New Relic

White Paper

데브옵스의 다음 단계

현대 클라우드 환경의 복잡성을 극복하는 3가지 방법

개요: 복잡성의 쓰나미

수년 동안 뉴렐릭은 조직들이 데브옵스(DevOps)를 올바르게 구현할 수 있도록 지원해왔습니다. 뉴렐릭은 자사의 데브옵스 경험을 공유하고, 조직이 데브옵스 및 클라우드 네이티브 기술을 최대한 활용하는 데 도움이 되는 새로운 기능들을 지속적으로 제공했으며, 데브옵스 성공을 측정하고 추적하기 위한 데이터의 기반 접근 방식과 모범 사례를 주도했습니다.

데브옵스는 주류로 자리를 잡았고, 이를 수용하는 조직들은 계속해서 성공을 거두고 있습니다. 이제 데브옵스와 클라우드 네이티브 기술 및 기법이 야기하고 있는 심각한 결과도 인지해야 합니다. 클라우드에서 마이크로서비스, 서버리스, 컨테이너 및 기타 접근 방식과 기술을 도입하는 기업들이 직면하게 되는 복잡성이 바로 그러한 결과입니다.

데브옵스와 클라우드 네이티브 접근 방식이 나쁘다는 얘기가 아닙니다. 그 반대로, 두 가지의 이점은 복잡성으로 야기되는 비용보다 훨씬 큽니다. 중요한 것은, 복잡성이 IT와 비즈니스 결과에 부정적인 영향을 미치기 전에, 완화 조치를 취해야 한다는 것입니다. 현대 애플리케이션 개발의 다음 단계를 위한 기술, 모범 사례 및 툴을 아직 확보하지 못한 채 데브옵스를 구현하고자 하는 기업은 애플리케이션 성능 및 가용성 목표를 계속 충족하기가 매우 어렵기 때문입니다.

이 화이트 페이퍼에서는 데브옵스의 현재 위치와 복잡성이 증가하는 이유, 그리고 조직이 복잡성을 보다 신속하게 감소시킬 수 있는 방법을 살펴봅니다.

데브옵스에 대한 배경 지식이 더 필요하십니까?

이 문서는 데브옵스에 대한 기본적인 이해를 전제로 합니다. 필요한 경우 데브옵스 여정의 모든 단계에서 조직에게 도움을 주는 몇 가지 추가 리소스를 확인해보시기 바랍니다.

- *데브옵스(DevOps)란?*
- *효과적인 데브옵스: 성공적으로 장애물을 제거하기 위한 모범 사례*
- *측정할 수 없는 데브옵스는 실패작*

데브옵스, 영원하라!

데브옵스의 원칙, 문화, 프로세스 및 툴은 오늘날의 조직들이 소프트웨어 중심의 세상에서 직면하는 과제, 즉 디지털 트랜스포메이션, 혁신, 시장 출시 가속화, 신속한 변화 적응, 완벽한 고객 경험을 제공하는 데 도움이 됩니다.

DevOps Research and Assessment(DORA)는 2019 데브옵스 현황보고서에서, "많은 분석가들이 데브옵스 및 기술 혁신과 관련해 업계가 '고비를 넘겼으며,' 올해의 분석을 해본 결과 이러한 사실을 확인할 수 있었다"며 "산업 속도가 증가하고 있고, 속도와 안정성 모두를 확보하는 것이 가능해졌으며, 클라우드 기술로의 전환이 이러한 가속화를 추진하고 있다"고 적었습니다.

DORA의 보고서는 속도와 안정성이 증가된 요인으로 클라우드를 지목하고 있지만, 분석 결과에 따르면 클라우드 컴퓨팅의 사용은 소프트웨어 배포의 성능과 가용성을 예측할 수도 있는 것으로 나타났습니다.

데브옵스에서 역할

원래 데브옵스 내에서 역할과 책임이 더 뚜렷했습니다.

- ***데브:*** *엔지니어가 인프라, 프로비저닝, 용량 등 전체 애플리케이션을 책임집니다.*
- ***옵스:*** *엔지니어가 데브와 유사한 툴을 사용해 모니터링 및 자동화/스크립팅 등 인프라를 관리합니다.*

오늘날 데브옵스 목표가 '고객에게 향상된 기능을 제공할 수 없게 가로막는 모든 조직적 장벽을 제거하고, 툴링과 자동화로 가속화하는 것'으로 발전함에 따라, 역할을 명확하게 정의하기가 어려워졌습니다.

DORA 보고서는 가장 높은 성과를 내는 팀의 경우, 낮은 성과를 내는 팀보다 National Institute of Standards and Technology(NIST)가 정의한 클라우드 컴퓨팅의 5가지 기능 모두를 수행할 확률이 24배 더 높다고 설명합니다.

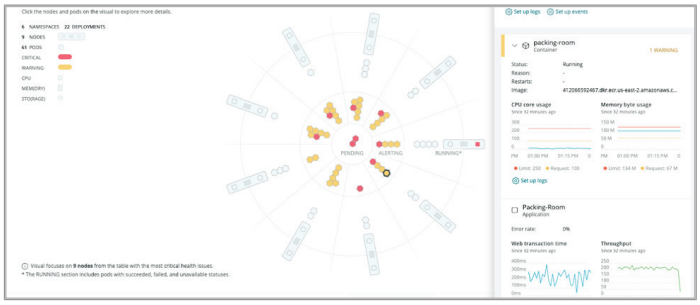
예를 들어, AWS의 클라우드 컴퓨팅은 엔지니어가 AWS 서비스 카탈로그를 통해 인프라 프로비저닝을 위한 셀프 서비스를 생성할 수 있게 해줍니다. 개발자는 서비스가 프로비저닝될 때까지 기다릴 필요 없이, 신속하게 새로운 것을 시도해보고 신제품을 더 빠르게 출시할 수 있습니다. 완전 관리형 서비스는 팀이 AWS 리소스를 활용할 수 있도록 지원해주며, AWS 개발자 툴의 형태로 제공되는 자동화는 더 빠르고 효율적으로 개발 작업을 할 수 있도록 도와줍니다.

데브옵스를 올바르게 구현하면, IT와 비즈니스 모두 엄청난 성공을 거둘 수 있다는 사실은 확실합니다. 그런데 어두운 그림자가 드리워지고 있습니다. 역풍이 불기 시작하는 것일까요?

성공의 대가

현대적인 애플리케이션 개발의 속도, 민첩성, 탄력성, 확장성 및 기타 핵심적인 혜택을 얻는 대신, 데브옵스 팀은 대가를 지불해야만 하게 되었습니다. 바로 복잡성이라는 대가입니다. 모놀리식(monolithic) 애플리케이션은 복잡하지 않고(취약하고 다루기 어렵고 확장하기 어려운 기타 특성들로 인해 혁신 속도를 느려지게 하는), 현대 클라우드 네이티브 환경은 일부 팀이 미처 대처할 준비가 되지 않은 다른 유형의 복잡성을 야기하고 있습니다.

팀들은 모놀리식(monolithic) 애플리케이션을 독립적으로 소유 및 배포할 수 있는 더 작은 마이크로서비스로 분할함으로써 현대화를 합니다. 마이크로서비스는 유지 관리가 더 용이하고 재사용이 가능해, 개발을 가속화할 수 있습니다. 동시에 Amazon Elastic Container Service(ECS), Amazon Elastic Kubernetes Service(EKS) 또는 서버리스 AWS Lambda 등을 사용해, 컨테이너(도커)와 컨테이너 오케스트레이션(쿠버네티스) 같은 최신 기술을 도입해 애플리케이션의 휴대성과 확장성을 향상시킬 수 있습니다.



특정 파드(pod)의 KPI를 표시해주는 쿠버네티스 클러스터 탐색기

이 모든 것으로 인해, 수백 개의 컨테이너에서 실행되거나 서버리스 함수로 실행되는 마이크로서비스의 수가 증가합니다. 거기에다 매일, 매시간마다 다양한 서비스가 동시에 배포되고, 컨테이너, 로드 밸런싱, 자동 확장과 같은 유연한 인프라와 자동화까지 추가됩니다.

결과적으로, 매 순간 성장하고 변화하는 '이동 부품'이 무수하게 생겨납니다. 문제의 근본 원인을 어디에서 찾아야 할까요? 문제가 특정 상황에서 짧은 시간 동안만 컨테이너 내부에서 존재하는 경우도 물론 있을 수 있습니다.

복잡성을 야기하는 것은 이동 부품 뿐만 아니라, 각 부품이 생성하는 운영 소음의 양입니다. 모든 인프라와 마이크로서비스에서 메트릭, 이벤트, 로그 및 추적 데이터가 생성됩니다. (이 텔레메트리 데이터 모음을 MELT라고 하는데, 여기에서 자세한 내용을 확인할 수 있습니다.) 성능 문제의 진정한 원인을 이해하려면 어디를 확인해야 할까요?

데브옵스 여정의 다음 단계는 실행 가능한 인사이트를 지속적으로 확보하며 복잡성을 줄이고 소음을 줄이는 것입니다.

데브옵스의 다음 단계를 위한 핵심 원칙

지금까지 팀은 대부분의 노력을 데브옵스의 '데브(개발)' 부분에 집중하여, 소프트웨어의 안정성과 가용성은 높이고, 오류는 줄이면서 더 빠르고 더 자주 배포할 수 있는 능력을 얻는 데 중점을 두었습니다.

이제 데브옵스의 '옵스(운영)' 부분에 초점을 맞출 때입니다. 현대 클라우드 네이티브 애플리케이션 및 환경의 복잡성을 헤쳐나가는 데 아래의 두 가지 주요 원칙은 도움이 될 것입니다.

1. 올바른 문화 구축

데브옵스 성공에 문화는 중요한 역할을 하며, 앞으로도 그러한 사실은 변함이 없을 것입니다. 문화는 팀이 어떻게 구성되고, 팀 구성원이 어떻게 공동으로 협력하며, 구축 중인 소프트웨어에 대한 책임을 어떻게 공유하느냐에 중점을 둡니다. 팀 내, 그리고 팀 간의 투명성과 소통은 데브옵스의 목표를 지원하는 공통된 이해를 확보하는 데 도움이 됩니다.

"소프트웨어를 구축하는 새로운 마이크로서비스 방식은 신속한 대규모 변화에 최적화되어 있지만 그 대가로 복잡성이 증가합니다. 분산 시스템에 뒤따르는 복잡성에 준비가 되지 않은 조직이 쿠버네티스 같은 강력한 기술을 도입하면, 실질적인 위험이 발생할 수 있습니다. 고객들에게 최신 시스템들에 대한 준비를 갖추고 최신 툴을 효과적으로 사용할 수 있도록 조치를 진지하게 검토해 줄것을 권장하고 있습니다."

켈리 루니(Kelly Looney), SI 실무 책임자, 데브옵스, Amazon Web Services

그러나 핵심 특성 중 하나인 신뢰가 때로 간과됩니다. 신뢰는 데브옵스 팀이 받아들이고 익혀야 하는 가장 중요한 문화적 요소입니다. 리더는 팀원들을 신뢰하고, 팀은 다른 팀들을 신뢰해야 하며, 팀 구성원들은 서로를 신뢰해야 합니다.

신뢰 없이는 자율성과 책임이라는 다른 필수적인 문화적 특성이 제 역할을 할 수 없습니다. 팀이 출고하는 제품을 온전히 소유할 수 있도록 자율성과 책임을 부여하고, 올바르게 수행하는 데 필요한 툴 (예: Amazon Elastic Beanstalk, AWS Lambda 및 AWS CloudFormation)을 제공하여, 그들이 제대로 일을 해낼 것이라고 믿어야 합니다.

2. 모든 것을 계측 및 측정

측정은 데브옵스 전문가인 제즈 험블(Jez Humble)이 만든 CALMS 프레임워크(Culture, Automation, Lean, Measurement, Sharing)를 지탱하는 5개 기둥 중 하나입니다. 복잡성이 증가함에 따라, 데브옵스가 성공적으로 기능을 발휘하는 데 포괄적인 문맥(context) 데이터가 필요해집니다. 이러한 복잡성에 대처하려면, 아무리 짧은 기간이라도 아키텍처를 전체적으로 파악하는 것이 중요합니다. 올바른 데이터가 있으면 무엇이 제대로 작동되고, 무엇이 그렇지 않은지, 어디에 팀을 집중시켜야 하는지를 알 수 있습니다.

데이터는 의사 결정 과정에서 감정과 책임전가를 배제하고, 대신 협업 및 공감의 문화를 촉진합니다. 데브옵스 팀의 모든 구성원에게 기술, 경험 및 역할에 대한 공통 언어를 제공합니다.

넷플릭스, Operate-What-You-Build 모델 도입

기술 블로그 게시물에서, 넷플릭스는 소프트웨어 구축 팀이 운영과 지원까지 담당하는 소유권 공유 접근 방식으로 전환하기로 결정한 이유와 그 방법을 설명합니다.

복잡성을 극복하고 결과를 지속적으로 개선하는 3가지 방법

복잡성의 다음 단계를 준비하며 데브옵스 노력을 계속 성장 및 발전시켜 나가는 과정에서, 팀이 옵스의 성과를 향상하기 위해 마스터해야 하는 세 가지 역량이 있습니다.

1. 옵저버빌리티(Observability)

현대 클라우드 네이티브 환경의 복잡성에 대한 해결책은 옵저버빌리티, 즉 관찰성 입니다. 조직이 모든 것을 계측하고 있고, 텔레메트리 데이터를 사용해 시스템 내의 관계와 종속성은 물론 성능과 상태에 대한 기본적인 실무 지식을 구축하고 있다면, 관찰성을 활용하고 있는 것입니다.

옵저버빌리티는 실시간으로 모든 성능 데이터를 연결해 한 곳에서 문맥적으로 표시해주며, 문제를 더 신속하게 파악하고 문제의 근본 원인을 이해하여 궁극적으로 탁월한 고객 경험을 제공할 수 있도록 해줍니다. 업데이트 중인 서비스의 변경 사항으로 인해 다른 연결된 서비스가 중단되지는 않는지 등, 팀이 정보에 입각해 변경 사항에 대한 결정을 내리는 데도 도움을 줍니다.

옵저버빌리티에 대해 자세히 알아보려면 저버빌리티의 시대: 미래가 오픈, 커넥티드, 프로그래머블인 이유를 확인해 보시기 바랍니다.

---

2. 신속한 피드백 루프

'빠르게 실패'함으로써 실패에서 성공하는 것이 데브옵스가 원하는 역량입니다. 저자이며 연구원 및 데브옵스 전문가인 진 킴(Gene Kim)은 증폭된 피드백 루프를 데브옵스의 프로세스, 절차 및 관행을 구성하는 세 가지 방법 중 하나로 꼽습니다. 그는 지속적으로 수정이 가능하도록 단축 및 증폭하는 것을 목표로, 부정적 피드백에서 긍정적 피드백으로 이어지는 루프를 생성하는 것이 중요하다고 말합니다.

빠른 피드백 루프는 신뢰의 원칙에 팀이 다음 작업을 수행하는 데 도움이 되는 툴 및 프로세스를 결합합니다.

- **실험 장려:** 혁신과 대담한 아이디어(실패 포함)를 존중하는 문화를 촉진해 시장 출시 시간을 단축하고, 새로운 기술을 채택하며 비즈니스 성과를 개선합니다.
- **빈도 높은 배포:** 마이크로 기능 및 버그 수정을 소규모로 빈번하게 배포하도록 권장하고 각 배포의 성공 여부를 측정합니다. 이 웨비나를 통해 파이프 라인을 계측하는 방법을 자세히 알아보시기 바랍니다.
- **위험 수용:** 배송을 더 빠르게 하기 위해서는 어쩔 수 없다는 사실을 인지하면 작고 영향이 적은 장애를 허용합니다. 일관성이 완벽함보다 더 중요하다는 것을 이해하며, 더 소소한 운영 문제가 더 빈번하게 생길 것이라는 사실을 인정합니다. 발생하는 운영 문제는 전체적으로 대규모 중단보다 영향력이 훨씬 적습니다.
- **모든 실수로부터 배우는 교훈:** 인시던트를 해결하는 과정을 정확하고 철저하게 문서화해 교훈을 얻고, 인시던트가 재발하지 않도록 예방 조치를 취합니다. 이를 달성하는 바람직한 방법은 처벌이나 비난이 아닌 건설적인 학습과 개선에 초점을 맞춰 회고를 하는 것입니다.
- **릴리스 대시보드 사용:** 모든 기능 릴리스에 대해 관련된 핵심 성과 지표를 추적하는 '기능 대시보드'를 만듭니다. 대시보드는 애플리케이션에 미치는 영향을 이해하기 위해 기능 배포와 해당 기능별 AWS 서비스 사용을 추적해야 합니다.

특정 기능 배포와 관련된 KPI

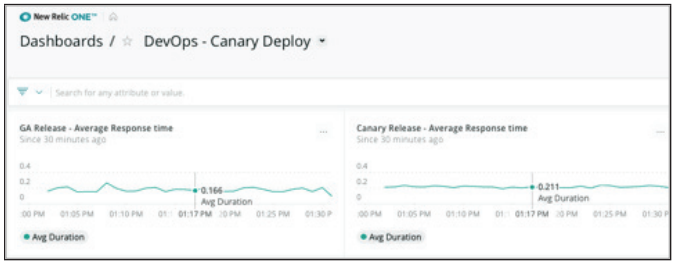
3. 지속적인 향상

데브옵스를 도입한 후, 복잡성을 제거하고 결과를 최적화해 높은 성과를 내는 데브옵스 조직으로 발전하려면, 지속적인 개선을 위한 문화와 프로세스가 필요합니다. DORA의 데브옵스 현황 보고서에서 높은 성과를 내는 기업들 수가 전년 대비 3 배 증가한 이유 중 하나는, 바로 지속적인 개선의 문화 때문입니다.

성능을 비즈니스 결과와 연관시키는 것에서부터 시작해야 합니다. 이를 통해 인프라 및 애플리케이션 변경이 궁극적으로 고객 경험과 비즈니스 지표에 어떤 영향을 미치는지를 확인할 수 있습니다. 비즈니스 성과를 측정 할 수 없다면, 배포가 성공적이었다고 주장하기 어렵습니다.

복잡성 속에서 데브옵스의 성과를 지속적으로 개선하는 다른 방법은 고급 릴리스 전략을 사용하는 것입니다.

- **부작용이 없는 샌드박스 활용:** 실험을 장려하는 데서 중요한 점은 위험을 감수하고 새로운 접근 방식과 기술을 시도하다 실패를 해도 책임을 묻지 않은 환경을 조성하는 것입니다.
- **기능 플래그 활성화:** 릴리스가 발생하면 기능 플래그가 뒤집힙니다. 상황이 예상대로 진행되지 않으면 뒤집힐 수 있습니다.
- **서비스 소비자와 소통:** 각 사업부는 운영 수준에서 서로를 소비자로 간주해야 합니다. 이는 새로운 버전과 이전 버전을 동시에 유지하는 기간이 필요하다는 의미합니다. 이로 인해 더 많은 오버 헤드가 발생하긴 하지만, 위험이 적고 고민할 거리가 없어진다는 장점이 생깁니다.
- **롤아웃 접근 방식 활용:** 더 위험한 기능을 사용하려면, 사용자 기반 전체에 롤아웃 방식을 사용합니다. 기능 플래그는 활성화/비활성화의 선택이 아니라 구체적인 수치를 측정하는 미터가 될 수 있습니다. (예: 사용자의 0% → 5% → 10% → 25% → 50% → 70% → 100%).
- **카나리 테스트 사용:** 카나리아 테스트는 일부 사용자에게 코드 변경 사항을 공개한 다음 대부분의 사용자가 아직 실행 중인 이전 코드와 비교하여 해당 코드의 성능이 어떤 차이가 있는지를 살펴보는 관행입니다. 카나리 서버 또는 컨테이너는 새 코드를 실행하고 새 사용자가 유입되면 이들 중 일부를 카나리 호스트로 전환합니다. 뉴렐릭은 새 코드가 올바르게 작동하는지 모니터링하고 측정하는 데 도움을 줄 수 있습니다.



카나리 배포 KPI

- **블루/그린 배포 채택:** 다운타임을 방지하고 문제가 심각하게 잘못되었을 때, 빠르게 롤백을 하려면 블루-그린 배포 접근 방식을 고려해야 합니다. 블루-그린 배포를 사용하면, 거의 동일한 두 개의 운영 환경(그린과 블루)이 생깁니다. 예를 들어, 블루 환경은 현재의 운영 환경이고 그린 환경은 최종 테스트를 수행하는 환경으로, 그린 환경에서 성공하면 실시간으로 사용자를 블루 환경으로 라우팅하기 시작합니다. 그러면, 블루 환경은 유휴 상태가 되지만, 필요한 경우 빠르게 롤백할 수있는 방법이 생깁니다.
- **다크 런칭 활용:** 다크 런칭은 카나리아 테스트와 유사하지만, 새 기능을 런칭한 후 코드가 이전 버전과 어떻게 비교되는지를 평가하는 것이 아니라, 소규모 사용자에게 먼저 릴리스를 해서 새 기능에 대한 사용자들의 응답을 평가합니다. 일반적으로 새 기능을 강조하지 않기 때문에 사용자들은 새로운 기능을 인식하지 못합니다. 그래서 '다크' 런칭이라고 합니다. 예를 들어, 웹 사이트에서 새로운 사용자 인터페이스를 다크 런칭하려면, 다른 대체 도메인이나 URL에서 런칭을 하거나 홈페이지의 숨겨진 IFrame에 포함시킵니다.

책임 전가 없음

책임를 묻지 않는 회고를 하는 목적은 모든 당사자가 인시던트로 이어진 상황, 그에 따른 사고 대응, 그리고 프로세스 개선을 위한 격차 또는 포인트를 이해하도록 만들기 위해서이며, 모두 문제의 재발을 방지하거나 적어도 완화하는 것을 목표로 합니다.

책임를 묻지 않는 회고를 하는 이유와 그 방법을 이 블로그 게시물에서 확인해보시기 바랍니다.

결론

데브옵스를 통해 여러움을 극복하는 조직들이 늘어나면서, 복잡성을 줄이고 데브옵스의 옵스 부분을 개선하는 데 집중하는 것이 필수가 될 것입니다. 옵저버빌리티, 즉 관찰성을 구축하고, 빠른 피드백 루프를 사용하며 지속적으로 개선함으로써, 팀은 더 높은 성과를 낼 수 있습니다.

뉴렐릭은 현대 애플리케이션 환경의 복잡성을 줄이는 데 도움을 줄 수 있습니다. 뉴렐릭은 AWS로 밀접하게 통합되어, EC2, 람다, 쿠버네티스 배포 등이 포함된 복잡한 AWS 환경을 손쉽게 관리할 수 있도록 해줍니다.

보다 자세한 정보를 원하시면 newrelic.co.kr/devops를 방문하시기 바랍니다.