

AWSKRUG CONTAINER

출석체크 부탁드립니다!



PRESS ← → OR SPACE

AWSKRUG CONTAINER

2025 AWS re:Invent Recap - Container

저는 이렇게 봤습니다 👀

Research: Gemini Deep Research | Development: Codex | Editing: Claude Sonnet 4.5

Compiled & Presented by 임지훈 (AWS Ambassadors & Community Builder)

AWSKRUG CONTAINER

오늘 다룰 서비스들

- Amazon ECR Enhancements (Managed Signing, Archive Storage)
- Amazon ECS Express Mode
- AI-powered experience for ECS/EKS
- Fully managed MCP servers on ECS/EKS
- EKS Provisioned Control Plane
- EKS Capabilities: Argo CD
- EKS Capabilities: ACK & KRO
- Enhanced Network Observability

AWSKRUG CONTAINER

Amazon ECR Enhancements

소프트웨어(Software Supply Chain) 공급망 보안 기능 강화

Archive Storage Class

보안 사고 대응, 규제 준수

'리포지토리 이미지 수 제한'이라는 물리적 제약을 해소함으로써 무제한에 가까운 데이터 보존 가능

Managed Image Signing

신뢰성, 키 관리 사고 차단

개인 키(Private Key) 관리의 부담을 클라우드 제공자에게 이양, '기본적으로 안전한(Secure by Default)' 배포 파일 프라인 구축 가능

AWSKRUG CONTAINER

Amazon ECR Enhancements 1

Archive Storage Class

- 장기 이미지 보관 비용 절감 (서울 리전 기준, 150TB 이상일 경우 GB당 \$0.1 -> \$0.07)
- 복원 지연 고려한 운영 정책 필요 (활성화 시, 약 20분 이내)
- [라이프사이클 정책](#)으로 자동 아카이빙
- 아카이빙 된 이미지는 이미지 수 제한(10,000개)에 포함되지 않음
- Lifecycle Policy의 지원 범위는 확장되는 중([#2728](#))

주요 특징 요약

구분	Standard	Archive
사용성	즉시 사용 가능	복원 필요 (약 20분)
쿼터	10,000개 제한 포함	제한에서 제외
메타데이터	유지	유지(describe, list 가능)
적합 용도	활성 이미지	장기 보관

AWSKRUG CONTAINER

Amazon ECR Enhancements 2

기존의 클라이언트 기반 서명 방식(예: Notation)은 운영 장벽(직접 PKI 구축 및 생성, 보관, 로테이션) 존재

Managed Image Signing

자체 공개 키 인프라(PKI)를 유지 관리할 필요 없이 Amazon ECR 리포지토리에 푸시되는 컨테이너 이미지에 디지털 서명 지원

- AWS Signer 통합 관리형 서명
- 서명 검증으로 미승인 이미지 배포 차단

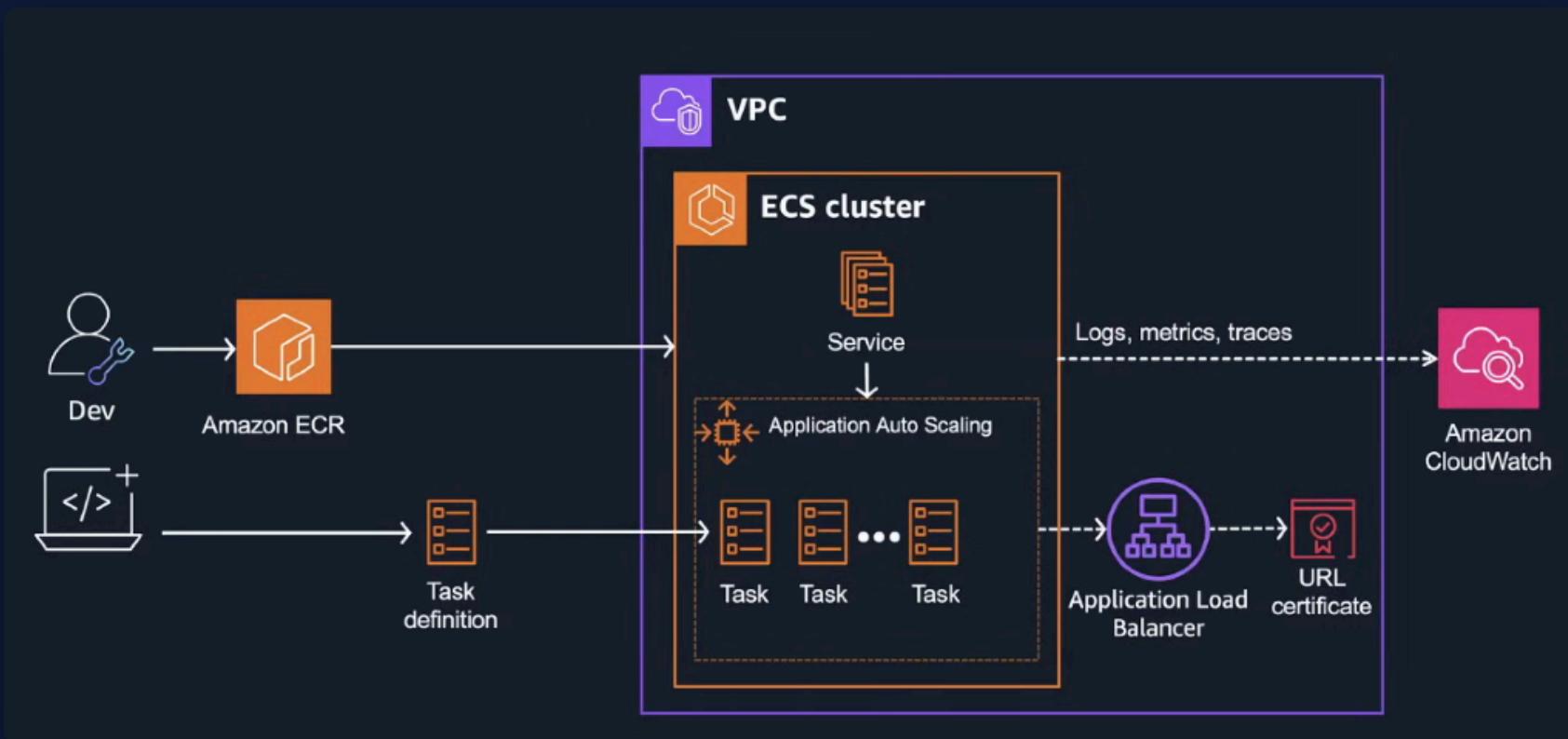
KMS 암호화와는 어떻게 다른가요?

고려사항

- 비용 : 서명 생성 건당 \$0.02의 비용이 부과, `release-*` 태그가 붙은 프로덕션 후보 이미지에만 규칙을 적용하는 필터링 전략 필요
- 쿼터 : 생성된 서명 아티팩트도 리포지토리의 이미지 수 제한(10,000개)에 포함. [이미지 1개당 서명 1개가 생성](#), 실질적인 이미지 저장 용량이 절반으로 감소

AWSKRUG CONTAINER

Traditional ECS



* 이미지 : [Amazon ECS Express Mode 컨테이너 이미지 손쉽게 배포하기](#)

AWSKRUG CONTAINER

Amazon ECS Express Mode

주요 특징

ECS 초기 설정의 진입장벽을 제거하여 "배포까지의 시간"을 단축합니다.

- 클러스터/서비스/ALB/보안그룹/IAM까지 자동 오케스트레이션
- Fargate 기반으로 EC2 관리 불필요
- 호스트 헤더 기반 리스너 규칙을 사용하여 최대 25개의 ECS 서비스에서 자동으로 공유 > 비용 효율

AWS Elastic Beanstalk, AWS App Runner와는 어떻게 다른가요?

PaaS(Platform as a Service)는 생성된 리소스에 대한 직접적인 접근이나 세밀한 제어가 제한됩니다.

ECS Express Mode는 인프라를 자동 구성하지만, 결과물(ALB, 타겟 그룹, 보안 그룹 등)을 사용자 계정에 투명하게 생성합니다.

그래서 사용자는 생성된 리소스를 언제든지 확인하고 필요 시 직접 수정할 수 있습니다.

AWSKRUG CONTAINER

Amazon ECS Express Mode

시작하기

The screenshot shows the AWS Elastic Container Service (ECS) Express Mode setup interface. The left sidebar contains navigation links for Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. A feedback link 'Tell us what you think' is also present.

The main content area is titled 'Express Mode' and describes it as a way to deploy highly available, scalable, containerized applications with simplified configurations. It automatically sets up the necessary AWS services and resources.

The 'Let's set up your app' section includes:

- Image URI:** A text input field containing a placeholder URL ending in '.amazonaws.com/demo/hello-world@sha256:b584bc5'. To its right is a blue 'Browse ECR images' button.
- Private registry - optional:** A checkbox labeled 'Private registry authentication' which is currently unchecked.
- Task execution role:** A dropdown menu showing 'ecsTaskExecutionRole' with a blue circular icon and a 'Create new role' button.
- Infrastructure role:** A dropdown menu showing 'ecsInfrastructureRole' with a blue circular icon and a 'Create new role' button.

Below these fields is a section titled 'Additional configurations - optional'.

At the bottom right of the form are two buttons: 'Cancel' and 'Create'.

AWSKRUG CONTAINER

Amazon ECS Express Mode

✓ Scaling policy: [agents-md-bc7d](#)  Active

Last updated at February 2, 2026, 16:03 (UTC+9:00).

⚠ Service: [agents-md-bc7d](#)  Active

service [agents-md-bc7d](#) was unable to place a task. Reason: CannotPullContainerError: pull image manifest has been retried 7 time(s): image Manifest does not contain descriptor matching platform 'linux/amd64'.

✖ Deployment [4X0k15nVPGr_8vKz5-eeM](#)  rollback has failed.

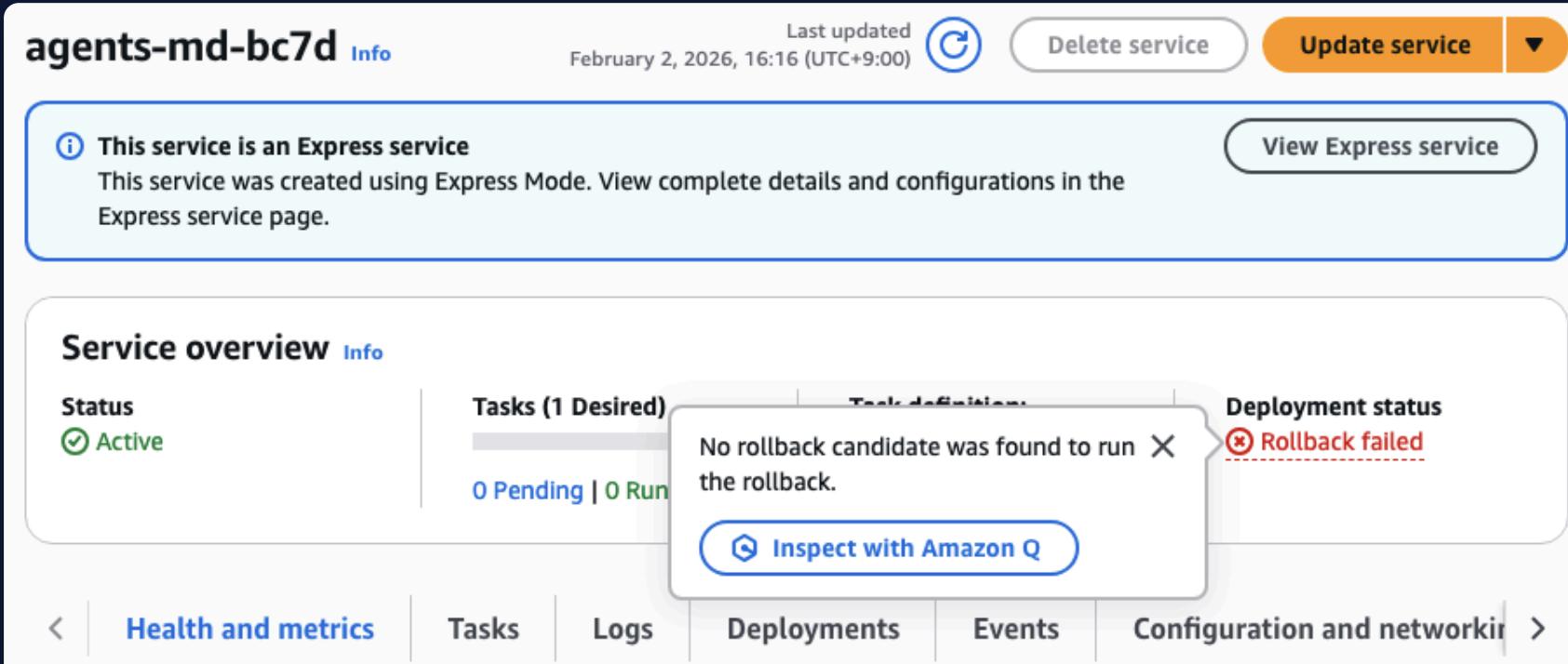
No rollback candidate was found to run the rollback.

- Express Mode Additional configurations에 운영 체제/아키텍처 선택 옵션 미제공

AWSKRUG CONTAINER

Amazon ECS enhanced AI-powered troubleshooting in the Console

Troubleshooting with Amazon Q Developer



The screenshot shows the AWS ECS Service Overview page for a service named "agents-md-bc7d".

Service Information: Last updated February 2, 2026, 16:16 (UTC+9:00). Actions: Delete service, Update service.

Info Panel: This service is an Express service. This service was created using Express Mode. View complete details and configurations in the Express service page. View Express service.

Service overview: Status: Active. Tasks (1 Desired): 0 Pending | 0 Run. Deployment status: Rollback failed. A tooltip indicates: No rollback candidate was found to run the rollback. Inspect with Amazon Q.

Navigation: Health and metrics (selected), Tasks, Logs, Deployments, Events, Configuration and networking.

AWSKRUG CONTAINER

Amazon ECS enhanced AI-powered troubleshooting in the Console

Root Cause Analysis

- 문제 발생 경로와 메커니즘 상세 설명
- 왜 이 문제가 발생했는지 컨텍스트 제공

Resolution Recommendations

- 단계별 해결 가이드
- 다양한 해결 옵션 제시
- 각 방안의 장단점 설명

Documentation & Reference

- AWS 공식 문서 링크
- 관련 베스트 프랙티스 가이드
- 유사 사례 및 해결 패턴

The screenshot shows a troubleshoot step titled "ECS Service Deployment Architecture Mismatch". It includes a "ROOT CAUSE ANALYSIS" section with a primary cause of "Container Image Architecture Mismatch" and evidence of deployment failure due to incompatible architecture. It also lists five key pieces of evidence: Service Status, Deployment Status, Container Pull Error, Task Definition Configuration, and Image Reference.

Amazon Q

ECS Service Deployment Architecture Mismatch

ROOT CAUSE ANALYSIS

Primary Root Cause: Container Image Architecture Mismatch

The deployment failure is caused by a **container image architecture incompatibility**. The specific error "manifest not compatible with linux/amd64" indicates that the container image stored in ECR was built for a different CPU architecture than what AWS Fargate requires.

Key Evidence:

1. **Service Status:** ROLLBACK_FAILED with reason "No rollback candidate was found to run the rollback"
2. **Deployment Status:** FAILED with "ECS deployment circuit breaker: tasks failed to start"
3. **Container Pull Error:** "Failed to pull container image - manifest not compatible with linux/amd64"
4. **Task Definition Configuration:** Specifies X86_64 architecture for Fargate
5. **Image Reference:** Uses ECR image with SHA256 digest:

AWSKRUG CONTAINER

Fully managed Amazon ECS MCP server (preview)

Amazon ECS MCP 서버는 CLI 및 IDE에서 지능형 AI 에이전트 상호 작용을 지원할 뿐만 아니라 Amazon Q를 통해 Amazon ECS 콘솔 내에서 직접 지능형 문제 해결 환경을 제공합니다. ([2025.11.21](#))

- 관리형 ECS MCP 서버를 사용하면 설정이 간소화되고 자동 업데이트가 제공됩니다.
- 호스팅 서비스를 통해 로컬 설치가 필요 없으며 AWS IAM 통합을 통해 엔터프라이즈급 보안을 제공합니다.

Amazon ECS MCP Server

awslabs.ecs-mcp-server 0.1.1

```
pip install awslabs.ecs-mcp-server==0.1.1
```



사용 가능한 최신 버전
(0.1.25)



릴리스 날짜: 2025년 5월 30일

AWSKRUG CONTAINER

Concepts of MCP

MCP는 클라이언트-서버 아키텍처를 따르며, Kiro와 같은 AI 애플리케이션이 MCP 호스트가 하나 이상의 MCP 서버에 연결을 설정합니다. MCP 호스트는 각 MCP 서버에 대해 하나의 MCP 클라이언트를 생성함으로써 이를 구현합니다. 각 MCP 클라이언트는 해당 MCP 서버와 전용 연결을 유지합니다.

MCP 아키텍처 구성 요소

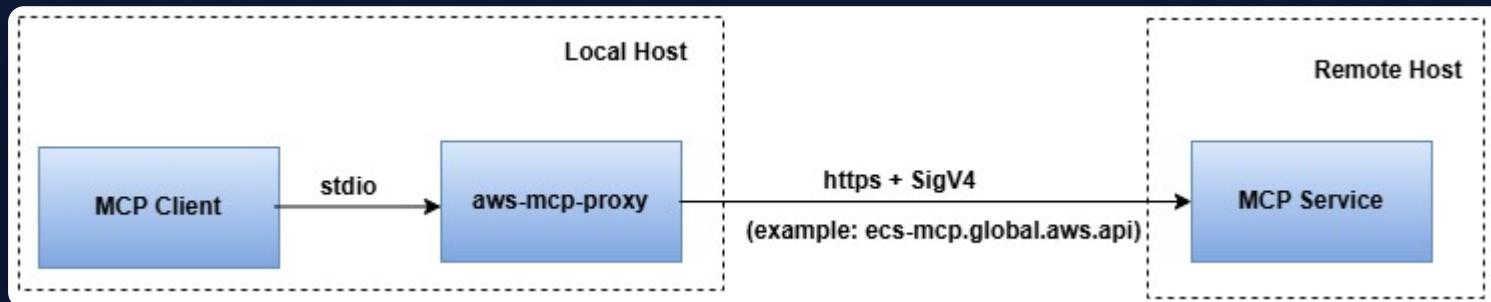
- MCP 호스트 : 하나 이상의 MCP 클라이언트를 조정하고 관리하는 AI 애플리케이션
- MCP 클라이언트 : MCP 서버와의 연결을 유지하고 MCP 호스트가 사용할 수 있도록 MCP 서버로부터 컨텍스트 정보를 가져오는 구성 요소
- MCP 서버 : MCP 클라이언트에 컨텍스트를 제공하는 프로그램

AWSKRUG CONTAINER

Model Context Protocol(MCP) Proxy for AWS

MCP Proxy for AWS

- 공식 MCP 사양은 OAuth 기반 인증을 지원하지만, AWS의 MCP 서버는 AWS IAM 인증(SigV4)도 사용
- 표준 MCP 클라이언트는 AWS 자격 증명으로 요청에 서명하는 방법을 알지 못함



- 클라이언트 측 프록시는 MCP 클라이언트가 AWS SigV4 인증을 사용하여 AWS에서 호스팅되는 원격 MCP 서버에 연결할 수 있도록 지원
- Amazon Q, Kiro, Cursor와 같은 인기 있는 에이전트 AI 개발 도구와 Strands Agents와 같은 인기 있는 에이전트 프레임워크를 지원

AWSKRUG CONTAINER

MCP server configuration

Hosted MCP Server

```
{  
  "mcpServers": {  
    "ecs-mcp": {  
      "disabled": false,  
      "type": "stdio",  
      "command": "uvx",  
      "args": [  
        "mcp-proxy-for-aws@latest",  
        "https://ecs-mcp.{region}.api.aws/mcp",  
        "--service",  
        "ecs-mcp",  
        "--profile",  
        "{profile}",  
        "--region",  
        "{region}"  
      ]  
    }  
  }  
}
```

Local MCP Server

```
{  
  "mcpServers": {  
    "awslabs.ecs-mcp-server": {  
      "command": "uvx",  
      "args": [  
        "--from",  
        "awslabs-ecs-mcp-server",  
        "ecs-mcp-server"  
      ],  
      "env": {  
        "AWS_PROFILE": "your-aws-profile",  
        "AWS_REGION": "your-aws-region",  
        "FASTMCP_LOG_LEVEL": "ERROR",  
        "FASTMCP_LOG_FILE": "/path/to/ecs-mcp-server.log",  
        "ALLOW_WRITE": "false",  
        "ALLOW_SENSITIVE_DATA": "false"  
      }  
    }  
  }  
}
```

AWSKRUG CONTAINER

Amazon ECS MCP Server

(Recommended) Hosted MCP Server  [Tools](#)

Local MCP Server (Legacy)  [Tools](#)

고려사항

ECS MCP 서버는 현재 개발 중이며 다음과 같은 환경을 위해 설계되었습니다.

- 개발 및 프로토타이핑 : 로컬 애플리케이션 개발, 컨테이너화 방식 테스트, 배포 구성의 신속한 반복 작업에 이상적입니다.
- 학습 및 탐구 : 컨테이너화, ECS 및 AWS 인프라에 대해 배우고자 하는 사용자에게 매우 유용합니다.
- 테스트 및 스테이징 : 중요하지 않은 환경에서의 통합 테스트 및 사전 운영 검증에 적합합니다.

다음과 같은 경우에는 권장하지 않습니다.

- 현재 개발 중이므로 운영 환경 배포 또는 비즈니스 핵심 애플리케이션 및 민감한 데이터를 처리하거나 규제 준수 요건이 적용되는 애플리케이션에는 적합하지 않습니다.

AWSKRUG CONTAINER

Fully managed Amazon EKS MCP server (preview)

보안 중심의 통신 아키텍처

- 로컬 프록시와 SigV4 서명: 별도의 장기 자격 증명을 발급받거나 관리할 필요 없이, 기존 IAM 정책을 그대로 활용 가능
- 중앙 집중식 권한 관리: 모든 접근은 IAM 정책에 의해 제어 (기본적으로 `AmazonEKSMCPRReadonlyAccess` 정책을 제공)
- 감사 및 규정 준수: 모든 API 호출 및 도구 실행 내역은 AWS CloudTrail에 기록(AI 에이전트가 수행한 작업에 대한 투명성 및 추적 가능성 보장)

AWSKRUG CONTAINER

Auto Scaling으로는 대응할 수 없는 규모와 복잡도에 도달 한 현대 워크로드

1. 오퍼레이터 생태계의 폭발

- ArgoCD, Crossplane, Flux, Prometheus Operator, Cert-Manager, External Secrets...
- 각 오퍼레이터마다 수천~수만 개의 CRD 생성 → **8GB etcd 한계** 도달

2. GitOps와 대규모 멀티 테넌시

- 수백 개 네임스페이스, 수천 개 애플리케이션을 GitOps로 관리
- 배포 시 수만 개 리소스 동시 생성/업데이트 → API 서버 과부하

3. Observability 스택의 무게

- Prometheus, Grafana, Loki, OpenTelemetry Collector
- 메트릭, 로그, 트레이스 수집을 위한 DaemonSet, ServiceMonitor → 지속적인 API 폭격

4. 초대형 AI/ML 훈련 및 추론 워크로드의 출현

- 2025.07 EKS 단일 클러스터 100,000개 노드 지원 발표

AWSKRUG CONTAINER

Amazon EKS Provisioned Control Plane



컨트롤 플레인 용량을 미리 할당해, 까다로운 워크로드에도 예측 가능하고 높은 성능을 보장

- 기존 Standard 모드: 워크로드 부하에 따라 반응적(Reactive)으로 자동 확장
- Provisioned 모드: 사용자가 선택한 티어에 따라 선제적(Proactive)으로 용량 확보

Provisioned Control Plane 티어별 성능 사양

성능 지표 (Kubernetes v1.30+ 기준)

티어	API 동시성(Seats)	파드 스케줄링(Pods/sec)	etcd 크기(GB)	월 예상 비용
Standard	-	-	8 GB	기본 포함
XL	1,700	167	16 GB	~\$1,200
2XL	3,400	283	16 GB	~\$2,400
4XL	6,800	400	16 GB	~\$5,000

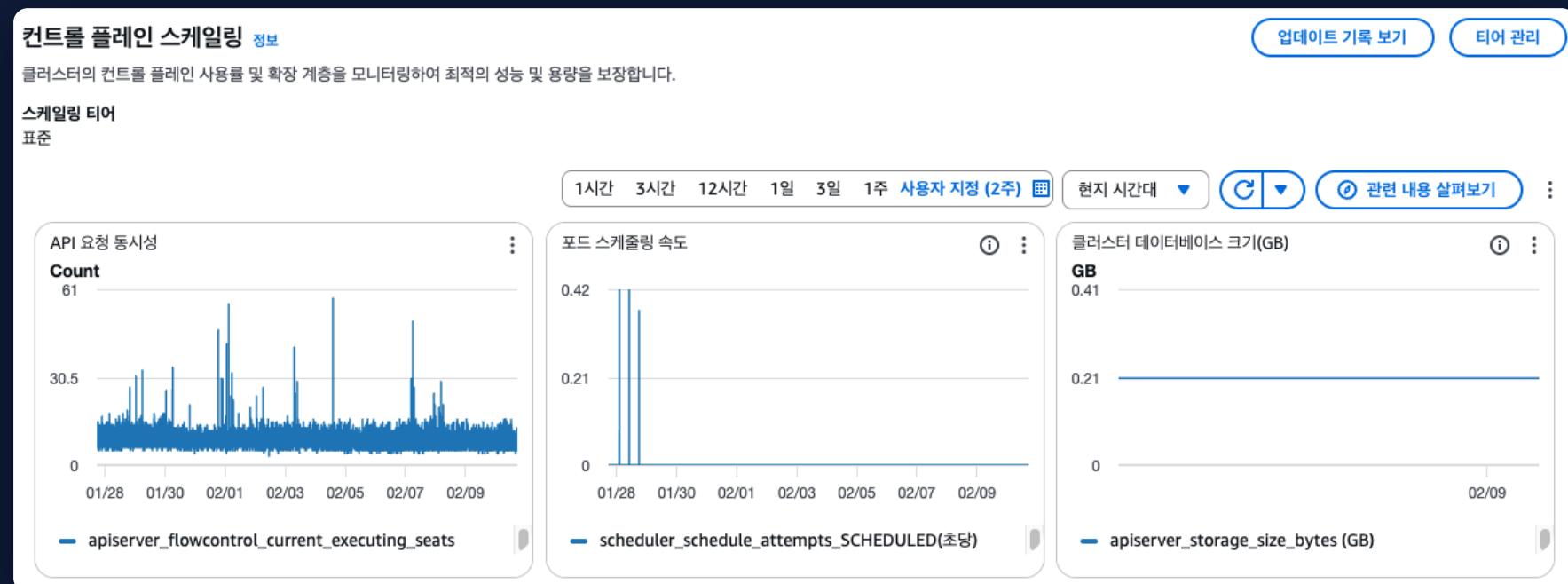
관리형 서비스가 '통제권'을 돌려주기 시작했다

AWSKRUG CONTAINER

Amazon EKS Provisioned Control Plane

적정 티어 산정 방법

클러스터 > 모니터 클러스터 > 컨트롤 플레인 모니터링



고려사항

- Standard → Provisioned 전환은 언제든 가능, 역방향은 etcd 크기 제약 존재 (8GB 미만)

AWSKRUG CONTAINER

EKS Capabilities: Argo CD

 EKS Capabilities를 사용하면 복잡한 솔루션 인프라를 관리하지 않고도 Kubernetes 애플리케이션을 구축하고 확장할 수 있습니다.

EKS Blueprints

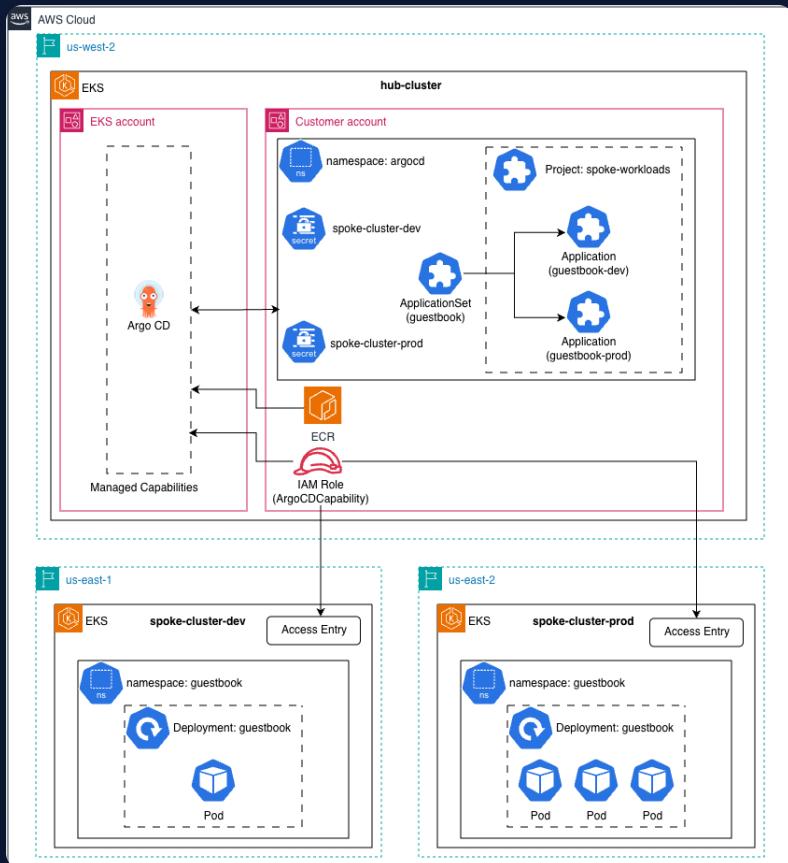
- 계정 및 리전 전반에 걸쳐 일관되고 필요한 모든 기능을 갖춘 EKS 클러스터를 구성하고 배포하는 데 도움이 되는 IaC 모듈 모음
- Amazon EKS 애드온은 물론 Prometheus, Karpenter, Argo CD 등 다양한 오픈 소스 애드온을 활용하여 클러스터를 간편하게 부트스트랩

관리형 Argo CD

- EKS Capability for Argo CD는 단순히 사용자의 클러스터에 헬름 차트(Helm Chart)를 대신 설치해주는 기능이 아님
- AWS의 인프라 위에서 실행되는 완전 관리형 컨트롤 플레인(Managed Control Plane) 형태로 제공

AWSKRUG CONTAINER

Amazon EKS Capability for Argo CD Architecture



컨트롤 플레인과 데이터 플레인의 분리

- 리소스 격리: Argo CD 컨트롤러가 사용자의 워크로드와 컴퓨팅 리소스(CPU, Memory)를 경합하지 않음
- 보안 강화: Argo CD가 실행되는 인프라에 대한 접근 권한이 사용자에게 없으며, 물리적으로 분리된 환경(ArgoCD가 내 VPC에 없음) 구성 가능

네트워크 혁신: EKS Access Entries

- VPC Peering이나 Transit Gateway를 필요로 하지 않음
- AWS의 백본 네트워크를 통해 EKS API 서버와 직접 통신

통합된 아이덴티티 관리: AWS IAM Identity Center

[Deep dive: Streamlining GitOps with Amazon EKS capability for Argo CD](#)

AWSKRUG CONTAINER

EKS Capabilities: ACK & KRO

AWS Controllers for Kubernetes (ACK)

YAML 형식을 사용하여 AWS 서비스 리소스(S3, RDS, SQS 등)를 정의하고 관리할 수 있게 해주는 프로젝트

- 작동 원리: 사용자가 Kubernetes API 서버에 Bucket이나 DBInstance와 같은 커스텀 리소스를 생성하면, AWS 관리 영역에 있는 ACK 컨트롤러가 이를 감지하고 AWS API를 호출하여 리소스를 생성

Managed ACK vs Self-managed ACK

비교 항목	Self-managed ACK	Managed EKS Capability ACK
설치 및 유지보수	사용자가 Helm 차트, CRD, 이미지 버전 직접 관리	AWS가 설치, 패치, 업그레이드 전담
인프라 비용	사용자 클러스터 워커 노드 리소스(CPU/Mem) 소모	AWS 서비스 계정 실행 (클러스터 리소스 미사용)
IAM 인증	IRSA (OIDC + ServiceAccount + Role) 구성 복잡	IAM Capability Role (Service Principal 신뢰) 단순화

- Self-managed 방식은 IRSA를 구성하기 위해 OIDC 공급자를 생성하고, 서비스 계정과 IAM 역할을 매핑하는 복잡한 과정이 필요

AWSKRUG CONTAINER

EKS Capabilities: ACK & KRO

Kube Resource Orchestrator (KRO)

저수준 ACK 리소스(예: RDS 인스턴스 + 보안 그룹 + 서브넷)를 묶어 하나의 고수준 추상화 리소스(예: CorporateDatabase)로 정의할 수 있게 함

KRO 등장 이전, 쿠버네티스 API를 확장하여 자신만의 플랫폼 로직을 구현하는 방법

- Helm 차트: 여러 리소스를 묶어 배포할 수 있지만, 배포 후 리소스 간의 의존성 관리나 상태 변화에 따른 자동 복구(Reconciliation) 기능 없음
- Operator 개발: 쿠버네티스 컨트롤러를 직접 개발하여 완벽한 제어권을 가질 수 있지만, Go 언어, client-go, controller-runtime 라이브러리에 대한 깊은 이해가 필요하며, 유지보수 비용 높음

복잡한 코딩 없이 선언적 정의만으로 오퍼레이터 수준의 런타임 관리 능력을 제공하는 것이 KRO의 핵심 목표

- 플랫폼 엔지니어링 팀이 개발자에게 복잡한 AWS 설정을 숨기고, 자체적인 커스텀 API(Self-service API)를 제공 할 수 있게 하는 도구
- 여러 리소스를 하나의 템플릿으로 정의하고, ResourceGraphDefinition이라는 CRD를 통해 리소스 간의 관계 정의

AWSKRUG CONTAINER

Enhanced Network Observability

Amazon EKS가 CloudWatch Network Flow Monitor 기반의 새로운 네트워크 관찰성 기능을 출시하여 세분화된 네트워크 지표와 AWS 콘솔 시각화를 통해 Kubernetes 클러스터의 트래픽 모니터링 및 문제 해결을 강화했습니다.

등장 배경

- 마이크로서비스 아키텍처 전환 → 시스템 복잡성이 애플리케이션 코드에서 **네트워크 레이어로 이동**
- 운영의 어려움 : 네트워크가 병목인지, 애플리케이션 로직 문제인지 구분 불가
- eBPF 기반 심층 모니터링 : 각 노드에 `aws-network-flow-monitor-agent` 배포

AWSKRUG CONTAINER

VPC Flow Logs의 구조적 한계

1. 컨텍스트의 부재

- VPC Flow Logs: 5-tuple (Src IP, Dst IP, Port, Protocol) + Action (ACCEPT/REJECT)
- 문제: 10.0.1.50이 특정 시점에 '결제 서비스'였는지 '로그 수집기'였는지 사후 파악 불가능
- 문제 발생 시점의 IP 할당 기록을 추적하는 과정 필요 → MTTD(Mean Time To Detect, 평균 탐지 시간) 지연

2. 성능 데이터의 결여

- ACCEPT 로그: 방화벽 통과 사실만 의미, 통신 품질(Quality) 정보 없음
- 패킷 손실로 재전송 급증 → 응답시간 200ms에서 2초 증가해도 "정상 트래픽"으로 기록
- 결과: 네트워크 병목인지 애플리케이션 코드 문제인지 구분 불가능

3. 인프라 제한에 의한 사각지대

- AWS 인스턴스별 네트워크 제한: 대역폭, PPS, conntrack 수
- 제한 초과 시 패킷 드롭 → 보안 그룹/NACL 차단 아니므로 REJECT에 기록 안 됨 (Blind Spot)

AWSKRUG CONTAINER

Amazon EKS Container Network Observability

구성 요소	기능적 특징	운영적 가치
Service Map	클러스터 내 워크로드 간의 통신 관계를 방향성 그래프(Directed Graph)로 시각화. 노드는 서비스/파드를, 엣지는 트래픽 흐름을 나타냄.	전체 시스템 아키텍처의 직관적 파악 가능. 병목 구간, 의존성 관계, 비정상적인 트래픽 흐름을 시각적으로 즉시 식별
Flow Table	'Top Talkers' 분석을 위한 상세 테이블 뷰. 클러스터 내부(East-West), AWS 서비스행, 외부 인터넷행 트래픽을 구분하여 제공	대역폭을 과도하게 점유하는 서비스 식별, 비용 최적화 대상(예: S3 과다 호출) 파악, 보안 정책 위반 통신 추적
Performance Metrics	패킷 손실, 지연 시간, 처리량 등 네트워크 품질에 직결된 정량적 데이터 제공. 네트워크 상태 지표(NHI) 포함	'네트워크가 느리다'는 막연한 불만을 데이터 기반으로 검증. 인프라 문제인지 애플리케이션 문제인지 격리(Isolation) 가능

AWSKRUG CONTAINER

EKS Container Network Observability 비용 계산

구성 요소

Container Network Observability 활성화 시(NFM)

- Kubernetes 워커 노드당 **1개의 에이전트** 생성
- **1개의 Flow Monitor** 생성 (5개의 CloudWatch 메트릭 제공)

비용 산출 기준

1달 30일 기준 50개의 노드 사용 시

항목	계산식	월 비용
CloudWatch 메트릭 비용	5개 메트릭 × \$0.30/메트릭/월	\$1.50
Flow Monitoring 비용	50 노드 × \$0.0069/리소스/시간 × 720시간	\$248.40
총 비용	메트릭 비용 + Flow Monitoring 비용	\$249.90/월

AWSKRUG CONTAINER

Key Takeaways

Managed의 재정의

"관리형 + 통제권 유지"가 새로운 기준, 운영 부담은 줄이면서 아키텍처 결정권은 유지하도록 설계되었습니다.

Provisioned Control Plane, EKS Capability (Argo CD, ACK, KRO)

Secure by Default는 선택이 아니라 기본값

보안은 별도 절차가 아니라 플랫폼에 자동으로 적용됩니다

ECR Managed Signing, IAM 기반 MCP

Day 0 마찰 제거 = 경쟁력

"설정"과 "문제 분석"에 드는 시간을 줄입니다

ECS Express Mode, AI-powered Troubleshooting, EKS CNO

규모의 시대, 관측 가능성이 생존 조건

병목을 데이터로 식별하는 능력을 제공합니다

Provisioned Control Plane, EKS CNO

AWSKRUG CONTAINER

감사합니다!!

