aws modern apps

AWS 기반 컨테이너를 활용한 현대화

전 세계의 많은 기업이 디지털 트랜스포메이션을 추진하고 있습니다. 애플리케이션을 현대화하면 고객에게 더 나은 서비스를 제공하고 경쟁 환경에서 뒤처지지 않을 수 있습니다. AWS는 컨테이너를 구현하고 개발을 간소화하도록 문화의 변화를 이끌면서 기업의 현대화를 지원합니다. 이 eBook에서는 컨테이너화의 모범 사례와 AWS에서 컨테이너를 시작하는 방법을 살펴봅니다.

변화하는 환경에서 살아남기

지난 몇 년간, 전 세계에서 디지털 트랜스포메이션이 불붙듯 퍼져나갔습니다. 다양한 규모의 기업이 민첩성을 높이고 고객의 요구에 더 효과적으로 부응하기 위해 기술을 활용하는 방법을 찾고 있습니다. 변화하는 환경에서 살아남으려는 노력이 이 불에 기름을 부었습니다. 오늘날은 디지털 트랜스포메이션을 이루지 못하면 사양길에 접어들게 됩니다. 클라우드 네이티브 기업이 레거시 비즈니스를 크게 앞지르며 업계를 뒤흔드는 사례는 쉽게 찾아볼 수 있습니다. 많은 기업이 애플리케이션을 현대화하고 클라우드에서 자동화된 환경을 활용하는 것으로 디지털 트랜스포메이션의 첫걸음을 내딛고 있습니다. 현대화를 통해 기업은 다음과 같은 이점을 얻을 수 있습니다.

탄력성: 급증하는 고객 수요에 대응할 수 있는 능력

가용성: 언제 어디서나 고객의 요청을 처리할 수 있는 능력

민첩성: 문제를 신속하게 해결하거나 고객이 원하는 새로운 기능을 빠르게 배포할 수 있는 능력

디지털 트랜스포메이션을 지원하는 컨테이너

디지털 트랜스포메이션은 시간이 걸리는 일입니다. 하지만 이를 추진하는 원동력이 되는 것은 생산성 향상이라는 이점입니다. 기업의 현대화를 지원하는 많은 도구가 있지만, 컨테이너는 개발자가 애플리케이션을 효율적으로 패키징하고 배포하는 데 믿고 사용할 수 있는 솔루션으로 계속해서 모멘텀을 얻고 있습니다. 2020년 Cloud Native Computing Foundation(CNCF)의 설문 조사에 따르면 프로덕션에 컨테이너를 사용하는 비율이 2016년 이후로 300% 증가했다고 합니다. 오늘날 널리 사용되는 두 가지 컨테이너 기술인 도커와 Kubernetes의 도입률을 보면 이런 현상이 분명히 눈에 띕니다. 클라우드에서 호스팅하는 Kubernetes 워크로드 중 82%가 AWS에서 실행됩니다.[1]

개발 간소화를 위한 컨테이너

컨테이너는 어디서나 쉽게 애플리케이션을 실행하고 확장할 수 있도록 일관성을 지니고 가벼우면서 이동성이 좋은 소프트웨어 환경을 제공합니다. 애플리케이션은 초기에 테스트 환경에서 프로덕션 환경으로 이동하거나, 마이그레이션을 위해 온프레미스 가상 머신에서 클라우드로 이동하는 것과 같이 전체 수명 주기의 여러 다양한 환경에서 운영됩니다. 컨테이너를 사용하기 전에는 IT 팀이 새로운 각 환경의 호환성 제한을 고려하여 애플리케이션이 제대로 작동하도록 추가 코드를 작성해야 했습니다. 애플리케이션을 종속성, 구성 파일 및 인터페이스와 함께 패키징할 수 있는 컨테이너가 개발되면서 개발자는 서로 다른 호스트 간에 원활하게 이동하는 단일 이미지를 사용할 수 있게 되었습니다. 그에 따라 개발 팀은 인프라 관리와 같이 차별화 요소도 없으면서 업무 부담은 큰 작업에서 벗어날 수 있게 되었습니다.

개발자의 경우 컨테이너를 사용하면 서로 다른 환경의 호환성 요구 사항을 관리하는 데 시간을 소비하는 대신, 새로운 기능이나 최신 보안 항목을 추가하는 등 애플리케이션 구축에 집중할 수 있습니다.

또한 컨테이너는 기존 모놀리식 애플리케이션 아키텍처를 없애는 데 없어서는 안 되는 도구로, 마이크로서비스로 전환하여 더 쉽게 크기를 조정할 수 있도록 지원합니다. 마이크로서비스를 사용하면 각 애플리케이션 구성 요소가 하나의 서비스로 실행되므로 개발자가 다양한 측면에서 독립적으로 작업할 수 있습니다.

[1] *Nuclear Research 보고서, 2019년*

지속 가능한 개발 문화의 변화를 이끄는 컨테이너

컨테이너는 애플리케이션을 현대화하는 도구일 뿐만 아니라 개발 방식의 개선을 촉진하는 도구이기도 합니다. 컨테이너는 개발된 앱과 코드의 품질 관리에 대해 개발자가 주인 의식을 가질 수 있도록 함으로써 기존의 개발 방식을 완전히 바꾸어 놓았습니다. 예전에는 개발자가 애플리케이션 구축에만 집중하고 패키징과 배포의 성공에는 거의 관여하지 않았습니다. 컨테이너를 도입하고 나서부터는 '변화'를 경험하고 있습니다. 즉, 개발 프로세스뿐만이 아니라 최전방에서 품질을 관리하게 된 것입니다.

개발자에게 개발 작업의 결과를 책임질 수 있는 권한이 주어진 상태에서 다음 단계는 일찍 실패하고 실수로부터 배우는 문화를 조성하는 것입니다. 컨테이너화는 코드 통합과 배포의 자동화를 통해 이러한 문화를 조성하고 기업의 전반적인 민첩성을 향상시킵니다.

레거시 기업의 현대화

설립된 지 100년이 넘은 한 대기업은 해외 시장 중 한 곳에서 애플리케이션을 현대화할 기회를 포착했습니다. 이 회사의 개발 팀은 클라우드 기반 접근 방식을 통해 민첩성을 향상하고 더욱 빠르게 반복 테스트를 진행할 수 있었습니다. 컨테이너와 AWS에서 지원하는 자동화 환경을 활용하여 이 회사에서는 단 6주 만에 새로운 플랫폼에 대해 개념 단계에서 프로토타입 제작, 그리고 배포까지 완료할 수 있었습니다.

지금 바로 AWS 기반 컨테이너로 시작하기

AWS에서는 원활하게 컨테이너화를 진행하는 데 필요한 모든 솔루션을 제공합니다. 인프라 프로비저닝, 오케스트레이션, 보안, 네트워킹, 자동화 및 모니터링을 위해 검증된 도구를 사용하여 손쉽게 컨테이너를 시작할 수 있습니다.

프로비저닝

기본 인프라와 리소스의 원활한 프로비저닝

컨테이너를 실행하려면 기본 인프라를 프로비저닝해야 합니다. AWS는 원하는 관리 또는 자동화 정도에 따라 두 가지 솔루션을 제공합니다.

- 기본 인프라 프로비저닝을 자동화하려면 AWS Fargate를 사용합니다.
- 인프라의 컴퓨팅, 스토리지 및 네트워크 기능을 수동으로 정의하려면 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 사용합니다.

오케스트레이션

도커 또는 Kubernetes 컨테이너 크기 조정 및 관리

- Amazon Elastic Container Service(Amazon ECS)를 오케스트레이터로 사용하고 Amazon Elastic Container Registry(Amazon ECR)를 활용하여 도커 이미지를 저장 및 관리합니다.
- Amazon Elastic Kubernetes Service(Amazon EKS)를 도입하여 Kubernetes 컨테이너를 오케스트레이션합니다.

보안

컨테이너의 취약성 보호, 스캔 및 탐지

- AWS Identity and Access Management(IAM) 및 태깅, Amazon EC2 인스턴스의 보안 그룹, Amazon Virtual Private Cloud(VPC)를 사용하면 컨테이너를 보호할 수 있습니다.
- 이미지 스캔 솔루션은 도커 컨테이너 이미지의 취약성을 탐지합니다.

네트워킹 및 연결

여러 컨테이너에 애플리케이션 트래픽 분산

- AWS Elastic Load Balancing을 사용하여 여러 컨테이너와 서버리스 환경에 애플리케이션 트래픽을 분산합니다.
- AWS Global Accelerator와 AWS Elastic Load Balancing을 사용하여 컨테이너에서 실행되는 배포 애플리케이션 전체에 대해 트래픽의 경로를 지정합니다.
- AWS Global Accelerator와 AWS Elastic Load Balancing으로 애플리케이션 성능을 향상합니다.
- AWS App Mesh로 서비스 간 통신과 보안을 관리합니다.

자동화

CI/CD를 활용하여 자동으로 코드 배포

- AWS CodeCommit을 사용하여 소스 코드 리포지토리를 생성합니다.
- AWS CodePipeline을 사용하여 CI/CD 파이프라인을 구성합니다.
- AWS CodeBuild를 배포하여 컨테이너 이미지를 구축합니다.
- AWS App Runner를 사용하여 컨테이너화된 웹 애플리케이션을 구축, 배포 및 실행합니다.

관찰 및 모니터링

컨테이너에서 실행되는 서비스가 정상적으로 작동하고 통신하는지 확인

- AWS App Mesh를 배포하여 로깅, 지표 및 추적에 대한 가시성을 제공하고 로드 밸런싱 및 트래픽 성형을 지원합니다.
- 도커 컨테이너 이미지에 대한 상태 확인을 실행하여 컨테이너가 실행되고 있는지, 앱이 작동하는지 등을 확인합니다.
- Amazon CloudWatch Application Insights를 사용하여 Amazon ECS, Amazon EKS 또는 Amazon EC2의 Kubernetes에 배포된 컨테이너에서 실행되는 애플리케이션의 상태를 모니터링합니다.

프로비저닝

AWS는 Amazon EC2와 AWS Fargate를 사용하여 인프라를 프로비저닝하는 옵션을 제공합니다. 두 도구의 주요 차이점은 컨테이너 애플리케이션을 실행하는 기본 인프라에 대한 관리 및 제어 수준이 다르다는 것입니다.

- **AWS Fargate**: Fargate에서는 서버 또는 클러스터를 관리하지 않고도 컨테이너를 실행할 수 있습니다. 컨테이너에 애플리케이션을 패키징하고, CPU 및 메모리 요구 사항을 지정하고, 네트워킹 및 IAM 정책을 정의하고, 애플리케이션을 시작하기만 하면 됩니다.

- **Amazon EC2**: 보다 전통적인 방식인 Amazon EC2 시작 유형을 사용하면 컨테이너를 실행하기 위해 인스턴스를 불러오고 컨테이너 애플리케이션을 실행하는 인프라를 서버 수준에서 보다 세밀하게 제어할 수 있습니다.

오케스트레이션

애플리케이션이 컨테이너화되었으면, 다음 단계는 프로덕션 단계에서 컨테이너를 실행하는 것입니다. 아키텍처의 크기를 조정하려면 오케스트레이션 도구가 필요합니다. AWS는 온프레미스와 클라우드에서 모두, 기업의 요구에 맞는 오케스트레이션 플랫폼을 제공합니다.

Amazon ECS와 Amazon ECS Anywhere

Amazon Elastic Container Service(Amazon ECS)를 사용하면 AWS에서 컨테이너화된 워크로드를 손쉽게 배포할 수 있습니다. 단순하면서 강력한 Amazon ECS를 통해 하나의 도커 컨테이너에서 전체 엔터프라이즈 애플리케이션 포트폴리오 관리까지 확장할 수 있습니다. 제어 플레인이나 노드를 복잡하게 관리하지 않고도 클라우드와 온프레미스의 여러 가용 영역에 있는 컨테이너 워크로드를 실행하고 크기를 조정할 수 있습니다. Amazon ECS는 자동으로 관리되며 무제한으로 확장 가능한 제어 플레인을 제공합니다. 이러한 오케스트레이션 도구는 자체적인 운영 체제를 사용하거나 자체 인프라에 대한 제어권을 원하는 기업에 유용합니다. AWS Copilot은 고객이 AWS에서 컨테이너화된 애플리케이션을 빠르게 시작하고 쉽게 관리할 수 있도록 하는 명령줄 인터페이스(CLI)입니다. AWS Copilot은 고객이 빠르게 배포할 수 있도록 예제와 가이드를 포함하는 단순한 선언형 명령 세트를 제공합니다. 서비스를 프로덕션 상태로 실행하기 위해서는 AWS Copilot과 AWS 계정, 코드만 있으면 됩니다.

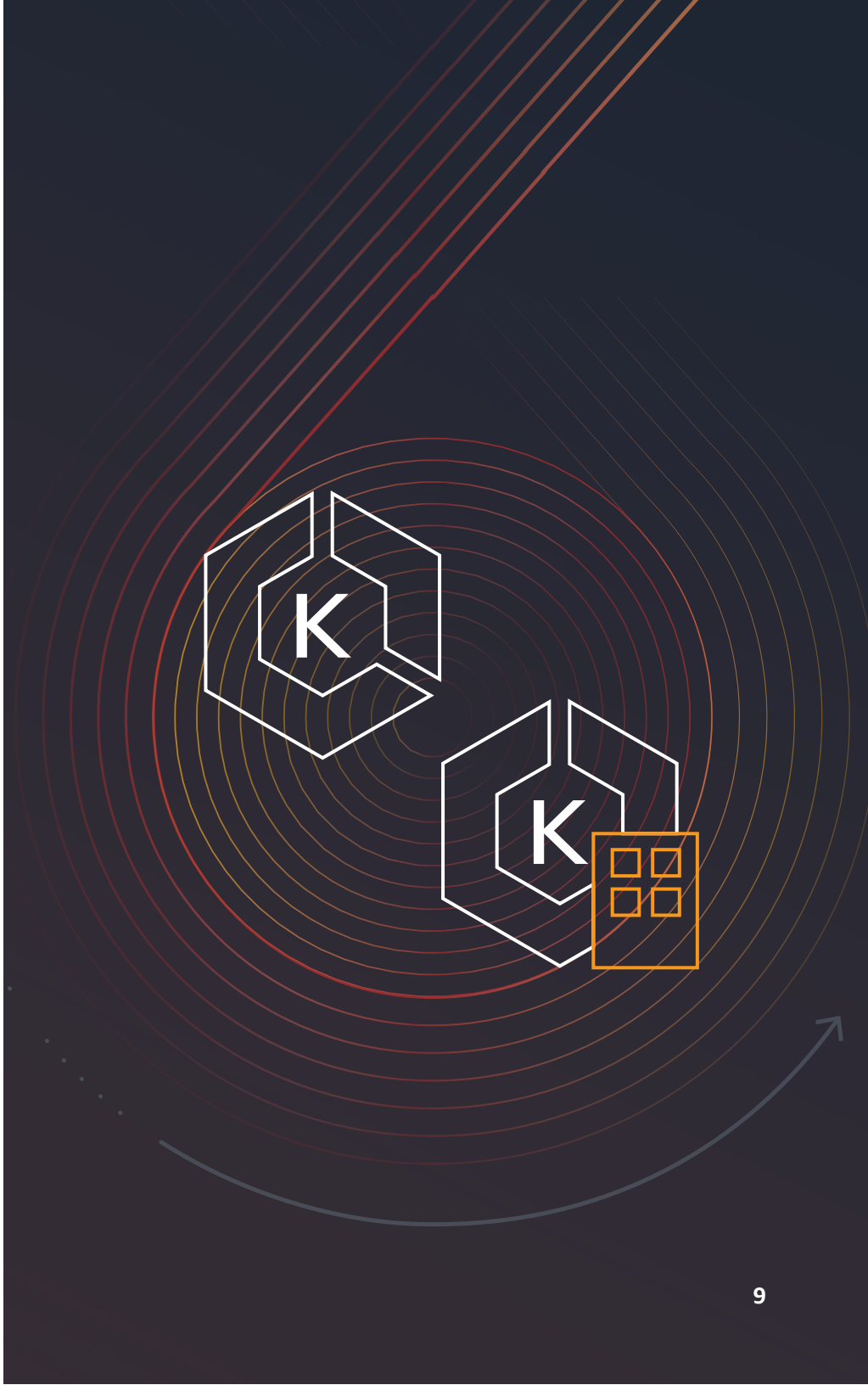
Amazon ECS Anywhere를 사용하면 친숙한 Amazon ECS 콘솔과 운영자 도구로 온프레미스 컨테이너 워크로드를 관리하고 컨테이너 기반 애플리케이션 전체에서 일관된 경험을 얻을 수 있습니다. AWS Systems Manager(구 SSM) 통합을 통해 온프레미스 하드웨어와 AWS 제어 플레인 간에 자동으로 안전하게 신뢰성을 확보할 수 있습니다.

Amazon EKS와 Amazon EKS Anywhere

Kubernetes는 대규모로 컨테이너를 실행할 수 있도록 구축된 오픈 소스 컨테이너 관리 플랫폼으로 빠르게 성장하고 있습니다. Amazon EKS를 사용하면 복수의 AWS 가용 영역에서 Kubernetes 관리 인프라를 실행할 수 있습니다.

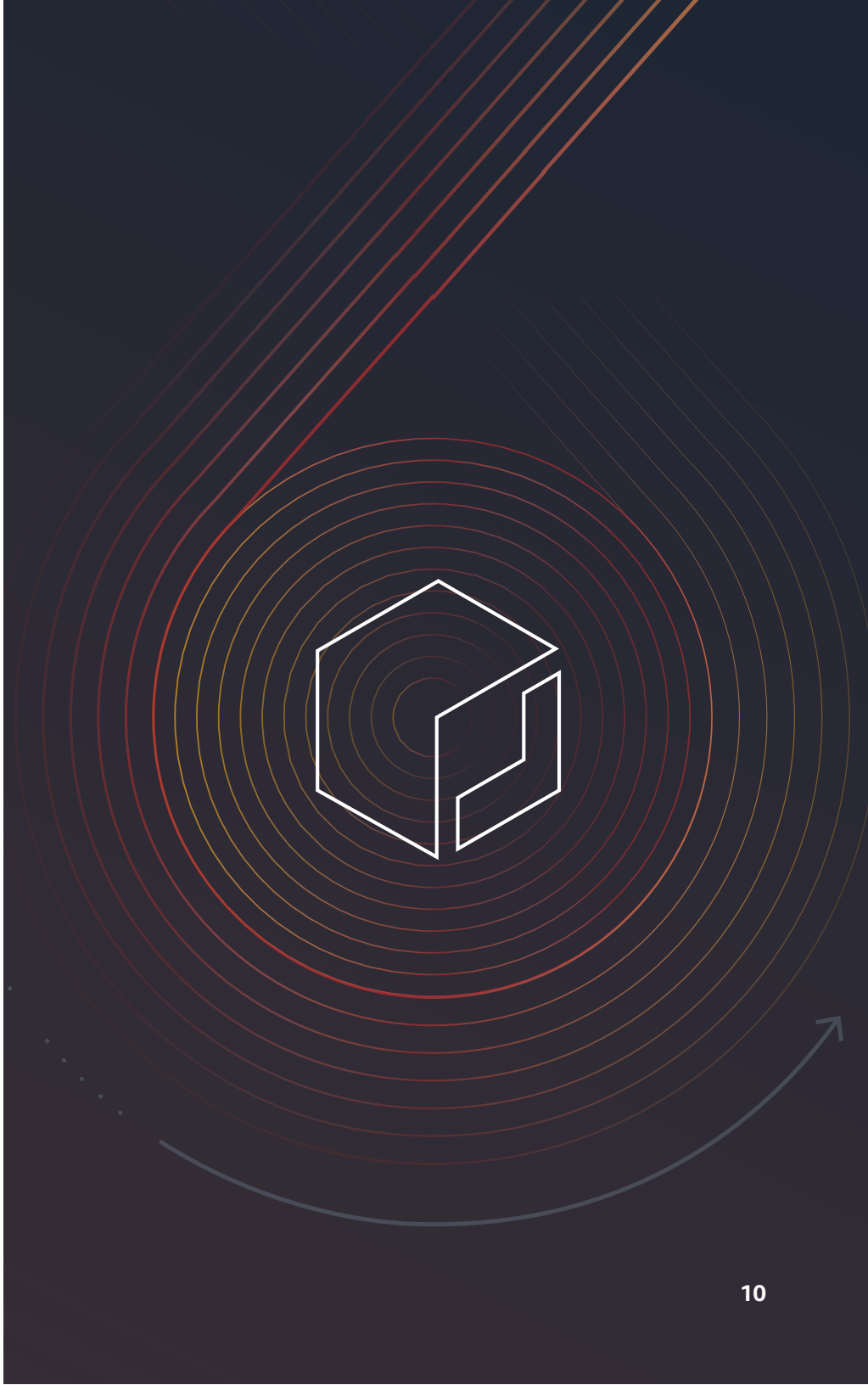
많은 AWS 고객이 AWS에서 Kubernetes를 실행하면 막대한 이점을 누릴 수 있다고 생각합니다. Amazon EKS는 컨테이너 일정 관리, 애플리케이션 가용성 관리, 클러스터 데이터 저장 및 기타 주요 태스크에 책임이 있는 Kubernetes 제어 플레인 노드의 가용성과 확장성을 자동으로 관리합니다. 컨테이너에 서버리스 컴퓨팅을 제공하는 Amazon EC2 또는 AWS Fargate에서 모두 Kubernetes 애플리케이션을 실행할 수 있습니다.

Amazon EKS Anywhere를 사용하면 자체 가상 머신(VM)과 베어 메탈 서버 등을 포함하여 온프레미스에서 Kubernetes 클러스터(Amazon EKS Distro의 소프트웨어로 구축)를 손쉽게 만들고 운영할 수 있습니다. Kubernetes 클러스터를 관리하기 위해 자체적으로 도구를 구축하고 지원해야 하는 복잡함을 EKS Anywhere를 통해 해소할 수 있습니다. EKS Anywhere는 로깅, 모니터링, 네트워킹, 스토리지에 대한 기본 구성을 통해 클러스터 생성과 관리를 간소화하고 베어 메탈, vSphere, 클라우드 가상 머신과 같은 인프라에서의 운영을 간소화하는 자동화 도구를 제공합니다. 동시에 프로덕션에서 Kubernetes를 실행할 때 필요한 클러스터 설치 및 수명 주기 관리, 관찰, 클러스터 백업, 정책 관리와 같은 편향적 도구와 추가 구성 요소 또한 제공합니다. Amazon EKS Distro는 AWS의 EKS에서 사용되는 동일한 오픈 소스 Kubernetes 소프트웨어 배포를 패키징하여 고객이 자체 온프레미스 인프라에서 사용할 수 있도록 합니다. EKS Distro 클러스터는 고객의 자체 도구 또는 Amazon EKS Anywhere로 관리할 수 있습니다.



Amazon ECR

도커 컨테이너를 사용하려면 먼저 도커 이미지가 필요합니다. 이 이미지는 컨테이너 인스턴스를 생성하는 블루프린트 역할을 합니다. Amazon ECR은 도커 컨테이너 이미지를 쉽게 저장, 관리 및 배포할 수 있도록 하는 완전관리형 도커 컨테이너 레지스트리입니다. Amazon ECR은 Amazon ECS와 통합되어 개발에서 프로덕션까지의 워크플로를 간소화할 수 있습니다. Amazon ECR을 사용하면 자체 컨테이너 리포지토리를 운영할 필요가 없으며 기본 인프라 확장에 대한 걱정도 필요 없습니다.

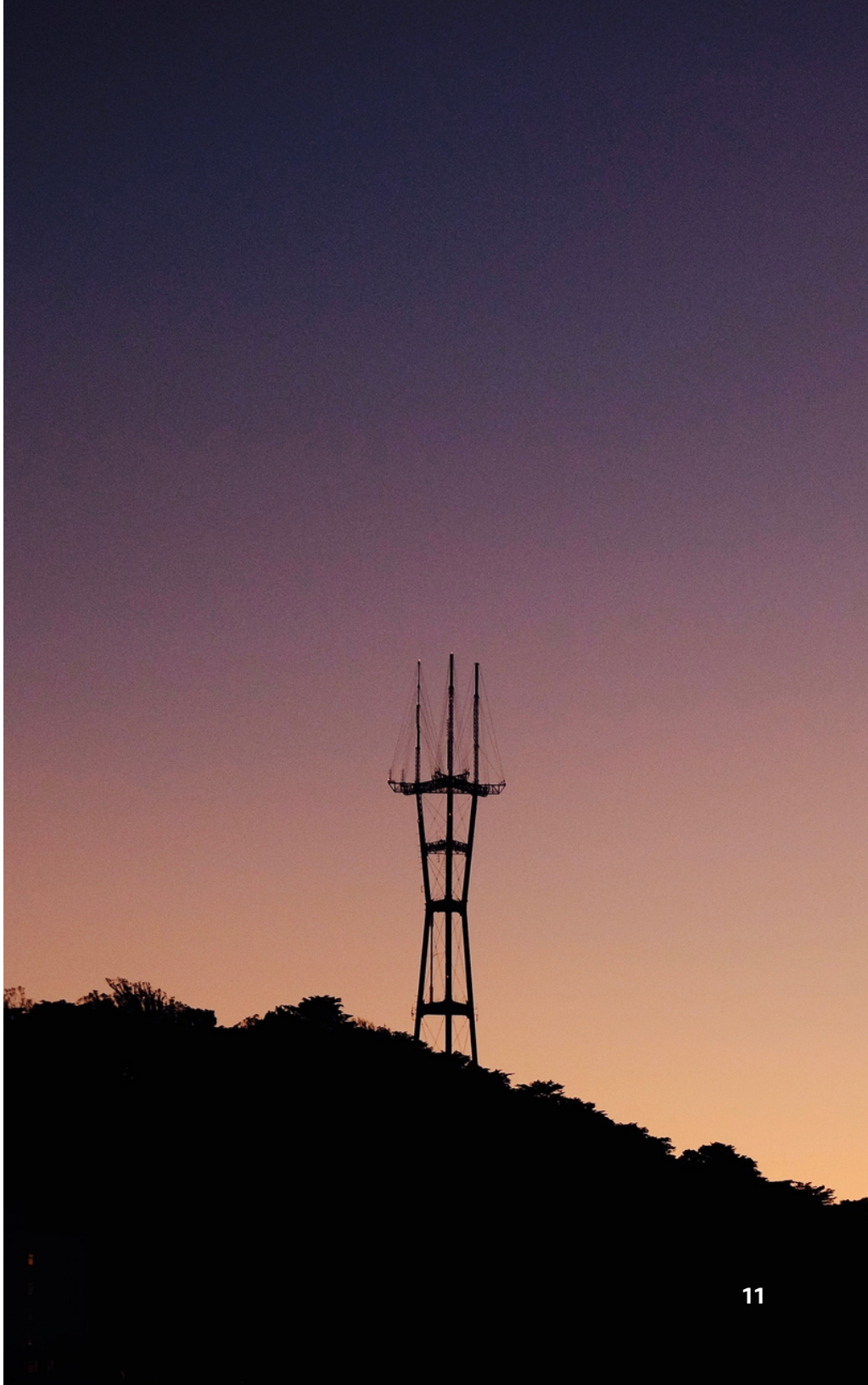


보안

품질 관리와 마찬가지로 보안 고려 사항도 개발 주기의 초기 단계에서부터 확인하는 것으로 바뀌었습니다. 자율성이 강화되어 개발자는 최신 보안 위험에 대응하기 위해 직접 작성한 코드를 더욱 손쉽게 추가할 수 있습니다. 그러나 보안은 조직 전체에서 우선시되어야 합니다. 보안을 우선으로 하는 문화로 변화하기 위해서는 보안 작업을 최대한 투명하게 진행하고 보안 모범 사례와 도구를 고려하면서 아키텍처를 미리 정의해야 합니다.

AWS는 컨테이너 액세스 권한을 제어하는 데 사용할 수 있는 다양한 도구를 제공합니다. AWS IAM을 사용하여 누가 인증(로그인)되었고 리소스를 사용하도록 승인(허가)되었는지 결정할 수 있습니다. Amazon VPC를 사용하면 직접 정의한 가상 네트워크에서 컨테이너 작업(도커) 또는 Pod(Kubernetes)를 논리적으로 격리할 수 있습니다. 보안 그룹을 정의하여 EC2 인스턴스 간에 가상 방화벽을 만들 수도 있습니다.

이미지 스캔 솔루션을 사용하면 컨테이너 이미지의 취약성 또는 이미지 종속성을 탐지할 수 있습니다. 보안 팀은 이러한 컨테이너 또는 이미지 종속성을 스캔한 다음 사전 승인된 리소스를 게시하여 개발자가 안심하고 사용하도록 할 수 있습니다.



네트워킹 및 연결

애플리케이션이 실행되기 시작하면 최종 사용자가 중단 없이 애플리케이션을 사용할 수 있도록 트래픽이 컨테이너 전체에 분산되도록 해야 합니다.

- **Elastic Load Balancing**을 사용하면 AWS 컨테이너 서비스에 기본적으로 통합되어 있는 정교한 로드 밸런싱 알고리즘이 작동하여 사용자가 쉽게 애플리케이션에 액세스할 수 있습니다.

- **Amazon Global Accelerator**는 애플리케이션이 우수한 성능을 내고 전 세계 사용자의 액세스를 허용하며 사용자와 가장 가까운 AWS 리전에서 서비스될 수 있도록 지원합니다.

- 컨테이너 서비스 간의 통신을 관리하기 위해서 **AWS App Mesh**를 사용하면 세분화된 보안 기능 및 가시성을 얻을 수 있습니다.

자동화

지금부터 2024년까지 하이브리드 통합 기능에 대한 수요가 높아지는 동시에 대부분의 PaaS 제품 및 서비스의 경쟁 구도가 점차 더 강화될 것입니다.[2] Amazon EKS Anywhere와 Amazon ECS Anywhere를 사용하여 기업이 소유한 인프라에서 워크로드를 실행함으로써 단순하고 친숙한 도구를 활용하면서 규정을 준수하고 데이터 중력 및 그 외 비즈니스 요구 사항도 충족할 수 있습니다. 로컬 제어 플레인을 설치하거나 운영하지 않고, 동일한 하이퍼스케일의 신뢰할 수 있는 완전관리형 제어 플레인을 온프레미스 컨테이너 워크로드에 사용할 수 있습니다. 기업의 자체 하드웨어 엣지 로케이션에서 컨테이너화된 데이터 처리 워크로드를 실행하면 최종 고객과 가까운 거리를 유지하면서 대기 시간 또한 줄일 수 있습니다.

자동화된 환경에서는 코드 배포 작업을 수동으로 수행할 필요가 없습니다. 인프라에 수백 또는 수천 개의 컨테이너가 포함되어 있는 경우, 지속적 통합 및 지속적 전달(CI/CD)을 통해 자동화하면 인적 오류의 위험을 최소화하면서 더 빠르게 확장하고 대응할 수 있습니다. AWS CodeCommit, AWS CodePipeline 및 AWS CodeBuild를 사용하면 소스 코드 리포지토리를 생성하고, CI/CD 파이프라인을 구성하고, 컨테이너 이미지를 구축할 수 있습니다.

완전관리형 애플리케이션 서비스인 AWS App Runner를 사용하면 개발자는 클릭 몇 번으로 손쉽게 컨테이너화된 웹 애플리케이션과 API를 대규모로 배포하고 실행할 수 있습니다. 즉, 인프라를 구성하고 관리할 필요가 없게 됩니다. 소스 코드와 컨테이너 이미지 또는 배포 파이프라인을 제공하기만 하면 App Runner가 웹 애플리케이션을 구축 및 배포하고, 네트워크 트래픽을 로드 밸런싱하고, 수요에 맞게 용량을 확장하거나 축소하며, 애플리케이션 상태를 모니터링할 뿐만 아니라 기본적으로 트래픽을 암호화합니다. 또한, 이전에 컨테이너를 사용한 경험이 없더라도 App Runner를 사용하여 컨테이너의 이동성, 효율성, 비용 절감이라는 장점을 누릴 수 있습니다.

[2] *Gartner Forecast 분석, 2021년 2월*

관찰 및 모니터링

컨테이너를 사용하여 마이크로서비스를 배포하고 나면 컨테이너 상태를 모니터링하고 서비스가 정상적으로 서로 통신하는지 확인해야 합니다.

- **AWS App Mesh**를 사용하면 다양한 유형의 컴퓨팅 인프라에 구축된 서비스에 대해서 일관된 가시성을 확보할 수 있고 네트워크 트래픽을 제어할 수 있으므로 서비스를 쉽게 실행할 수 있습니다. App Mesh를 사용하면 모니터링 데이터가 수집되는 방법이나 서비스 간에 트래픽이 라우팅되는 방법을 변경하기 위해 애플리케이션 코드를 업데이트할 필요가 없습니다. App Mesh는 모니터링 데이터를 내보내도록 각 서비스를 구성하고 애플리케이션 전체에 걸쳐 일관된 통신 제어 로직을 구현합니다. 따라서 오류가 발생하거나 코드 변경 사항을 배포해야 할 때 정확한 오류 위치를 신속하게 찾아내고 네트워크 트래픽을 자동으로 재라우팅할 수 있습니다. App Mesh는 오픈 소스 Envoy 프록시를 사용하므로 광범위한 AWS 파트너 및 오픈 소스 도구와 호환됩니다.

- 도커 컨테이너 이미지를 사용하면 컨테이너와 애플리케이션이 정상인지 확인할 수 있습니다. 간단한 상태 확인 명령으로 도커 파일은 컨테이너를 검사하여 작동 중인지 여부를 확인합니다. 이 프로세스는 서버 프로세스가 계속 실행 중임에도 불구하고 웹 서버가 무한 루프에 갇혀 새로운 연결을 처리할 수 없는 경우를 탐지할 수 있습니다.

- **Amazon CloudWatch Application Insights**는 컨테이너 모니터링을 지원합니다. AWS에서 실행되는 **Amazon ECS**, **Amazon EKS**, **EC2의 Kubernetes** 컨테이너에 배포된 애플리케이션의 모니터링, 경보, 대시보드 등을 설정할 수 있습니다. 이제 모니터링 티어 옵션을 사용하여 AWS의 컨테이너에서 실행되는 애플리케이션의 상태를 모니터링하는 지표, 텔레메트리, 로그를 확인할 수 있습니다.

컨테이너를 활용할 준비가 되셨나요?

AWS와 AWS의 대규모 파트너 에코시스템은 기업의 현재 현대화 여정 단계에 맞춰 필요한 도구를 제공하므로 바로 시작하실 수 있습니다.

언제든지 AWS 영업 담당자, 또는 선호하시는 AWS 파트너에게 문의하시기 바랍니다.

자세한 내용은 aws.amazon.com/containers에서 확인하세요.

aws modern apps