RED HAT®
CONSULTING

redhat.

백서

컨테이너 플랫폼을 사용하여 애플리케이션 제공 현대화

요약

비즈니스 관리자는 새로운 애플리케이션과 기능을 더욱 신속하게 사용하기를 원하고, 개발자는 소프트웨어를 구축하고 출시하는 데 있어서 효과적이고 편리한 툴을 필요로 하며, 관리자는 리소스를 효과적으로 사용하고 보안을 유지하며 규정을 준수하기를 원합니다. 또한, 조직의 모든 팀들은 비용 관리를 필수로 해야 합니다.

향상된 접근 방식과 협업, 기술 덕분에 소프트웨어의 라이프 사이클은 훨씬 빨라졌습니다. Linux® 컨테이너와 컨테이너 애플리케이션 플랫폼은 엔터프라이즈 소프트웨어의 새로운 형태로서 기업이 이러한 고도화된 기술의 혜택을 누리도록 도와줍니다. 그렇지만 이러한 새로운 기술을 이용하기 위해서 IT 관리자는 활동과 팀, 프로세스를 광범위하게 고려해야 합니다. 이 요소들이 합쳐져서 바람직한 DevOps 문화가 조성되는데, 이는 쉽지만은 않은 과제입니다.

효과적인 도입 프로그램을 통해 경험에서 배운 사례를 활용함으로써 조직은 컨테이너 기반 소프트웨어 제공 인프라를 성공적으로 구축할 수 있습니다. 이러한 인프라는 조직이 애플리케이션을 철저하게 테스트하여 신속하게 출시하고, 자주 업데이트하고 광범위하게 복제하며, 효율적으로 유지관리하고 모니터링할 수 있도록 해 줍니다.

컨테이너의 진화

비즈니스 전략에 있어 사내 소프트웨어 개발이 핵심적인 위치를 차지함에 따라 내부 개발팀에 대한 요구가 늘어나게 되었습니다. 이제 우수한 품질과 예측 가능한 비용에 더해, 출시 시간 단축도 중요한 목표가 되었습니다. 개발팀은 새로운 애플리케이션을 성공적으로 출시해야 할 뿐만 아니라, 끊임없이 진화하는 시장 상황에 맞춰 이러한 애플리케이션을 자주 업데이트해야 합니다.

애자일(Agile) 및 기타 반복(Iterative) 방법론은 개발자가 소프트웨어를 빠르게 생성하고 도입하는 데 도움이 되지만, IT 운영팀에서 애플리케이션을 프로덕션으로 릴리스하는 단계에서는 속도가 느려지는 경우가 많습니다. 개발팀과 운영팀 각각은 성공적으로 작업을 수행하지만, 이 두 팀은 하나의 통합된 팀이 아니라 개별적인 팀으로 운영되도록 만들어졌습니다. 결과적으로 양 팀 간의 신뢰도와 커뮤니케이션은 물론이고 협업을 돕는 표준과 사례, 툴을 공유하는 데 문제가 발생하게 됩니다. IT 운영팀이 소프트웨어 출시 속도를 저하시킨다고 여겨지는 경우가 많지만, 보안과 규정 준수 등의 문제도 마찬가지로 중요합니다.

IT 운영팀이 개발자의 속도에 맞추는 것을 돕기 위해, 조직은 DevOps 수용이 가능하도록 진화했습니다. DevOps 접근 방식을 사용하면 팀이 자동화를 적용하고 CI/CD(지속적인 통합과 지속적인 제공) 사례를 사용하여 소프트웨어 출시 속도를 높일 수 있습니다. 여러 환경을 위한 애플리케이션을 개발하고 배포하는 것은 여전히 복잡한 일이며, 고도로 분산된 마이크로서비스에서 웹, 클라우드 레디, 모바일 애플리케이션을 구축하는 경우는 복잡성이 더 높아집니다.

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

kr.redhat.com

컨테이너는 이에 대한 솔루션으로 떠오르고 있습니다. 개발자는 컨테이너를 사용해 애플리케이션과 라이브러리, 서버, 시스템 리소스 등과 같은 해당 애플리케이션의 모든 런타임 디펜던시를 잘 정의된 이식성 아티팩트(Portable artifact)로 번들할 수 있습니다. 컨테이너는 제공 속도 향상이라는 잠재성에 유연성을 결합하여 개발 언어에서 운영 체제에 이르기까지 기반 플랫폼의 세부사항을 정의합니다.

또한, 목표와 프로세스에 대해 조직이 더 긴밀하게 조정되므로 컨테이너는 공통의 프레임워크와 언어를 제공하여 개발팀, 운영팀, 보안팀, QA(품질 보증)팀 외 기타 팀을 하나로 연결합니다. 해당 팀들은 함께 비즈니스 요구사항을 만족시킬 수 있도록 함께 보조를 맞추면서 애플리케이션을 구축하고 배포하기 위해 컨테이너를 사용합니다.

기업의 컨테이너 사용 방식

애플리케이션을 빠르게 개발하고 지속적으로 배포하기 위해서 IT 기능들을 연계하는 것은 향상되었지만, 더욱 발전할 필요가 있습니다. 복잡한 애플리케이션은 다른 호스트들에서 실행되는 서비스로부터 구축됩니다. 이러한 호스트들은 개별적인 계층으로 잠재적으로 구성되며 프라이빗 또는 퍼블릭 클라우드에서 실행되는 물리 서버 또는 가상 서버일 수 있습니다. 그 결과, 가용성과 확장성 향상을 위해 각 서비스에 대해 일반적으로 여러 개의 인스턴스가 존재하게 되며 시간이 지남에 따라 애플리케이션은 워크로드 주기를 가지게 되고 이 주기 동안 필요에 따라 스케일 업이나 다운이 예상됩니다.

애플리케이션이 사용되면 이해관계자는 모든 위치에서 전체 구성요소와 인스턴스에 대한 통합 리포트를 통해 애플리케이션이 어느 정도로 사용되는지, 성능은 어떤지, 문제는 없는지를 파악하려 합니다. 그리고 운영팀은 애플리케이션별로 리소스 사용을 트랙킹하여 관련 경비를 적절한 비즈니스 부서에 부과하는 방법을 필요로 합니다. 문제가 발생한 경우에는 모든 구성요소와 인스턴스의 활동에 대한 종합적인 가시성이 도움이 됩니다.

단일 컨테이너식 애플리케이션의 복잡성 외에, 운영팀의 광범위한 요구를 충족하려면 규정을 준수하고 인증과 허가, 그리고 애플리케이션 구성요소의 사용에 대한 프라이버시를 제공해야 합니다.

컨테이너 플랫폼의 이점

컨테이너 애플리케이션 플랫폼은 기업이 컨테이너 배포와 관련된 과제를 해결하는 데 도움을 줍니다. 즉, 이 플랫폼을 사용하면 관리자가 기존 서버와 가상 머신을 운영할 때와 유사한 방식으로 컨테이너를 모니터링하고, 관리하고, 보안을 유지하고, 확장할 수 있습니다. 컨테이너 애플리케이션 플랫폼은 컨테이너 관리를 위한 기본적인 기능과 함께, 분산된 컨테이너의 오케스트레이션을 수행할 수 있는 기능도 제공합니다. 이뿐만 아니라, 개발과 배포의 차원을 넘어 애플리케이션 라이프 사이클 전반으로 관리의 범위를 확장하여 운영 측면에서 지속적인 프로덕션 사용을 가능하도록 만듭니다.

관리자는 컨테이너 애플리케이션 플랫폼을 사용하여 표준 운영 플랫픔 세트를 정의할 수 있으며, 개발자는 이를 통해 선택의 유연성을 확보하고 툴과 아키텍처의 공통성을 구현할 수 있습니다. 이렇게 표준화가 이루어지면 기술이 격리되는 현상이 감소하고 개발자가 IT 관리자의 개입 없이 자체적으로 표준 플랫폼의 새 인스턴스를 빠르게 프로비저닝할 수 있게 됩니다.

이 외에도 컨테이너 애플리케이션 플랫폼은 다음과 같은 이점을 제공하여 어렵고 많은 시간이 걸리는 운영 태스크의 속도를 높이면서 복잡성은 낮춥니다.

- 스테이트풀(Stateful) 애플리케이션 또는 애플리케이션 서비스를 위한 퍼시스턴트 스토리지에 간편하게 액세스
- 설정을 통해 이러한 주요 기능을 제공하여 간소화된 로드 밸런싱, 일정 관리 및 자동 스케일링 구현
- 동일한 기본 컨테이너 오케스트레이션 및 자동화 시스템을 사용하여 복잡한 애플리케이션을 물리/가상, 그리고 퍼블릭, 프라이빗 또는 하이브리드 클라우드 인프라에 간단히 배포

모든 컨테이너 실행 환경은 개별 컨테이너를 모니터링할 방법을 제공해야 합니다. Red Hat® OpenShift Container Platform과 같은 컨테이너 애플리케이션 플랫폼은 전체 인프라에 걸쳐 가시성을 제공하며 관리자가 애플리케이션의 컨테이너를 그룹화할 수 있게 합니다. 관리자는 이들 컨테이너 간의 네트워킹을 설정할 수 있으며 애플리케이션 레벨에서 액세스 및 보안, 리소스 요구사항, 우선순위 및 기타 속성을 지정할 수 있습니다. 컨테이너 애플리케이션 플랫폼을 활용하면 카피 수량이나 그 위치에 관계없이 애플리케이션의 모든 컨테이너에 대한 가시성을 얻을 수 있습니다.

이러한 기능 덕분에 컨테이너 애플리케이션 플랫폼은 기업에서 가장 중요한 애플리케이션에 사용하기에 적합하며 기존 애플리케이션이 컨테이너식 애플리케이션으로 재배포되는 경우도 많습니다.

현대적인 소프트웨어 제공

컨테이너 애플리케이션 플랫폼은 엔터프라이즈 소프트웨어의 새로운 형태로서, 애플리케이션 개발과 배포를 위한 강력한 기술을 제공합니다. 하지만 최초의 관계형 데이터베이스에서 새로운 역할과 절차, 아키텍처, 워크플로우가 필요했던 것처럼 컨테이너 애플리케이션 플랫폼에서도 팀과 작업 프로세스를 적절히 조정해야 기술을 통한 생산성과 지속 가능성을 유지할 수 있습니다.

DevOps는 이러한 요구사항이 반영된 것으로서 소프트웨어 제공을 표준화하고 자동화하여 IT팀 간의 단절을 해결하려 합니다. DevOps 접근 방식에서 역할이 바뀌는 것인 아니지만, 몇 가지 책임이 세부적으로 조정되어 팀원들이 더 큰 자율성을 얻게 되고 과제는 줄어듭니다. 프로세스와 기술을 표준화하고 자동화하면 소프트웨어를 더욱 정확하고 효율적으로 제공할 수 있습니다.

프로세스 표준화에서는 새로운 환경에 애플리케이션을 출시하는 등의 태스크를 성취하기 위한 단계를 정확히 정의하게 되며 프로세스는 그 실행 방식에 관계없이 표준화됩니다. 팀의 자율성을 유지하고, 적절한 기술과 애플리케이션 적합성을 기반으로 개발자와 운영팀의 선택권을 제공하면서, 기술 플랫폼을 표준화할 수 있습니다.

그리고 나서 표준화된 프로세스를 자동화할 수 있습니다. 자동화는 CI/CD의 핵심으로, 비즈니스 목표에 맞춰 소프트웨어 제공을 조정하는 데 매우 중요한 역할을 합니다. 자동화는 세밀하게 제어되는 체크포인트를 사용하여 실행되거나 최소한의 개입만으로 실행되도록 개발할 수 있습니다. 예를 들어 개발자가 새로운 코드를 확인할 때 코드가 자동으로 통합되고 실행 가능한 포맷으로 구축되어 자동으로 테스트됩니다. 또한, 사용자가 프로세스를 시작하도록 요구할 수 있는 자동화를 관리자가 개발할 수 있습니다. 예를 들어 개발자가 새로운 개발 환경에 대한 요청을 개시하면 운영자의 개입 없이 환경이 자동으로 생성되고 프로비저닝됩니다.

개발, 통합, 테스트, 릴리스, 모니터링, 유지관리를 포함한 애플리케이션 라이프 사이클의 고도화된 수행과정에서는 미세하게 분류된 태스크도 정의해야 합니다. 이러한 태스크는 순수한 기술의 영역을 넘어서는 것으로, 빌트인 시스템 기능의 확장을 의미할 수 있습니다. 예를 들어 불필요하게 리소스를 소비하는 미사용 컨테이너가 축적되지 않도록 미사용 컨테이너 작동을 중단할 수 있는 경우에 관한 규칙을 수립할 수 있습니다. 대부분의 조직은 IT 비용을 비즈니스에 부과하기 위한 방법을 마련해 두고 있으며, 이는 공유된 컨테이너 기반 플랫폼을 위해 조정되어야 합니다.

고도로 자동화된 컨테이너 기반 환경은 개발자와 운영 직원 모두에게 큰 이점을 제공합니다. 개발자는 실제 개발에 더 많은 시간을 할애하고 플랫폼 구축에 드는 시간은 줄일 수 있으며, 제품 책임자(Product owner)로부터 빠른 시간 내에 피드백을 전달받아 기능을 빠르게 구축하고 문제를 해결할 수 있습니다. 자동화된 테스트에서는 제품을 출시하기 전에 문제를 찾아낼 수 있어 더욱 높은 품질의 소프트웨어를 구현할 수 있습니다. 그리고 자동화된 플랫폼 구축을 통해서는 QA, 테스트, 프로덕션 등의 플랫폼 간 일관성이 높아져 설정의 차이를 발견하는 데 드는 시간이 줄어듭니다. 운영팀은 모든 애플리케이션에 동일한 방식으로 구현되는 빌트인 가용성과 확장성으로부터 이점을 얻을 수 있으며, 맞춤형 플랫폼을 지원해야 할 필요성도 줄어들게 됩니다. 자동화된 배포에서는 인적 오류가 발생하지 않기 때문에 운영팀이 애플리케이션 픽스를 긴급하게 배포해야 할 필요성이 없어지며, 일관적인 단일 플랫폼은 보안을 유지하고 모니터링을 수행하기가 매우 쉽습니다.

컨테이너 도입

기업은 현재 애플리케이션 개발 상황을 평가하고, 점차적으로 컨테이너 기반의 현대적인 접근 방식으로 마이그레이션하기 위해 프로젝트가 상호 연결된 유연한 프로그램을 필요로 합니다. 이러한 프로젝트는 성과를 신속하게 보여주고 소프트웨어 제공 속도를 대폭 향상하는 데 필요한 전사적인 변화를 이뤄낼 수 있도록 기술적이고 전략적인 접근 방식을 사용하여야 합니다. 그리고 프로그램의 궁극적인 초점은 DevOps 향상에 필요한 문화적 변화를 실현하는 것이 되어야 합니다.

개발자나 운영팀이 혁신적인 신기술과 프로세스를 항상 환영하는 것은 아닙니다. 이 새로운 접근 방식이 어렵지는 않지만, 기존 방법에 익숙해져 있는 제품 책임자, 개발자, 운영 직원에게는 새로운 과제가 될 수 있습니다. 자신의 의견이 반영되지 않은 채 변화가 일어나면 사람들은 저항하기 마련입니다. 모든 참여자들이 자연스럽고 적극적으로 지원할 수 있도록 다음과 같은 접근 방식을 취해야 합니다.

- 초기 계획 단계부터 모든 이해관계자를 참여시킵니다.
- 측정 가능한 초기의 성과를 보여주고 함께 축하합니다.
- 조직에서 변화를 직접 경험하였으며 다른 팀에 이를 홍보할 수 있는 지원자를 모집합니다.

이 새로운 접근 방식은 반복적인 프로그램을 통해 구현되어야 하며 잘 정의된 태스크, 바람직한 결과, 핵심목표와 함께 개별 반복에서 비즈니스 가치를 입증한다는 목표를 포함해야 합니다. 이 프로그램의 단계는 다음과 같은 세 가지 작업 흐름으로 분류됩니다.

- **인프라:** 컨테이너 플랫폼을 구축하고 엔터프라이즈 시스템에 통합하는 데 필요한 서버와 소프트웨어
- **릴리스 관리:** 애플리케이션을 배포하고 제공하는 프로세스의 표준화 및 자동화
- **개발:** 개발자가 컨테이너식 애플리케이션을 구축하거나, 기존 애플리케이션을 컨테이너로 마이그레이션하거나, 컨테이너 오케스트레이션을 사용하는 마이크로서비스를 개발하는 데 필요한 프로세스와 툴 포함

탐색과 설계

초기 단계부터, 관련된 모든 팀의 이해관계자가 초기 계획에 참여합니다. 컨테이너와 컨테이너 플랫폼과 같은 신기술에서, 단기간의 파일럿 구현을 통해 팀은 장기적인 구현에 대한 초기 설계 논의를 진행하기 전에 기술을 충분히 파악할 수 있습니다. 컨테이너 애플리케이션 플랫폼의 레퍼런스 구현을 구축하고, 애플리케이션의 소규모 세트를 컨테이너로 마이그레이션하여 실제 환경에서 플랫폼을 시연할 수 있습니다. 설계 워크샵을 실시하기 전에 이렇게 레퍼런스를 구현하면 참여자들이 실제 플랫폼의 동작을 초기에 확인할 수 있게 됩니다.

구현과 자동화

다음 단계에서는 프로그램의 작업 흐름이 동시에 완료될 수 있도록 분리됩니다. 인프라팀은 초기 설계 반복에서 정의된 전체 컨테이너 애플리케이션 플랫폼을 구축할 수 있고, 프로덕션 구현의 설계는 시간이 지나면서 점점 발전하며 플랫폼에 대한 최소한의 실효성이 정의됩니다. 이와 동시에 릴리스 관리팀은 버전 생성, 아티팩트 리포지토리 그리고 구축, 테스트, 개발 단계를 위한 자동화 등의 소프트웨어 개발 구성요소를 포함하여 CI/CD 파이프라인의 설계와 구축을 시작할 수 있습니다. 이 작업은 이후의 반복에서 계속됩니다. 마지막으로 애플리케이션 개발팀은 컨테이너식 개발을 위한 툴과 기술에 대한 교육을 받고, 이 플랫폼을 사용하여 애플리케이션의 개발을 시작합니다.

다음 단계와 그 이후에 필요한 단계에서 릴리스 관리팀은 계속해서 포괄적이고 자동화된 파이프라인을 구축하며, 컨테이너식 애플리케이션을 자동으로 제공할 수 있도록 프로세스를 확장합니다.
애플리케이션팀은 기존 애플리케이션을 컨테이너 애플리케이션 플랫폼으로 마이그레이션하기 시작합니다. 하지만 모든 애플리케이션을 반드시 이전해야 하는 것은 아니며, 팀은 애플리케이션을 분석하여 마이그레이션에 어느 정도의 작업이 필요할지 평가하는 방법을 익혀야 합니다. 기존 애플리케이션 중 일부는 마이크로서비스로 분할하여 재설계함으로써 효율을 높일 수 있으며, 이 단계에서 개발팀은 컨테이너 기반의 마이크로서비스 접근 방식을 더 심층적으로 이해하게 됩니다.

지속적인 개선

이후 단계에서는 이전 단계의 피드백을 통해 지속적인 향상을 모색합니다. 인프라팀은 Identity 관리 추가, 컨테이너 플랫폼 기능을 기반으로 한 인프라의 동적 프로비저닝은 물론, 패치, 업그레이드, 그리고 재해 복구를 위한 개발 운영을 포함하여 점차 복잡해지는 통합과 관리 작업을 해결할 수 있습니다. 릴리스 관리팀에서는 자동화 접근 방식을 더 정교화하고 지표를 개발하여 실제 데이터를 기반으로 전반적인 파이프라인을 최적화할 수 있으며 애플리케이션팀은 초기에 애플리케이션을 컨테이너로 마이그레이션하면서 습득한 내용과 접근 방식을 사용하여 성공적인 마이그레이션을 위한 시간과 비용을 줄일 수 있습니다. 마이크로서비스 개발팀은 계속해서 서비스를 확장하여 텔레메트리를 통해 전체 기능이 갖춰진 유연한 분산 시스템을 구축함으로써 성능 관련 병목 현상을 식별하고 개선이 가능한 영역을 발견할 수 있습니다.

결론

가상화의 시대처럼, 컨테이너는 IT 트랜스포메이션의 현대화된 시대를 보여줍니다. 하지만 컨테이너는 가상화와 달리 IT 운영 범위를 초월하여 여러 팀의 세부적인 참여를 필요로 합니다. 이 결과로서, 비즈니스는 해당 팀들이 협력하는 방식을 변경해야 합니다. 컨테이너를 성공적으로 도입하려면 사람과 프로세스도 기술만큼 중요합니다.

사람, 프로세스, 기술을 모두 해결하는 체계인 애자일(Agile) 프로그램이 구현되었을 때 기업은 성공에 이를 수 있습니다. 앞서 설명한 단계적 접근 방식을 사용하면 초기에 가치를 입증할 수 있으며, 이후 반복 과정에서 성공을 위한 지원을 획득할 수 있습니다. 개별적인 작업 흐름은 각 팀이 변화를 수용할 수 있는 역량에 맞춰 원하는 속도로 진행할 수 있게 해 주며 계획을 더 확장하여 진행할 수 있도록 해 줍니다.

기술을 더 큰 규모의 조직적인 변화를 실현하기 위한 기반으로 사용함으로써 조직은 오늘날 기업이 성공하기 위한 열쇠인 효율적인 소프트웨어 제공을 달성할 수 있습니다.

Red Hat Consulting은 본 문서에 소개된 통합적인 접근 방식을 사용하여 기업이 컨테이너의 전체 가치를 성취할 수 있도록 도와줍니다. 자세한 내용은 redhat.com/ko/resources/modernize-application-delivery-container-platforms-datasheet에서 해당 데이터시트를 참고해 주십시오.

한국레드햇 홈페이지 https://www.redhat.com/ko/global/south-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, 전세계 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.

아시아 태평양
+65 6490 4200

인도네시아
001 803 440224

뉴질랜드
0800 450 503

베트남
800 862 6691

호주
1 800 733 428

일본
03 5798 8510

필리핀
800 1441 0229

중국
800 810 2100

브루나이 및 캄보디아
800 862 6691

한국
080 708 0880

싱가포르
800 448 1430

홍콩
852 3002 1362

인도
+91 22 3987 8888

말레이시아
1 800 812 678

태국
001 800 441 6039

대만
0800 666 052

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

kr.redhat.com
#INC0413975_v1_201606_KVM