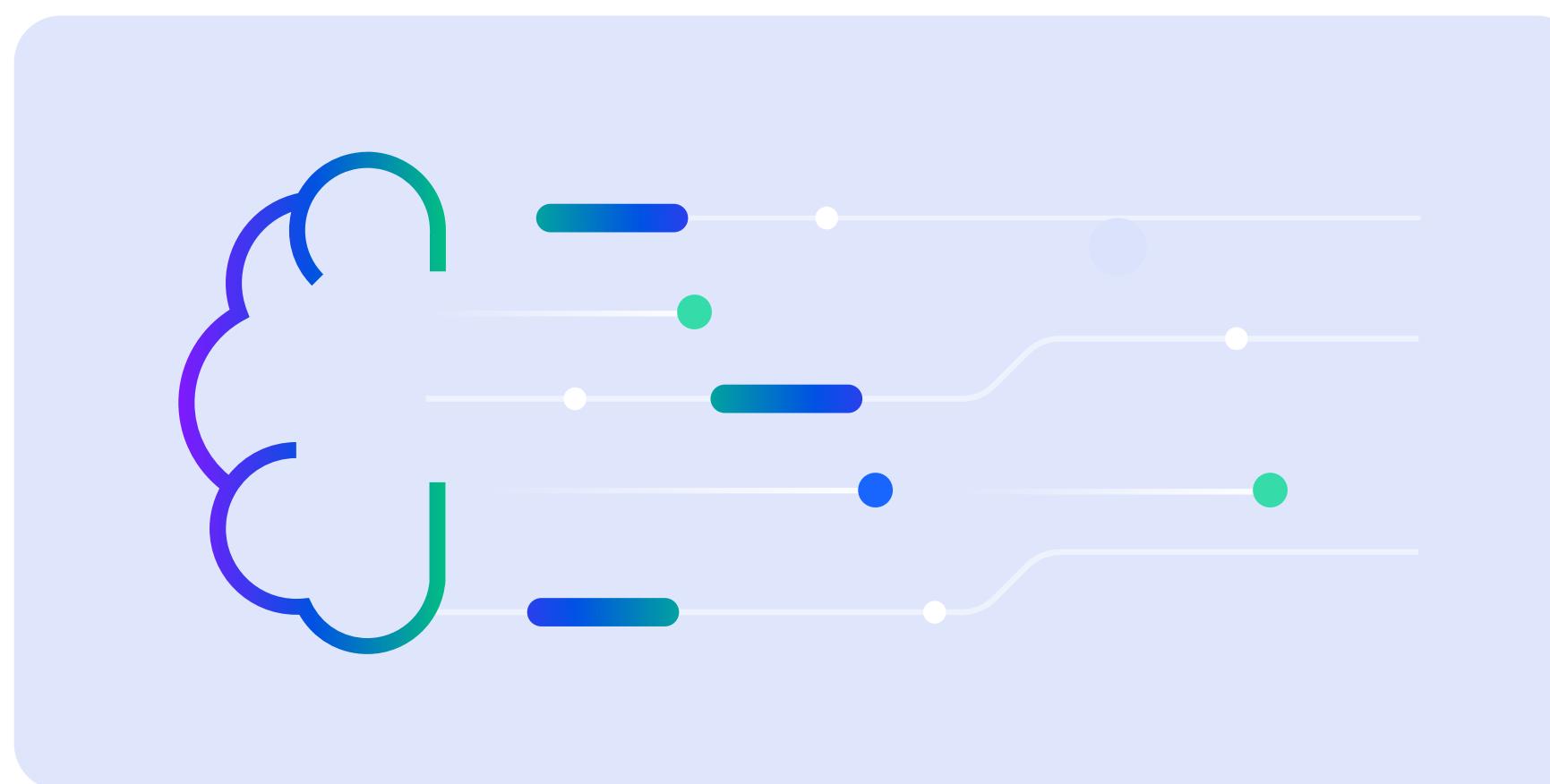
코드형 구성과 같은 방식을 사용하여 기술 및 프로세스를 표준화하는 방법을 자세히 알아보려면 [Dynatrace 코드형 구성](#)이라는 Dynatrace 문서를 참조하시기 바랍니다.

## 개발자 포털과의 통합

제품처럼 관리되는 IDP가 최고의 IDP라고 할 수 있는데, 제품은 고객의 니즈를 충족해야 합니다. 개발자는 플랫폼 팀의 주요 고객이므로, 동급 최고의 개발자 경험을 제공해야 합니다. 개발자 경험을 개선하는 데 있어서 핵심은 구체적인 니즈와 요청에 따라 맞춤형 서비스를 추가하는 것입니다. 조직의 니즈에 적합한 방식으로 구현될 때, 개발자 포털과의 통합을 통해 개발자 경험을 크게 개선하고 간소화할 수 있습니다.

Backstage와 같은 개발자 포털은 IDP를 확장하여 전체 SDLC에 걸쳐 유연성과 자율성을 보장하는 개발자 지향 기능을 제공합니다,

개발자 포털을 최대한 활용할 수 있도록 옵저버빌리티와 보안 인사이트에 쉽게 액세스할 수 있는 플러그인을 설치해야 합니다. 개발자 포털을 옵저버빌리티 솔루션과 통합하는 방법을 자세히 알아보려면 [Dynatrace를 Backstage에 통합하여 개발자 경험 개선](#)이라는 주제의 Dynatrace 문서를 참조하시기 바랍니다.



## Kubernetes 복잡성을 해결해 줄 중앙 집중식 관리 콘솔 도입

다중 클러스터 환경은 관리하기가 쉽지 않을 수 있지만, 기능을 잘 갖춘 중앙 집중식 관리 콘솔을 사용하면 전문가와 비전문가 모두 Kubernetes의 복잡성을 훨씬 손쉽게 극복할 수 있습니다. 이 단일 정보 소스를 통해 팀은 협업과 커뮤니케이션을 개선하면서, 전체 Kubernetes 자산에 대한 포괄적인 가시성을 확보할 수 있습니다.

이러한 수준의 포괄적인 가시성을 제공하는 Dynatrace 플랫폼은 통합 옵저버빌리티 및 보안 솔루션으로서 중앙 집중식 관리 콘솔 역할을 합니다. 여러 개별 도구를 사용해야 할 필요성을 해소해 주는 Dynatrace는 클라우드 네이티브 환경을 모니터링, 최적화, 보호하는데 필요한 모든 구성 요소를 제공합니다. 팀은 [Kubernetes 플랫폼 모니터링 솔루션](#)을 통해 여러 다중 클러스터 환경에 걸쳐 Kubernetes 객체 수준에서 텔레메트리 데이터를 중앙에서 수집하고, 시각화하고, 분석할 수 있습니다. 모든 옵저버빌리티 데이터(지표, 로그, 트레이스 및 이벤트)는 시스템 의존성 및 기능과 관련한 컨텍스트와 함께 중앙의 위치에 저장됩니다. 이 중앙 집중식 뷰는 Kubernetes 워크로드를 명확하게 파악하여 플랫폼, 개발 및 보안 팀이 일관된 정보를 확보하고 리소스를 보존하며 혁신 기술을 수용할 수 있게 해줍니다.

## Kubernetes 최적화를 자동화하기 위한 AI 기반 인사이트

Dynatrace는 단순한 대시보드를 넘어 Kubernetes 객체에 대한 인사이트를 제공하고 Kubernetes 자산 전반의 복구 및 최적화 프로세스를 지원하는 인과형 AI 기반의 지능형 추천을 제공합니다. 또한 [Dynatrace Davis](#)의 예측형 AI는 리소스 사용 및 소비 문제와 같은 예측된 문제 이벤트를 해결하기 위한 인사이트를 사전에 제공하여 문제가 발생하기 전에 미리 예방하는 데 도움을 줍니다.

Dynatrace를 중앙 집중식 관리 콘솔이자 단일 정보 소스로 사용하면 Kubernetes 스택을 완전하게 파악하기 위해 여러 도구 간을 전환해야 할 필요성이 적어져 개발자 경험과 팀 간 협업이 대폭 개선됩니다. Dynatrace 기능을 켜고 끄는 기능도 다양한 팀에 최적화된 맞춤형 경험을 제공합니다.

# Kubernetes 및 플랫폼 여정의 어느 단계에 있든, Dynatrace는 고객의 모든 니즈를 지원할 수 있습니다.

이제 생산성을 희생하거나 개발자에게 엄청난 작업 부담을 안기지 않고도 복잡한 Kubernetes 인프라를 관리할 수 있습니다. Dynatrace를 배포하기만 하면 Kubernetes의 복잡성을 해결할 수 있으니까요. 지금 바로 Dynatrace Kubernetes 플랫폼 모니터링 솔루션을 사용해 보시기 바랍니다.

Kubernetes 환경에서 Dynatrace 설정

플랫폼 엔지니어링 설명서

Dynatrace(NYSE: DT)는 모든 소프트웨어가 완벽하게 작동하는 세상을 만드는 것을 목표로 합니다. Dynatrace의 통합 플랫폼은 광범위하고 심층적인 옵저버빌리티와 지속적인 런타임 애플리케이션 보안 기능을 최첨단 AIOps와 결합하여, 대규모 데이터를 통해 필요한 정보와 지능형 자동화를 제공합니다. 혁신적인 기업에서는 이를 통해 클라우드 운영을 현대화 및 자동화하고, 소프트웨어를 더 빠르고 안전하게 제공하며, 완벽한 디지털 경험을 보장할 수 있습니다. 세계 최대 기업들이 디지털 혁신을 가속화하기 위해 Dynatrace®를 신뢰하는 이유가 여기에 있습니다.

클라우드를 보다 쉽게 운영하고 디지털 팀의 영향력을 극대화할 수 있는 방법이 궁금하신가요? 저희가 직접 보여 드리겠습니다. 지금 등록하셔서 [무료 15일 Dynatrace 평가판](#)을 사용해 보시기 바랍니다.

