RED HAT® OPENSHIFT

redhat.

기술 세부 정보

엔터프라이즈급 쿠버네티스(KUBERNETES) 소개

소개

컨테이너는 애플리케이션 아키텍처에 대한 사고방식과 각 팀의 비즈니스 요구 사항 이행 속도를 변화시키며, 하이브리드 클라우드 환경 전체에 걸쳐 애플리케이션의 이식성(portability)을 지원하므로, 개발자들은 기반 인프라 또는 실행 세부 사항에 신경 쓰지 않고 탁월한 제품을 구축하는 데 집중할 수 있습니다.

컨테이너 사용으로 간소화된 미니멀리즘을 구현하고 복잡성을 제거한 반면, IT 운영팀은 컨테이너 배포 관리에 대한 새로운 과제에 직면하고 있습니다. 컨테이너는 기반 리소스의 활용률을 높여 VM(가상 머신)보다 훨씬 짧은 기간에 배포되며, 관련 컨테이너 기술은 회전율을 높여 더욱 많은 오브젝트를 관리해야 하기 때문에 더욱 자동화된 정책 기반 관리를 요구합니다.

현재 기업의 많은 팀들이 쿠버네티스로 전환하고 있으며 쿠버네티스의 풍부하고 복합적인 기능들을 활용하여 컨테이너의 프로덕션, 개발 및 테스트 환경을 오케스트레이션하고 관리합니다. 쿠버네티스는 컨테이너 오케스트레이션 및 관리의 실질적인 표준으로 부상하여 기업에서 이해해야 하는 핵심 플랫폼으로 자리매김하고 있습니다.

쿠버네티스란?

쿠버네티스는 분산화되고 컨테이너화된 애플리케이션을 큰 규모로 관리하도록 돕는 오픈소스 컨테이너 오케스트레이션 플랫폼입니다. 따라서, 소프트웨어를 실행하고자 하는 환경에서 쿠버네티스를 활용하면 플랫폼이 가상으로 모든 요소를 처리합니다.

쿠버네티스는 통합 API(애플리케이션 프로그래밍 인터페이스)를 제공하여 웹 애플리케이션, 배치 작업, 데이터베이스를 배포하며, 쿠버네티스의 애플리케이션은 컨테이너로 패키징되어 해당 환경에서 확실하게 분리(decouple)됩니다. 쿠버네티스는 애플리케이션의 설정을 자동화하고 리소스 할당을 유지관리 및 트래킹합니다.

보그(BORG), 오메가(OMEGA), 그리고 쿠버네티스의 기원

컨테이너 도입에 대한 관심은 최근에 높아졌지만, 훨씬 작은 용량의 컨테이너는 이미 수십년 전에 등장했습니다. Google은 대규모로 컨테이너를 실행한 초기 기업들 중 하나였으며, 쿠버네티스를 오픈소스로 개발한 2014년보다 훨씬 이전에 컨테이너를 사용하기 시작했습니다.

www.facebook.com/redhatkorea
구매문의 080-708-0880
buy-kr@redhat.com

www.redhat.com/ko

쿠버네티스는 컴퓨팅 역사상 가장 까다로운 프로덕션 환경 중 하나인 Google의 15년이 넘게 축적된 경험을 바탕으로 개발한 제품입니다. 그 시작은 Google 내부 프로젝트인 보그(Borg, 이하 보그)로, 장시간 실행되는 서비스 및 배치 작업을 관리하여 두 개의 별도 시스템을 대체하도록 구축되었습니다. Google은 보그를 기반으로 더 많은 애플리케이션을 개발하게 되면서 다음과 같은 툴과 서비스의 광범위한 인텔리전트 에코시스템을 만들었습니다.

- 오토 스케일링
- 자가 치유(Self-healing) 인프라
- 배치 작업 설정 및 업데이트
- 서비스 디스커버리 및 로드 밸런싱
- 애플리케이션 라이프사이클 관리
- 쿼터(Quota) 관리

보그 프로젝트의 3세대가 쿠버네티스이며, 2세대는 Apache Mesos의 기준인 오메가(Omega, 이하 오메가)입니다.

오메가는 보그 에코시스템의 엔지니어링 향상을 위해 개발되었고, 점점 늘어나는 다양한 작업을 처리하는 데 필요한 더욱 지능화된 스케줄러에 주력했습니다. Google은 워크로드 일정을 예약할 수 있는 시스템을 필요로 했는데, 이 태스크는 애플리케이션과 장시간 실행되는 서비스의 양을 고려할 때 그 복잡성이 날로 증가했습니다. 이에 Google은 오메가를 개발하여 스케줄러를 중앙화된 공유 상태(shared-state) 리포지토리를 활용하여 충돌을 완화하는 두 개의 별도 스케줄러로 구분했습니다. 이로써 복잡성은 해결되었지만 여전히 시스템 개선이 필요했습니다.

쿠버네티스는 Google의 3세대 컨테이너 관리 시스템으로, 모놀리식(monolithic) 보그 컨트롤러에 사용된 접근 방식과 보다 유연한 오메가 컨트롤러 방식을 조합한 형태입니다. 쿠버네티스는 대규모 인프라 관리의 복잡성을 없애고 간소화하기 위해 설계되었습니다. 쿠버네티스 프로젝트가 시작된 후, 지속적으로 프로젝트를 지원하고 성장할 수 있도록 기여하는 전담 개발자 커뮤니티가 육성되었습니다.

컨테이너 및 현대적인 애플리케이션 개발

컨테이너가 인기 있는 이유는 애플리케이션의 자체 리소스를 사용하여 격리된 프로세스로 애플리케이션을 실행할 수 있게 하기 때문입니다. 컨테이너는 Linux® 커널이 고비용의 하드웨어 가상화 운영을 수행하는 대신 컴퓨팅 리소스를 관리하도록 지원한다는 점에서 VM과는 다릅니다. 이러한 경량화 특성 덕분에 각 애플리케이션은 자체 컨테이너를 갖추게 되어 종속성이 충돌하는 문제를 방지합니다.

간단히 말하면, 컨테이너는 zip 파일 유형인 타르볼(tarball)입니다. 이를 염두에 두고 볼 때, 컨테이너는 애플리케이션의 코드, 설정, 종속성을 번들로 구성하는 한 방법입니다. 이 번들은 일부 환경에서만 작동하는 문제를 없애줍니다. 이상적인 상황에서 애플리케이션은 환경에 상관없이 동일한 결과를 생산해야 하며 컨테이너화는 이를 손쉽게 실현하도록 합니다. 컨테이너화된 애플리케이션은 다양한 환경에서 동일한 방법으로 시작, 중지, 요청하고 로깅합니다.

기업은 컨테이너를 활용하여 다음의 이점을 실현할 수 있습니다.

- 개발자들이 환경을 디버깅하는 시간을 줄이고 코드 작성에 더 많은 시간을 할애하여 시장 출시 기간 단축
- 더 많은 컨테이너로 리소스 활용률을 높여 서버 비용 절감
- 어디에서나 실행할 수 있는 컨테이너를 통해 배포 인프라 옵션 증가

여러 구성 요소로 이루어진 복잡한 애플리케이션의 경우 컨테이너는 업데이트를 대폭 간소화합니다. 각 구성 요소를 컨테이너에 배치함으로써 파급 효과를 걱정할 필요 없이 간편하게 변경 사항을 적용할 수 있습니다. 이러한 편의성으로 인해 컨테이너는 마이크로서비스를 개발하고 배포하는 뛰어난 방식으로 자리매김했습니다. 마이크로서비스는 단일한 기능을 보유한 애플리케이션이며, 컨테이너가 구성 요소와 서비스를 분명하게 구분하기 때문에 컨테이너의 이점을 자연스럽게 갖춥니다.

컨테이너 관리 및 오케스트레이션

컨테이너에 포함하여 제공될 수 있는 개별 기능 부분으로 구축되는 애플리케이션이 증가하고 있습니다. 이는 모든 애플리케이션에 관리해야 할 부분이 증가한다는 뜻입니다. 뿐만 아니라, 컨테이너는 전통적인 VM 전용 배포보다 수명이 짧습니다. 애플리케이션에 오브젝트가 많고 이탈율이 높다면 애플리케이션 관리가 복잡해지므로 이를 해결하기 위해 설정, 서비스 디스커버리, 로드 밸런싱, 리소스 확장, 장애 발견 및 수정 등의 새로운 문제를 해결해야 합니다. 이러한 복잡성을 수동으로 관리하는 것은 불가능합니다. 클러스터는 일반적으로 1,000개 이상의 컨테이너를 실행하며, 자동화 없이는 이러한 대규모 클러스터를 업데이트하기 어렵습니다.

쿠버네티스는 컨테이너 설정 자동화, 확장 간소화 및 리소스 할당 관리를 통해 프로덕션 수준의 컨테이너 오케스트레이션을 제공합니다. 쿠버네티스는 어디에서나 실행할 수 있으며, 온사이트, 퍼블릭 클라우드, 또는 이 두가지를 결합한 하이브리드 설정에서 모두 쿠버네티스를 사용해 스케일이 큰 인프라를 실행합니다.

컨테이너 오케스트레이션으로서의 쿠버네티스

쿠버네티스는 컨테이너 오케스트레이션입니다. 쿠버네티스는 컨테이너를 실행할 위치와 방법을 결정하며, 다음의 3가지 주요 기능을 제공합니다.

스케줄러 및 스케줄링

스케줄러는 컨테이너의 요구 사항을 클러스터 상태와 인텔리전트한 방식으로 비교하여 새로운 컨테이너가 필요한 위치를 알려줍니다. 스케줄러를 대규모 일정을 관리하는 직원이라고 가정한다면, 컨트롤러는 스케줄러와 상담한 후, 작업을 할당하고 컨테이너를 모니터링합니다.

가정용 온도 조절 장치가 이러한 스케줄러의 대표적인 예입니다. 온도 조절 장치를 22도로 설정하면 그 온도를 계속 유지할 것으로 예측되며, 온도 조절 장치는 계속 온도를 확인하고 22도를 유지하도록 정기적으로 조절합니다. 즉, 온도를 설정하면 온도 조절 장치는 사용자 대신 설정 온도를 유지하는 역할을 합니다.

같은 방식으로 쿠버네티스는 사용자 대신 클러스터 상태를 유지관리하며, 개발자들은 유지관리에 대한 염려 없이 기업에 더 큰 가치를 제공하는 작업에 집중할 수 있습니다.

서비스 디스커버리 및 로드 밸런싱

어떤 시스템에서든 서비스 디스커버리는 어려운 작업입니다. 쿠버네티스도 예외는 아닙니다. 애플리케이션을 구성하는 서비스가 많을수록 애플리케이션을 트래킹하고 관리하기가 더욱 어려워집니다. 다행히 쿠버네티스를 활용하면 서비스 디스커버리를 자동으로 관리할 수 있습니다. 쿠버네티스에 데이터베이스 또는 RESTful API 같은 서비스를 실행하도록 요청할 수 있으며, 쿠버네티스는 이러한 서비스에 대해 기록한 후, 사용자가 나중에 확인할 경우 목록을 반환합니다.

쿠버네티스는 또한 개별 서비스 상태를 점검합니다. 서비스에서 충돌을 감지하면 쿠버네티스는 자동으로 서비스를 재시작하며, 이러한 기본 점검과 더불어 더욱 세밀한 상태 점검도 추가할 수 있습니다. 예를 들어, 데이터베이스가 충돌하지는 않지만 속도가 매우 느리다면 어떻게 할까요? 쿠버네티스는 느린 속도를 감지한다면 이를 트래킹하여 트래픽을 백업으로 보냅니다.

쿠버네티스는 또한 로드 밸런싱을 통합합니다. 현대적인 서비스는 서비스 복제를 실행하여 수평적으로 확장합니다. 로드 밸런서는 이러한 복제 서비스 전체의 트래픽을 분산하고 조정하는 주요 요소입니다. 쿠버네티스는 HAProxy와 같은 커스텀 로드 밸런싱 솔루션을 통합하거나 OpenStack®과 마찬가지로 Amazon Web Services, Microsoft Azure 또는 Google Cloud Platform의 클라우드 제공 로드 밸런서를 손쉽게 통합할 수 있습니다.

리소스 관리

모든 컴퓨터는 CPU(중앙 처리 장치) 처리 능력 및 메모리에 한계가 있습니다. 비디오를 트랜스코딩하는 애플리케이션에는 CPU 바운드가 있을 수 있으며, 텍스트를 파싱(parsing)하는 애플리케이션에는 메모리 바운드가 있을 수 있습니다. 다시 말해, 비디오 애플리케이션은 CPU 처리 능력을 제일 먼저 소모하는 반면, 텍스트 파싱 애플리케이션은 메모리를 먼저 소모하게 됩니다. 해당 리소스 중 하나라도 완전히 소모되면 시스템이 불안정해지고 충돌할 수 있습니다.

적절한 리소스 관리는 지능형 스케줄링을 통해 실현됩니다. 쿠버네티스는 애플리케이션을 스케줄링하여 CPU 처리 능력 및 메모리와 같은 리소스를 적절히 사용하는 한편, 활용률이 지나치게 높아 시스템 불안정을 초래하지 않도록 주의합니다.

쿠버네티스의 이점

컨테이너는 현대적인 애플리케이션을 구축하는 방식에 변화를 가져오고 있습니다. VM에서 컨테이너로의 전환은 복잡해 보이지만 그 이점은 분명합니다. 지금이 미래의 컨테이너화 환경을 시작하기에 완벽한 시기인 이유를 살펴보겠습니다.

확장성

쿠버네티스는 사용자의 요구 사항을 기반으로 클러스터를 자동으로 확장하여 리소스와 비용을 절감합니다. 1년 365일 24시간 티켓을 판매하고 시간에 따라 로드 상태가 바뀌는 온라인 이벤트 티켓 판매 서비스를 운영한다고 가정해 보겠습니다. 티켓이 판매되는 시간에는 트래픽 로드가 어떤 상태인지 대략적으로 파악하고 있고 야간에는 시스템 부하가 거의 없다는 것을 인지하더라도, 기하급수적으로 트래픽이 올라가는 매우 인기 있는 이벤트가 있을 것입니다. 이 시나리오에서 귀사는 클러스터가 동적으로 확장되어 수요를 충족시키기를 원할 것입니다. 이 때, 쿠버네티스의 Cluster Autoscaler와 Horizontal POD Autoscaler 기능을 활용하면 이러한 조정 작업을 자동으로 수행할 수 있습니다.

이식성

쿠버네티스는 어디에서나 실행됩니다. 데이터센터에서 온사이트로 실행하거나 퍼블릭 클라우드에서 실행할 수 있으며, 퍼블릭 및 프라이빗 인스턴스를 혼합한 하이브리드 설정에서 실행할 수도 있습니다. 쿠버네티스를 활용하면 동일한 명령을 어디에서나 사용할 수 있습니다.

지속적인 배포

쿠버네티스 배포는 인프라 전반에 걸쳐 지속적으로 구현됩니다. 컨테이너는 변경 불가능한 인프라의 개념을 구현하며, 애플리케이션 실행에 필요한 모든 종속성과 설정 지침은 컨테이너와 번들로 구성됩니다.

이는 "반려동물 대 가축(Pets vs. Cattle)" 논쟁으로 일컬어지기도 합니다. 반려동물은 고유의 개성을 가지고 있으며 우리는 반려동물의 이름을 지어줍니다. 예를 들어, 어떤 서버에서 동작 이상이 나타나면 그 서버의 이름을 기억하고, 시간을 들여 패치를 배포하고 커스텀 환경을 생성하여 프로덕션을 실행합니다. 하지만 이 서버를 폐기하고 다른 서버로 교체하는 것에는 많은 문제가 따를 수 있습니다. 이러한 문제는 종종 서버가 어떻게 설정되었는지 또는 초기 설정 이후 무엇이 변경되었는지를 기억하기 어렵기 때문에 나타납니다.

이에 비해 컨테이너는 가축의 경우와 같이 대량 생성되고 동일하게 설계됩니다. 컨테이너는 항상 동일한 방식으로 설정되는 데, 변경이 불가능하고 추가 설정 변경이 금지되어 있기 때문입니다. 하나의 컨테이너를 폐기하고 새로운 컨테이너를 시작하는 것을 걱정할 필요가 없습니다. 이러한 일관된 환경은 개발자들이 디버깅하는 시간을 줄이고 비즈니스 가치를 제공하는 작업에 더 많은 시간을 할애할 수 있음을 뜻합니다.

운영 및 개발의 분리 및 자동화

일반적으로 운영팀과 개발팀은 서로 경합하는 입장에 있습니다. 운영팀은 안정성을 중시하고 변경에 대해 다소 보수적이지만, 개발팀은 혁신을 중시하고 빠른 변화 속도를 추구합니다.

쿠버네티스는 이러한 충돌을 해결합니다. 쿠버네티스의 인텔리전스와 자동화 덕분에 운영팀은 시스템 안정성에 대해 확신할 수 있으며, 개발자들은 컨테이너를 사용하여 신속한 반복 주기를 유지할 수 있으므로 시간을 절약합니다.

기본 쿠버네티스 아키텍처

쿠버네티스는 수백 개의 구성 요소로 이루어진 완성도가 높은 플랫폼으로, 복잡한 특성을 가지며 뛰어난 기능들을 제공합니다. 운영자들에게는 내부 작동 원리를 이해하는 것이 중요하지만, 이 가이드에서는 적절한 쿠버네티스 추론 모델을 구축하기 위해 간소화된 아키텍처 버전으로 설명하겠습니다.

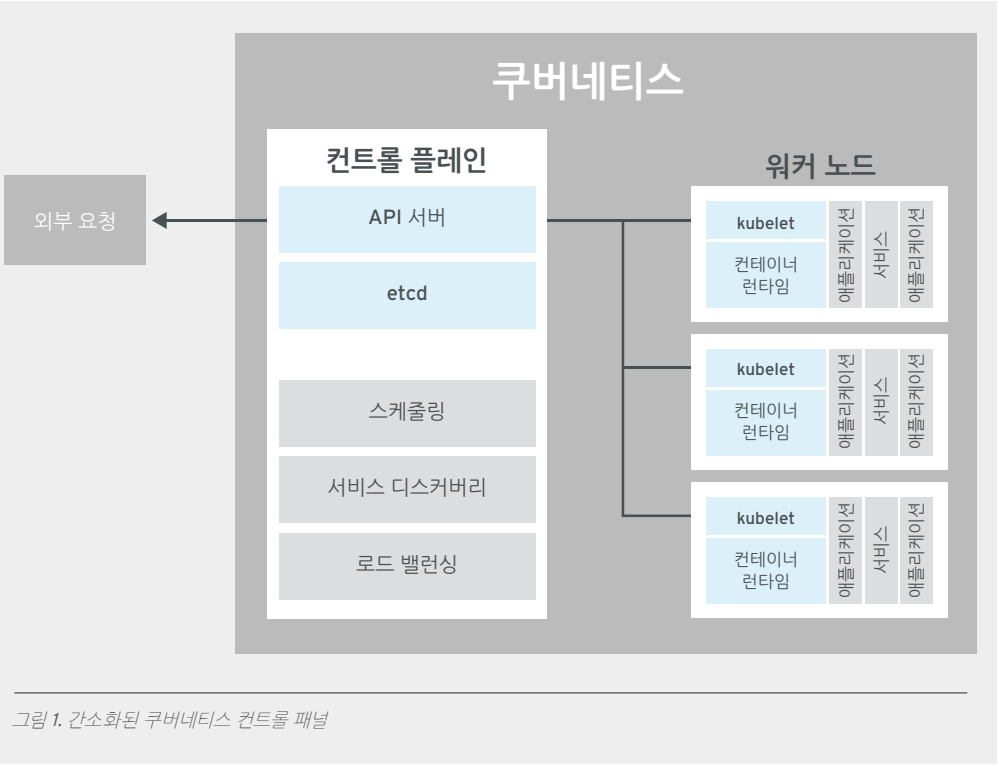

그림 1. 간소화된 쿠버네티스 컨트롤 패널

쿠버네티스는 서버의 클러스터 환경에서 컨테이너화된 애플리케이션을 관리하기 위한 소프트웨어입니다. 이러한 서버는 마스터(master) 또는 워커(worker) 노드이며, 함께 애플리케이션 또는 서비스를 실행합니다.

컨트롤 플레인(Control plane)

컨트롤 플레인은 마스터 노드 개념과 거의 동등하며, 쿠버네티스 클러스터의 핵심적인 역할을 합니다. 스케줄링, 서비스 디스커버리, 로드 밸런싱, 리소스 관리 기능은 모두 컨트롤 플레인에서 제공됩니다. 본 아키텍처는 고급 수준에서 논의되므로 기능들의 세부 정보 대신에 기능들을 컨트롤 플레인의 일부분으로서 설명하겠습니다.

- **API 서버:** 쿠버네티스의 API 서버는 모든 애플리케이션 또는 서비스의 연락 창구입니다. 모든 내부 또는 외부 요청은 API 서버로 이동하며, 서버는 요청의 유효성을 결정하고 요청자에게 적절한 수준의 액세스 권한이 있는 경우 해당 요청을 전달합니다.
- **etcd:** 컨트롤 플레인이 두뇌라면 etcd는 메모리가 저장되는 곳입니다. etcd가 없는 쿠버네티스 서버는 기억을 할 수 없는 두뇌와 같습니다. 내결함성을 갖추고 본질적으로 분산된 키-값 저장소인 etcd는 쿠버네티스의 핵심 구성 요소로서, 모든 클러스터, 클러스터 상태 저장, 설정에 대한 궁극적인 SOT(Source Of Truth) 역할을 합니다.

- **워커 노드:** 쿠버네티스의 워커 노드는 애플리케이션 또는 서비스를 실행합니다. 클러스터에는 많은 워커 노드가 있으며 새로운 노드를 추가하는 것은 쿠버네티스를 스케일링하는 방법입니다.
- **Kubelet:** kubelet은 모든 워커 노드에 위치한 매우 작은 애플리케이션입니다. kubelet은 컨트롤 플레인과 통신한 다음 워커 노드에서 요청된 작업을 수행합니다. 컨트롤 플레인이 두뇌와 같다면 kubelet은 팔과 같으며, 컨트롤 플레인은 명령을 전송하고 kubelet은 작업을 실행합니다.
- **컨테이너 런타임 엔진:** OCI 사양으로도 불리는 OCI(Open Container Initiative)에서 관리하는 표준을 준수하는 컨테이너 런타임은 컨테이너화된 애플리케이션을 실행합니다. 이는 이식 가능한 컨테이너와 기반 Linux 커널 간의 연결 통로입니다.

쿠버네티스에 없는 요소

쿠버네티스는 사용자들에게 이식성, 확장성 및 자동화된 정책 기반 관리를 제공하지만 불완전한 솔루션입니다. 운영 체제, CI/CD(지속적인 통합/지속적인 제공) 툴링, 애플리케이션 서비스 또는 스토리지와 같은 프로덕션 환경에서 컨테이너를 구축, 실행, 확장하는 데 필요한 모든 요소가 쿠버네티스에 포함되어 있는 것은 아닙니다. 또한 역할, 액세스 제어, 멀티 테넌시, 보안 기본 설정 구성을 위해 대규모 작업도 수행해야 합니다. 쿠버네티스는 사용자들에게 유연성과 선택 옵션을 제안하여 이러한 구성 요소와 서비스에 대한 플러그형 인터페이스를 제공합니다.

쿠버네티스 애플리케이션 실행 및 관리

쿠버네티스에서 애플리케이션을 실행하고 싶으신가요?

매니페스트(manifest)를 생성하고 쿠버네티스를 사용해 이를 공유하십시오. 매니페스트는 간단한 JSON 또는 YAML 파일로서, 실행할 애플리케이션 유형 및 건전한 시스템을 실행하기 위해 필요한 복제 수를 선언합니다. 쿠버네티스를 통해 매니페스트를 공유하면 API 서버를 통과하여 최종적으로 etcd에 저장됩니다. 그러면 컨트롤 플레인은 etcd에서 해당 매니페스트를 읽고 애플리케이션을 노드에 배치합니다.

확장할 준비가 되셨습니까?

애플리케이션 확장은 쿠버네티스의 편의성을 가장 잘 나타내는 사례 중 하나입니다. 매니페스트의 값을 변경하면 쿠버네티스가 요청을 충족할 때까지 더 많은 컨테이너를 배포하기 시작합니다.

pod가 충돌하면 어떻게 될까요?

쿠버네티스에 배포된 모든 요소에는 이상적인 상태가 있습니다. 예를 들어, 웹 서버를 배포하는 경우 원하는 상태를 "3"으로 설정하면, 실행할 웹 서버의 사본을 3개 요청할 수 있습니다. 이 컨테이너들 중 하나가 충돌하면 쿠버네티스는 자동으로 그 차이를 조정합니다. 쿠버네티스는 사용자가 3개의 컨테이너를 원하지만 2개밖에 없는 것을 알고, 원하는 상태를 충족하기 위해 하나 이상의 컨테이너를 추가합니다.

쿠버네티스 도입시 고려사항

쿠버네티스를 배포하기 전에 다음 질문을 고려해보십시오.

현재 애플리케이션을 어떻게 개발하고 관리하십니까?

오늘날 시장에서 경쟁력을 유지하기 위해서는 애플리케이션 개발 및 관리에 대한 컨테이너 중심적인 관점을 채택해야 하지만, 컨테이너 또는 현대적인 애플리케이션 개발은 귀사의 환경에서는 새로운 방식일 수 있습니다. 우선 애플리케이션 개발자, 비즈니스 오너, IT 운영 직원을 비롯한 모든 이해관계자가 참여하여 컨테이너 및 쿠버네티스 도입과 관련한 기업의 고유한 목표와 과제를 파악해야 합니다. 이미 컨테이너를 구축해 제공하고 있다면 쿠버네티스에서 애플리케이션을 실행할 준비가 된 것입니다.

RED HAT OPENSHIFT

redhat.

기술 세부 정보 엔터프라이즈 쿠버네티스 소개

귀사의 하이브리드 클라우드 전략은 무엇인가요?

많은 기업이 온사이트 데이터센터와 퍼블릭 또는 프라이빗 클라우드 솔루션을 결합한 하이브리드 클라우드 전략을 추진합니다. 멀티플 퍼블릭 클라우드 사용을 계획 중인 기업도 증가하고 있습니다.

쿠버네티스는 오픈소스이며 모든 유형의 인프라에서 실행할 수 있도록 설계되었습니다. 또한 쿠버네티스 페더레이션은 통합 인터페이스를 제공하여 여러 클라우드 제공업체들 간에 워크로드를 조정합니다. 가격, 서비스 수준, 또는 선호 조건이 변경되더라도 쿠버네티스는 그에 대응할 수 있는 역량을 제공합니다.

쿠버네티스 솔루션의 라이프사이클을 어떻게 관리할까요?

많은 조직들이 쿠버네티스를 자체적으로 설치하고 관리하고 있지만, 쿠버네티스 배포에 대한 상용 지원을 제공하는 벤더와의 협업을 선택하는 곳들도 있습니다. 쿠버네티스에 대한 프로덕션 지원 제공과 더불어 벤더는 다음의 이점을 제공합니다.

- 업그레이드 관리 및 중요 패치 제공
- 쿠버네티스 보안 및 쿠버네티스에서 실행하는 애플리케이션에 대한 보안 강화
- 컨테이너 레지스트리, 스토리지, 네트워킹, 클라우드 서비스, 모니터링, 로깅 및 경고(alert) 솔루션과 같은 제3사 통합 범위 검증

상용 지원되는 쿠버네티스 솔루션은 매우 다양하므로 추가 지원 및 협업 대상 요청 여부를 결정할 때 팀의 요구 사항을 주의 깊게 고려하십시오.

RED HAT OPENSHIFT를 사용한 쿠버네티스 애플리케이션의 보안, 간소화 및 스케일링

컨테이너 및 쿠버네티스를 사용할 준비가 되셨습니까? Red Hat은 쿠버네티스를 비롯한 오픈소스 컨테이너 기술의 선두주자이자 활발한 구축자로서, 컨테이너 인프라 보안, 간소화, 자동 업데이트를 위한 필수 툴을 개발합니다.

Red Hat® OpenShift®는 엔터프라이즈 쿠버네티스 애플리케이션 플랫폼입니다. Red Hat OpenShift를 통해 팀은 다음과 같은 이점을 얻을 수 있습니다.

- 각 릴리스에 수백 개의 보안, 결함 및 성능 픽스를 포함한 엔터프라이즈급 쿠버네티스 배포
- 운영팀 및 개발팀을 위한 단일 통합 플랫폼. Red Hat은 쿠버네티스에 대해 일반적인 스토리지 및 네트워킹 플러그인을 검증하고 빌트인 모니터링, 로깅, 그리고 IT팀을 위한 분석 솔루션을 포함합니다. Red Hat OpenShift는 개발자들이 언어, 프레임워크, 미들웨어 및 데이터베이스를 선택할 수 있도록 하며, CI/CD를 통한 자동화 구축 및 배포로 생산성을 대폭 강화합니다.
- Red Hat CoreOS, Linux 운영 체제 및 쿠버네티스에 대한 OTA(Over-The-Air) 업데이트 작업 자동화와 Red Hat 인증 제3사 애플리케이션 서비스에 대한 자동화된 업데이트(제공 예정)

Red Hat은 귀사의 컨테이너 및 클라우드 사용을 지원합니다. openshift.com/trial을 방문하여 Red Hat OpenShift로 쿠버네티스를 사용해보세요.

한국레드햇 홈페이지 https://www.redhat.com/korea

RED HAT 소개

Red Hat은 세계적인 오픈소스 솔루션 공급업체로서 커뮤니티 기반의 접근 방식을 통해 신뢰도 높은 고성능 클라우드, Linux, 미들웨어, 스토리지, 가상화 기술을 제공합니다. 또한, 전 세계 고객에게 높은 수준의 지원과 교육 및 컨설팅 서비스를 제공하여 권위있는 어워드를 다수 수상한 바 있습니다. Red Hat은 기업, 파트너, 오픈소스 커뮤니티로 구성된 글로벌 네트워크의 허브 역할을 하며 고객들이 IT의 미래를 준비하고 개발할 수 있도록 리소스를 공개하여 혁신적인 기술 발전에 기여하고 있습니다.

www.facebook.com/redhatkorea
구매문의 080-708-0880
buy-kr@redhat.com

www.redhat.com/ko
#f13237_0718