aws FOR GAMES

# GAMES ON AWS 2026

GS Neotek | MEGAZONE CLOUD

GAMES ON AWS 2026

aws FOR GAMES

# Amazon EKS 기반 VARCO 3D 서빙

## Karpenter와 AI Agent로 완성하는 GPU비용·가용성 최적화


이하준

AX Engineering AIOps Team Lead

NC AI

김현동

Solutions Architect

AWS


GS Neotek | MEGAZONE CLOUD

# Agenda

- AWS 상에서 모델 호스팅 선택 옵션
- 모델 호스팅을 위한 EKS 인프라 준비
- VARCO 3D 소개
- 기존 아키텍처 및 한계점
- 활용 및 아키텍처
- 주요 산출물 및 기대효과
- 요약 및 향후 계획

GAMES ON AWS 2026

aws FOR GAMES

# AWS 상에서 모델 호스팅 선택

GS Neotek | MEGAZONE CLOUD

# AWS 상에서 모델 호스팅 선택 옵션

## Amazon Bedrock



완전관리형 · 서버리스, 멀티 FM

**언제?**

✓ 빠른 GenAI 앱 출시
✓ 표준 FM으로 충분
✓ 가변 트래픽

## Amazon Sagemaker



관리형 ML 플랫폼 · 커스텀 EP

**언제?**

✓ 자체 모델 제어 배포
✓ MLOps 필요
✓ 무거운 추론 → Async

## Amazon EKS



관리형 Kubernetes 플랫폼

**언제?**

✓ 최대 제어·비용 최적화
✓ 하이브리드 환경
✓ K8s 역량 보유

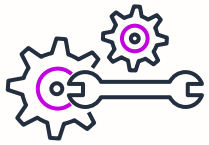
# AWS 상에서 모델 호스팅 선택 옵션

## Why hosting model on EKS?



**풍부한 오픈소스 생태계**

많은 모델 추론 도구가 Kubernetes 형태로 출시



**Extensibility**

K8S-Native 방식으로 AWS 인프라 서비스와 통합



**Cost optimization**

비용 최적화된 GenAI 운영 환경 제어



**Configurability**

완전한 구성제어.



**뛰어난 확장성**

GPU 인스턴스를 fleet 단위로 확장 운영하는 데 필요한 기능을 기본 제공



**어디서나 동작하는 이식성**

한 환경에서 익힌 것은 어디서나 그대로 통합니다.

GAMES ON AWS 2026

aws FOR GAMES

# 모델 호스팅을 위한 EKS 인프라 준비

GS Neotek | MEGAZONE CLOUD

# 모델 호스팅을 위한 EKS 인프라 준비

## GPU 추론을 EKS에서 운영할 때의 고려점

### 용량

**특정 GPU 타입은 "있을 때 잡아야" 합니다**

Spot은 물론 On-Demand도 용량 부족 (InsufficientInstanceCapacity)이 흔합니다

### 비용

**Spot과 On-Demand 사이의 균형**

폴백한 온디맨드를 Spot으로 되돌리는 시점과 유휴 노드 회수 속도가 월 비용을 좌우

### 속도

**신규 GPU 노드가 첫 추론을 처리하기까지 시간**

노드 생성 → GPU 인식 → 이미지 pull → 모델 다운로드·적재
그동안 요청은 큐에 대기.

# 모델 호스팅을 위한 EKS 인프라 준비

## GPU 구매 옵션 선택의 시작은 Spec Sheet 보다 "용량"에서 시작

| | On-Demand | Spot | ODCR | Capacity Block |
|---|---|---|---|---|
| **G 계열 (g6e·g7e ...)** | available | available · 최대 -80% | available | — |
| **P 계열 (p5·p6 ...)** | usually unavailable | available | available | available, up to -60% |
| **용량 보장** | No | No | yes | yes |
| **시작 시점** | 즉시 | 즉시 | 예약 시점부터(조정 가능) | 30분~8주 전 예약 |
| **Karpenter** | `capacity-type: on-demand` | `capacity-type: spot` | `capacity-type: reserved` | `capacity-type: reserved` |

**P 계열 온디맨드는 대체로 확보가 어렵습니다**
조달로 해결할 수 있는 문제가 아니라, 설계가 전제로 받아들여야 하는 제약입니다.

**리전 가용성이 최우선 제약입니다**
용량이 확보되는 리전이 클러스터의 위치를 결정합니다.

**예약 용량도 Karpenter 스케줄링에 그대로 명시할 수 있습니다**
`capacity-type: reserved`만 지정하면 ODCR과 Capacity Block이 Spot·온디맨드와 동일한 NodePool 요구사항 안에서 선택됩니다.

# 모델 호스팅을 위한 EKS 인프라 준비

## GPU 선택 기준 – 메모리 · 메모리 대역폭 · 노드 내 GPU 연결(Interconnect)

| 인스턴스 | GPU | 메모리 — GPU당 / 노드 전체 | GPU 메모리 대역폭 | 노드 내 GPU 링크(양방향) |
|---|---|---|---|---|
| g6e | 1–8 × L40S | 48 GB GDDR6 / 최대 384 GB | 864 GB/s | NVLink 없음 — PCIe |
| **g7e** | 1–8 × RTX PRO 6000 Blackwell | **96 GB GDDR7** / 최대 768 GB · MIG 지원 | ~1.6 TB/s | NVLink 없음 — PCIe |
| p4d / p4de | 8 × A100 | 40–80 GB HBM2e / 320–640 GB | 2.0 TB/s | NVLink 600 GB/s |
| p5 | 8 × H100 | 80 GB HBM3 / 640 GB | 3.35 TB/s | NVLink 900 GB/s + EFA |
| p5en | 8 × H200 | 141 GB HBM3e / 1,128 GB | 4.8 TB/s | NVLink 900 GB/s + EFA |
| p6-b200 | 8 × B200 | ~179 GB HBM3e / 1,432 GB | ~8 TB/s | NVLink 1.8 TB/s + EFA |
| p6-b300 | 8 × B300 | **~268 GB HBM3e / 2,144 GB** | ~8 TB/s | NVLink 1.8 TB/s + EFA |

* '대역폭' 열은 **GPU 자체의 메모리 대역폭**(HBM 또는 GDDR)을 의미하며, 네트워크 대역폭이 아닙니다. (NVLink 수치는 양방향 합계)

! **L40S·RTX PRO 6000은 GDDR(Not HBM)** — LLM 디코드처럼 메모리 대역폭에 병목이 생기는 작업에는 구조적으로 불리합니다. NVLink가 없어 노드 내에서 텐서 병렬을 구성하더라도 PCIe 통신 비용을 감수해야 합니다.

✓ 3D 생성 추론은 디코드형이 아니라 연산 바운드(compute-bound) 워크로드에 가까워, G-Series 가 적합합니다.

# 모델 호스팅을 위한 EKS 인프라 준비

## Karpenter – 노드 수명 주기 관리 도구

### Karpenter 특징

**Pending Pod → 노드 즉시 생성**

Pod의 CPU·memory·GPU 요청을 읽어 가장 적합한 EC2 인스턴스를 직접 생성. 인스턴스 타입마다 노드그룹을 미리 정의할 필요가 없습니다.

**다양한 인스턴스·구매옵션 유연성**

하나의 NodePool에 여러 instance-family와 capacity-type을 함께 선언하면, Karpenter가 비용·가용성 기준으로 선택하고 폴백합니다.

**Just-in-time provisioning & consolidation**

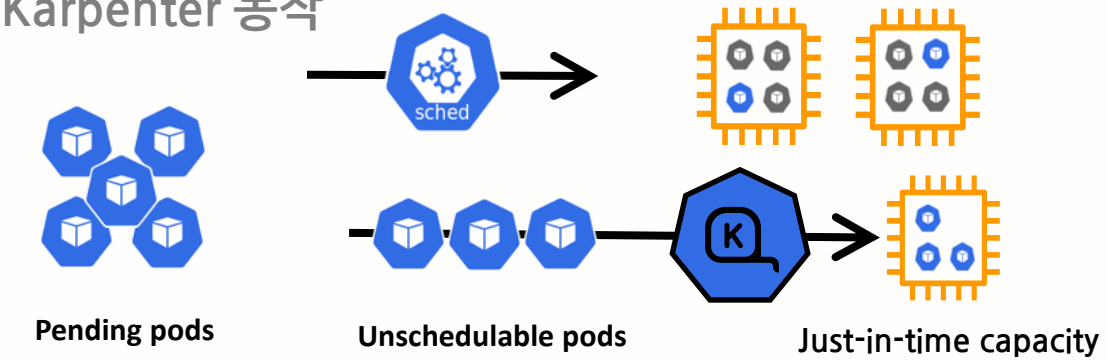
필요할 때 노드를 만들고, 노드에 Pod가 없거나 리소스가 남으면 consolidation으로 회수해 비용을 최적화합니다.

.

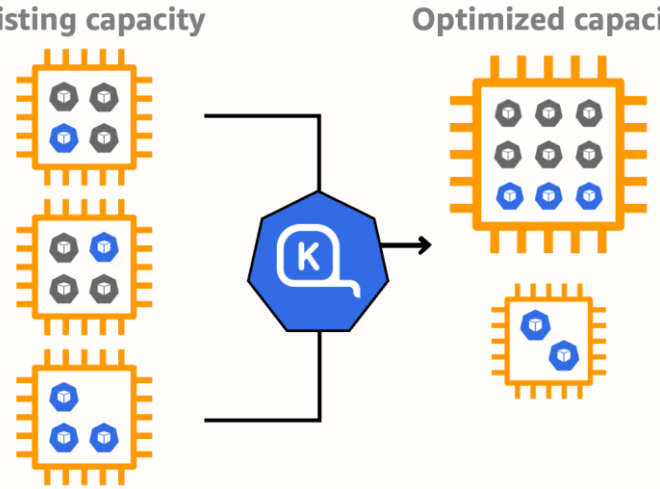
**Node Repair (노드 자동 복구)**

노드의 상태 이상(예: GPU XID 오류, kubelet 비정상)을 감지하면 문제 노드를 자동으로 cordon·drain하고 새 노드로 교체합니다.

Karpenter 동작




Consolidation



# 모델 호스팅을 위한 EKS 인프라 준비

## Karpenter가 GPU 추론에 적합한 이유

### 용량 부족(InsufficientInstanceCapacity) 회피

여러 **인스턴스 타입**·AZ·**구매옵션**을 한 NodePool에 명시. 자동 Fallback

상시 수요는 Reserved, 변동분은 Spot→On-Demand. Karpenter가 비용·가용성 기준으로 선택합니다

### Spot 중단 대응 ('--interruption-queue')

Spot 중단 경고 이벤트에 선제 drain·terminate · 신규 노드 조달합니다

### GPU에 맞는 스케줄 제어

well-known label(instance-gpu-name, instance-gpu-memory)로 모델 요건에 맞는 GPU를 고르고, consolidation은 WhenEmpty로 묶어 콜드스타트 낭비를 막습니다.

### Karpenter NodePool 예제

```yaml
kind: NodePool
metadata:
  name: gpu-nodepool
spec:
  template:
    spec:
      requirements:
        - key: karpenter.k8s.aws/instance-family
          values: [p5, p5en, p6-b300] # ① 여러 패밀리
        - key: karpenter.sh/capacity-type # ② 모두 선언
          values: [reserved, on-demand, spot]
      taints:
        - key: nvidia.com/gpu    # ③ GPU 사용하는 Pod만
          effect: NoSchedule
  disruption:
    consolidationPolicy: WhenEmpty   # ④ 기본값 변경
```

* 예약 ID(ODCR·Capacity Block)는 EC2NodeClass에 적습니다.

**Pending Pod의 요구사항에서 노드 생성**

Pod가 nvidia.com/gpu: 8을 요청하면 그것을 만족하는 인스턴스를 찾아 띄웁니다. 인스턴스 타입마다 노드그룹을 미리 정의하지 않습니다.

# 모델 호스팅을 위한 EKS 인프라 준비

## EKS Accelerated AMI 선택

### AL2023 Accelerated AMI + GPU Operator Node Start 절차

**AL2023**

NVIDIA

- NVIDIA Runtime
- NVIDIA kernel driver with MIG Support
- EFA Adapter
- Optimized for EKS OS

유연성·커스터마이즈 우선
셀·패키지 접근
호스트 설치형 에이전트

- **EC2 부팅**
  OS 부팅 · kubelet 등록 (31~33초)
- **★ Node Ready**
  아직 nvidia.com/gpu: 0
- **NFD** (Node Feature Discovery)
  GPU 하드웨어 탐지 · K8s 라벨
- **GPU Operator 라벨**
  배포 대상 노드 지정
- **driver validator**
  드라이버 정상 동작 검증
- **Device Plugin 기동**
  이미지 pull — 최대 지연 구간
- **MIG Manager 파티셔닝**
  GPU 분할(~40s) + Plugin 재등록
- **nvidia.com/gpu 노출**
  GPU 워크로드 배치 가능

### Bottlerocket Accelerated AMI Node Start 절차

**Bottlerocket**

NVIDIA

- Device Plugin/ MIG Manager
- NVIDIA Runtime
- NVIDIA kernel driver with MIG Support
- EFA Adapter
- Optimized for EKS OS

- **EC2 부팅**
  불변 이미지 · 드라이버·플러그인 OS 내장
- **드라이버 로드 (systemd)**
  AMI 내장 드라이버 즉시 로드 — 검증 Pod 불필요
- **nvidia-mig-manager.service**
  GPU idle에서 MIG 적용
- **nvidia-k8s-device-plugin.service**
  MIG 슬롯 광고
- **★ Node Ready**
  Kubelet 등록
- **nvidia.com/gpu 노출**
  GPU 워크로드 배치 가능

빠른 콜드스타트, 보안이 우선일때. GPU 스택은 AMI 제공 버전 고정.

# 모델 호스팅을 위한 EKS 인프라 준비

## Startup 최적화 - Best Practices

### Startup 시간 구간별 정리 (Flex.2 예제 모델 기준)

**노드 조달 → GPU 인식 → 이미지 pull → 모델 다운로드 → 적재·추론**



**주로 모델 다운로드가 단일 최대 구간**
네트워크가 아니라 디스크 쓰기가 병목
gp3 기본 125MB/s 처리량이 상한 (g7e)

### ML 아티팩트 관리 — 모델을 어디에 두는가

**1. 런타임 다운로드 — 기본 권장**
크고 자주 갱신되는 모델. Pod 기동 시 S3·HF에서 로드(S3 CSI·s5cmd로 가속). 단, 다운로드가 콜드 스타트 최대 병목

**2. 이미지에 베이킹(Bake) — 작고 드문 모델만**
일관성·무지연 로드. 단 빌드·푸시 느리고 갱신마다 재빌드·재배포 필요

**3. Serving Framework — vLLM·TGI·Triton**
프레임워크가 이미지 크기·시작 시간에 직접 영향 → 아티팩트와 분리 권장

### Startup 최적화 Best Practices

**이미지 크기 축소**
런타임 전용 이미지(dev 라이브러리 제외)로 pull 용량을 줄임

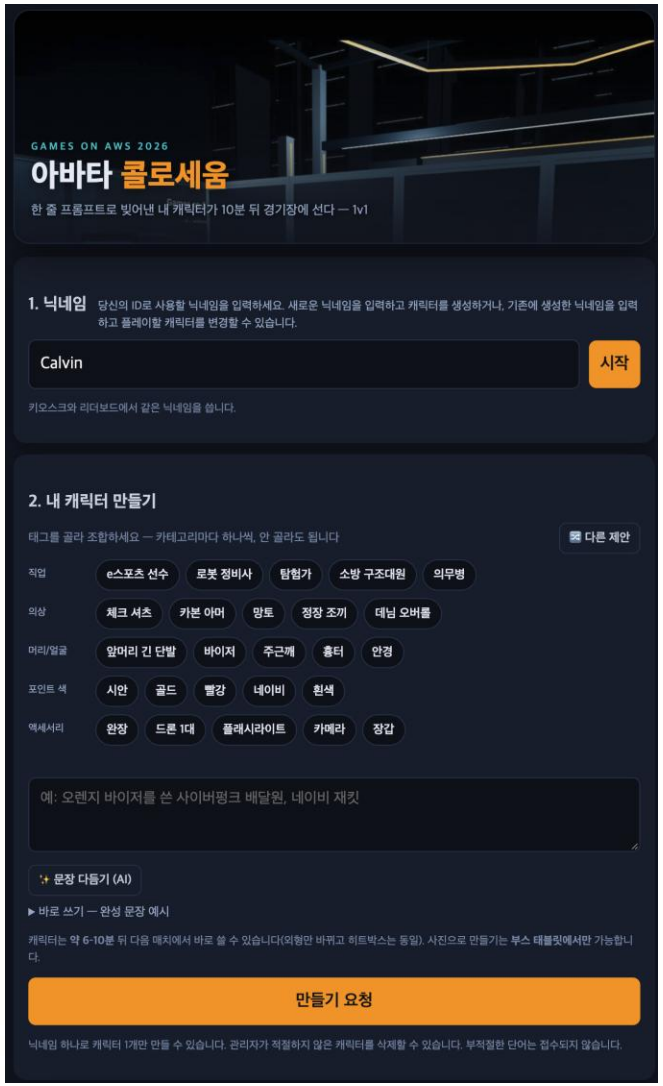
**이미지 pull 지연 단축**
Seekable OCI 병렬 pull/unpack
ECR VPC 엔드포인트
ECR pull-through 캐시
이미지 크기를 고려한 네트워크/스토리지 성능의 인스턴트 (baseline)

**NVMe 인스턴스 스토어**
이미지와 모델을 로컬 NVMe (RAID0)로 저장 — 높은 디스크 I/O·처리량

**사전 적재 (pre-pull)**
EBS 스냅샷

https://docs.aws.amazon.com/eks/latest/best-practices/aiml-performance.html

# 아바타 콜로세움 — 한 줄 프롬프트로 만든 나만의 3D 캐릭터로 게임 플레이 데모







https://bit.ly/3W4CQhK

GAMES ON AWS 2026

aws FOR GAMES

# VARCO 3D 소개

GS Neotek | MEGAZONE CLOUD

# VARCO 3D 소개

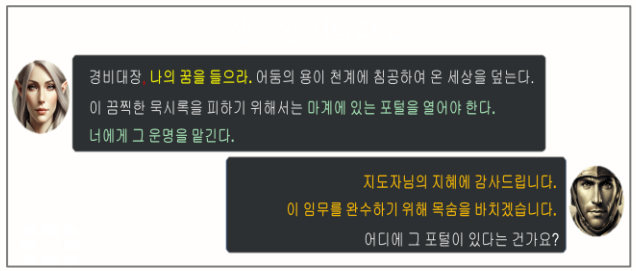
## Background

게임은 텍스트/이미지/오디오/3D 등 다양한 형식의
데이터가 일관된 조화를 이루는 종합 예술입니다.

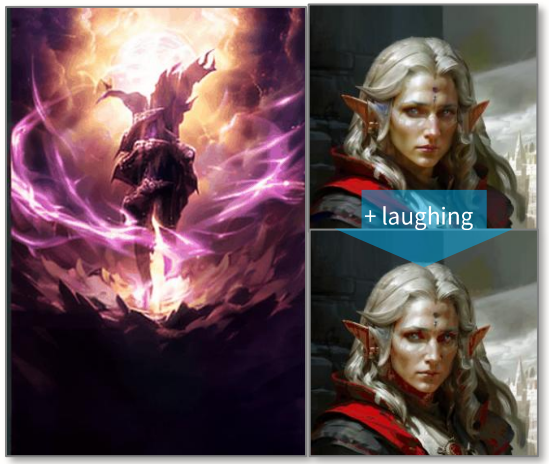


퀘스트 시나리오/NPC 대화 → 목소리/BGM → 캐릭터/배경 2D 컨셉 아트 → 2D 애니메이션 / 효과 영상 → 3D 모델링/텍스처 → 3D 애니메이션





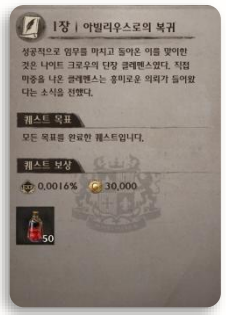






# VARCO 3D 소개

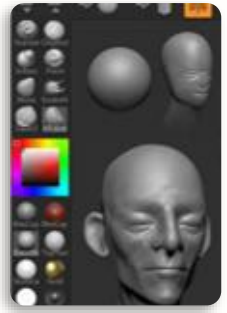
## Background

Art 분야 중에서도 가장 오래 걸리는 작업은 3D & 애니메이션 등 CG (Computer Graphics) 리소스 제작 과정
각 단계별로 전문성을 지닌 아티스트들이 작업하는 데에 최소 16시간 ~ 1주 이상의 시간이 소요됩니다.

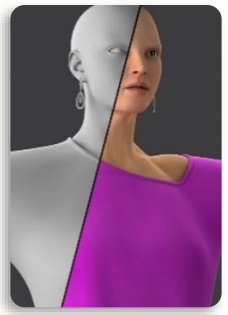


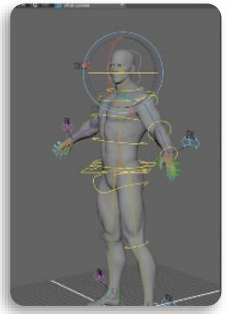
4 ~ 8시간

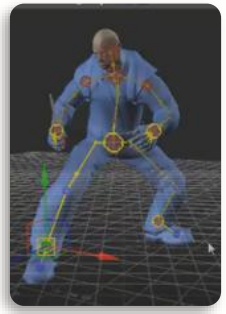

16 ~ 48시간


48 ~ 72시간


8 ~ 24시간


8 ~ 24시간


24 ~ 48시간


8 ~ 64시간

Character


8 ~ 16시간


4 ~ 8시간

Environment

**1개의 3D 애셋 제작 기간 = 4주 이상 소요**

(리깅 / 애니메이션 없는 경우에도, 외주 제작시 최소 3~5일 소요

# VARCO 3D 소개

## Problems

첫 결과를 기다리는 시간부터, 후처리와 도구 전환까지 시간이 쌓였습니다.

01 초안 제작

### 첫 결과를 보기까지 오래 걸리는 시간

초안이 늦게 나오면
비교와 수정 방향 결정도 늦어집니다.

02 후처리

### 생성 이후에도 남는 수작업 후처리

UV 정리와 메시 수정 등
반복적인 정리 작업이 필요합니다.

03 제작 연계

### 도구와 공정마다 분리된 제작 작업

결과물을 옮기고 작업을 이어가며
공정 사이의 연결이 필요합니다.

**제작 시간을 줄이면서, 기존 제작 환경에서 바로 이어 쓸 수 있어야 했습니다.**

# VARCO 3D 소개

## A Production-Focused 3D AI Platform for Game and 3D Content Teams

### Layer 1. Quality

"3D 생성 결과물이
실무 활용 가능한 수준인가?"

**Model & Studio**

실무 활용 가능한 자체 3D 생성 모델과 AI 기반 후처리 기능



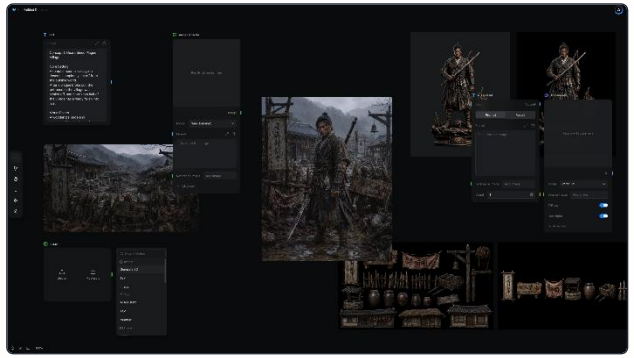
- 1M+ High-Detail Mesh, 4K Texture 생성 지원
- AI UV Unwrap / Remesh 등 AI 기반 후처리 절감 기능
- AI Rigging / Animation 기능

### Layer 2. Workflow

"단발성 생성이 아니라
반복 제작 공정으로 활용 가능한가?"

**Custom Workflow & MCP**

반복 제작 흐름을 재사용 가능한 AI Workflow / Agent로 구성



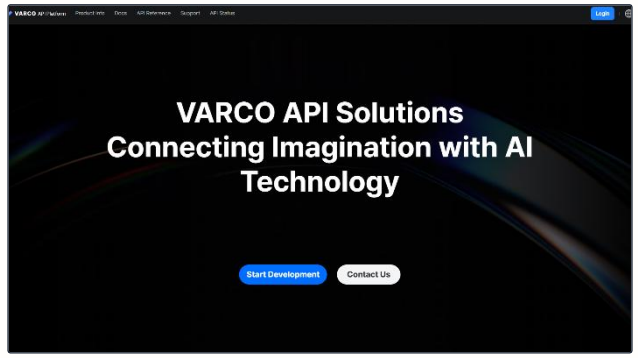
- Node 기반의 3D AI Workflow
- 반복 제작 과정을 사용자 정의 Workflow로 저장
- Export 및 DCC 연계 기반 제공

### Layer 3. Enterprise

"팀/기업 단위로 안전하고
관리 가능하게 도입할 수 있는가?"

**Enterprise Support**

보안 · 계정 · 연동 · API 까지 포함한 기업 도입 지원

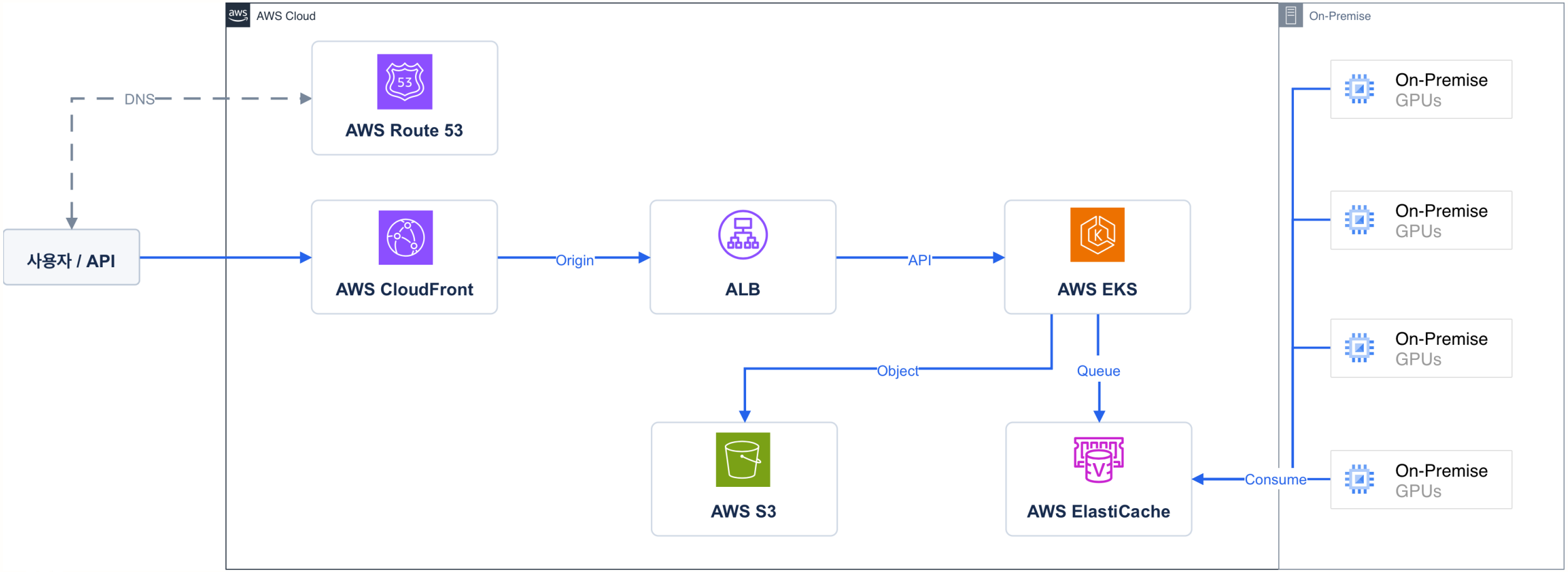



- 보안 / 데이터 관리 / 사용량 관리 요건 대응
- 고객사 Workflow에 맞춘 PoC 및 연동 지원
- API 플랫폼을 통한 3D 생성 API 제공

# 기존 아키텍처 및 한계점

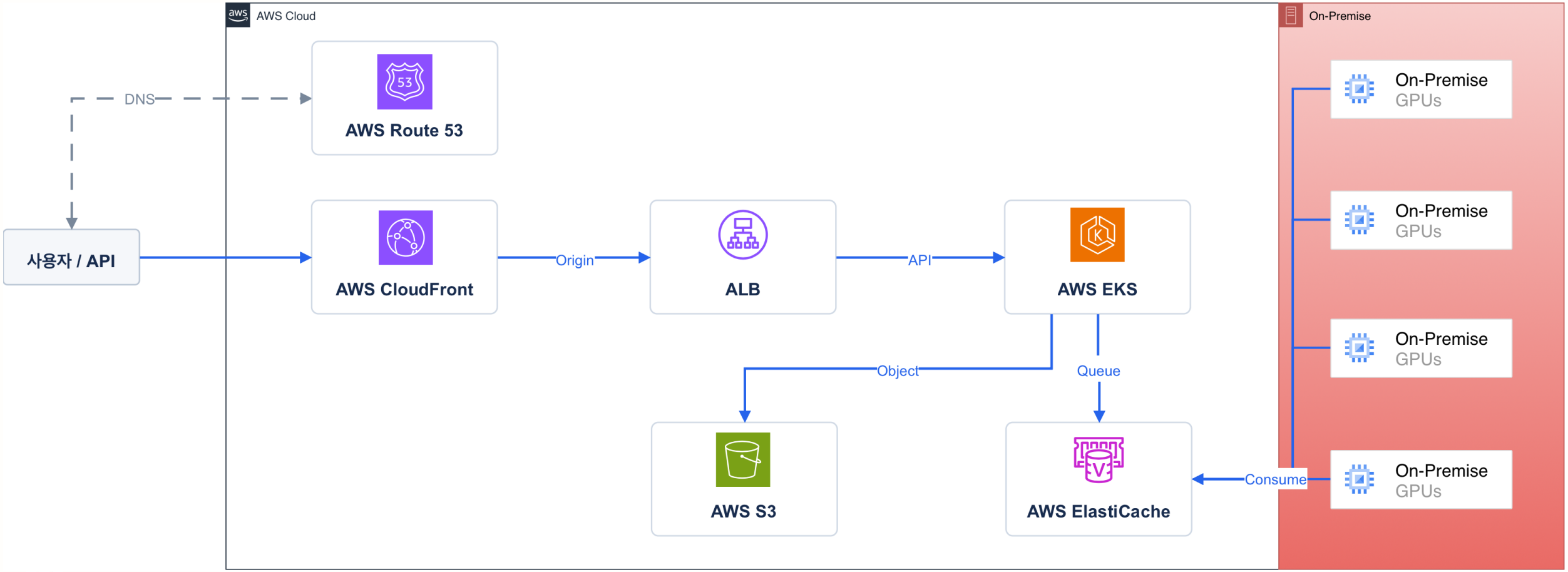
# 기존 아키텍처 및 한계점

## 기존 Architecture



# 기존 아키텍처 및 한계점

## 기존 Architecture



# 기존 아키텍처 및 한계점

## 한계점



### GPU 자원 한계

처리 자원이 한정적입니다.

### 대기·타임아웃

트래픽 급증 시 응답이 지연됩니다.

### 사용자 감소·이탈

사용성 저하가 이탈로 이어집니다.

GAMES ON AWS 2026

aws FOR GAMES

# 활용 및 현재 아키텍처

GS Neotek | MEGAZONE CLOUD

# 활용 및 현재 아키텍처

## 목표



### 수요에 맞춘 증설

큐를 감지해 Worker를 늘리고
Pending Pod에 맞춰 GPU를 확보

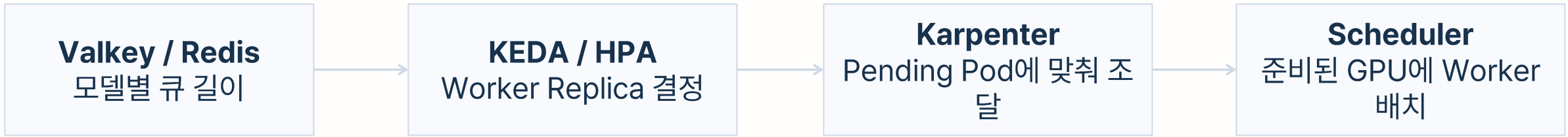
### 수요 감소 후 회수

유지 정책 이후 Worker를 축소하고
빈 GPU 노드를 회수

# 활용 및 현재 아키텍처

## Introduction to KEDA

- Kubernetes Event-Driven Autoscaling
- 외부 이벤트 (ex: Kafka, RabbitMQ, Redis)를 기반으로 Pod 개수를 조절해주는 Open Source




Worker Pod이 Pending 상태가 되면 필요한 노드를 요청

출처: https://github.com/kedacore/keda

# 활용 및 현재 아키텍처

## Introduction to MIG

- Multi-Instance GPU

- 하나의 물리적 GPU를 독립적인 GPU 인스턴스로 하드웨어 수준에서 분리해서 사용할 수 있게 해주는 NVIDIA의 자원 가상화 기술

- 단일 물리 GPU를 격리된 여러 인스턴스(슬라이스)로 분할하고, 각 슬라이스는 전용 메모리 / L2 캐시 / 대역폭을 가져 상호 간섭 없이 예측 가능한 QoS를 보장

- Hopper / Blackwell 지원

출처: https://www.nvidia.com/ko-kr/technologies/multi-instance-gpu/

# 활용 및 현재 아키텍처

## Introduction to MIG

RTX PRO 6000 Blackwell (g7e) MIG 프로파일

| 프로파일 | 컴퓨트 슬라이스 | 인스턴스당 메모리 | GPU당 최대 |
|---|---|---|---|
| 1g.24gb | 1 / 4 | 24 GB | 4 |
| 2g.48gb ★ | 2 / 4 | 48 GB | 2 |
| 4g.96gb | 4 / 4 | 96 GB (풀) | 1 |

### g7e.2xlarge

96GB GPU를 2g.48gb 두 개로 분할

**48GB**
Worker A

**48GB**
Worker B

### g6e.2xlarge

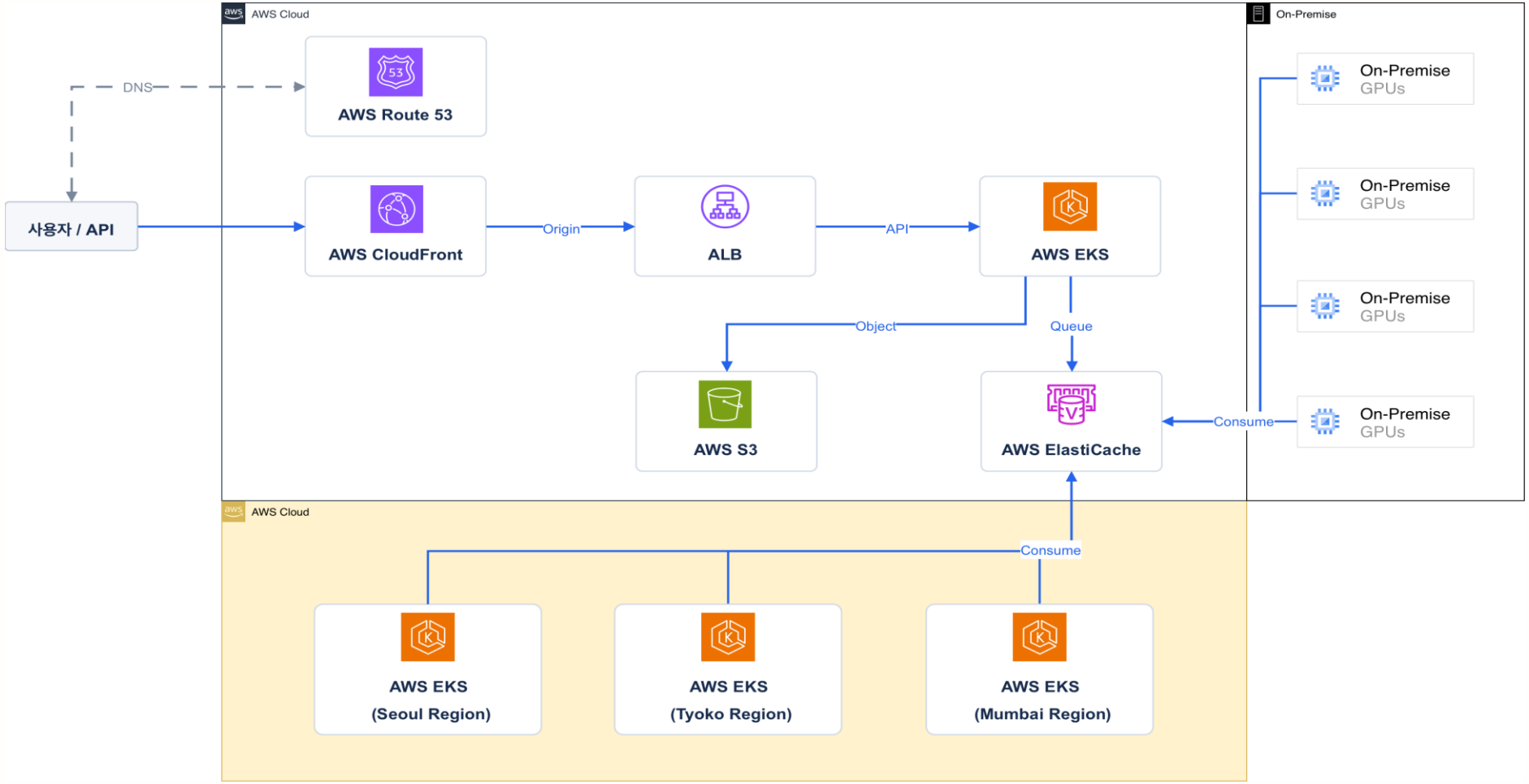
MIG를 끈 대체 구성

**전체 GPU 1개**
Worker A

출처: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/latest/supported-mig-profiles.html#rtx-pro-6000-blackwell-mig-profiles

# 활용 및 현재 아키텍처

## 현재 Architecture

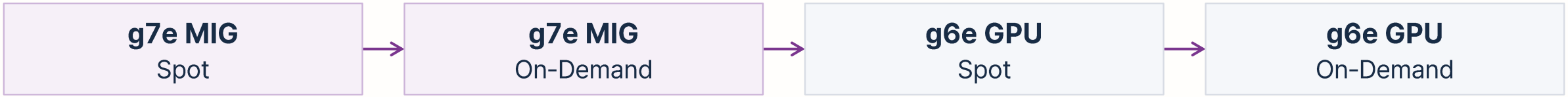


# 활용 및 현재 아키텍처

## 확장 흐름




### 각 리전의 GPU 선택 우선순위




**GPU 부족에 대응하려면, 여러 GPU에 맞춘 추론 최적화가 필요합니다**

### Standby | 대기 리전

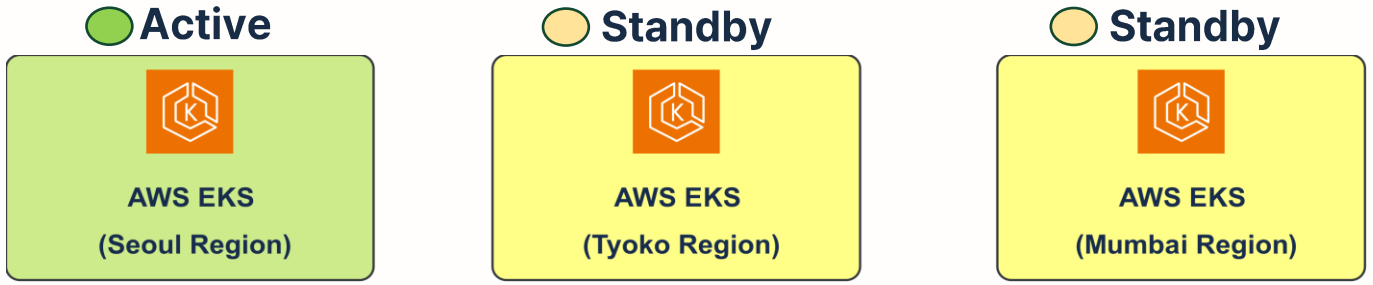
KEDA 일시정지 · GPU / Worker 0
EKS와 최소 CPU 제어부는 유지

### Active | 수요를 담당하는 리전

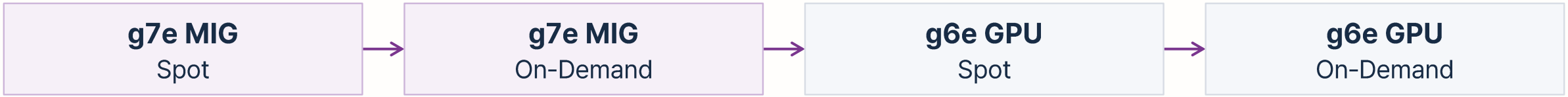
KEDA가 Worker 증설 · Karpenter가 GPU 확보
Active도 수요가 없으면 GPU 0 가능

# 활용 및 현재 아키텍처

## 확장 흐름




### 각 리전의 GPU 선택 우선순위




**GPU 부족에 대응하려면, 여러 GPU에 맞춘 추론 최적화가 필요합니다**

### Standby | 대기 리전

KEDA 일시정지 · GPU / Worker 0
EKS와 최소 CPU 제어부는 유지

### Active | 수요를 담당하는 리전

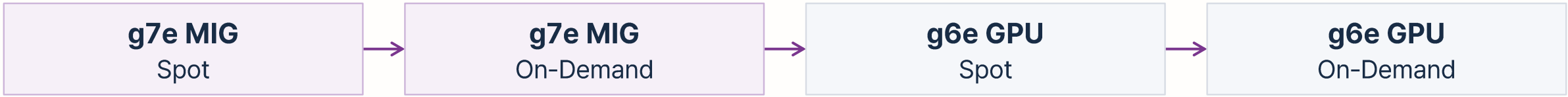
KEDA가 Worker 증설 · Karpenter가 GPU 확보
Active도 수요가 없으면 GPU 0 가능

# 활용 및 현재 아키텍처

## 확장 흐름




### 각 리전의 GPU 선택 우선순위




**GPU 부족에 대응하려면, 여러 GPU에 맞춘 추론 최적화가 필요합니다**

### Standby | 대기 리전

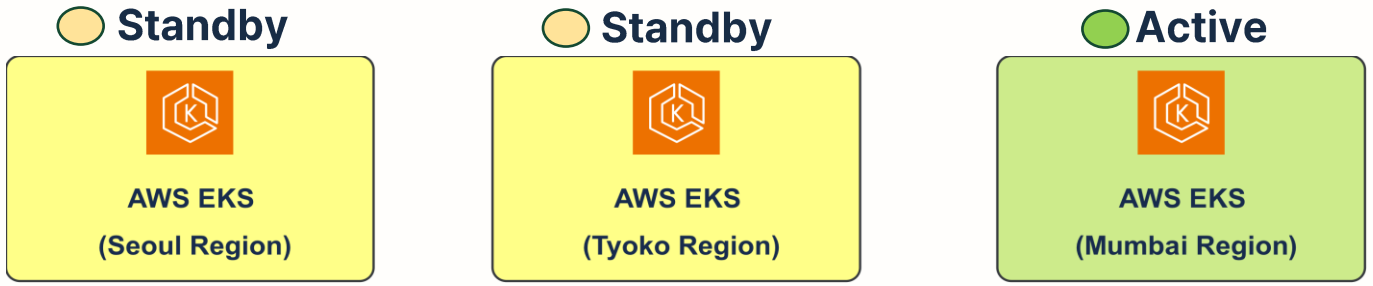
KEDA 일시정지 · GPU / Worker 0
EKS와 최소 CPU 제어부는 유지

### Active | 수요를 담당하는 리전

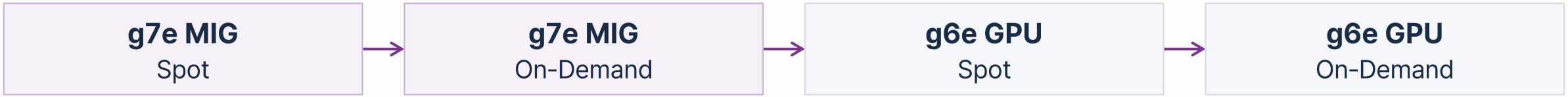
KEDA가 Worker 증설 · Karpenter가 GPU 확보
Active도 수요가 없으면 GPU 0 가능

# 활용 및 현재 아키텍처

## 확장 흐름




### 각 리전의 GPU 선택 우선순위




**GPU 부족에 대응하려면, 여러 GPU에 맞춘 추론 최적화가 필요합니다**

### Standby | 대기 리전

KEDA 일시정지 · GPU / Worker 0
EKS와 최소 CPU 제어부는 유지

### Active | 수요를 담당하는 리전

KEDA가 Worker 증설 · Karpenter가 GPU 확보
Active도 수요가 없으면 GPU 0 가능

# 활용 및 현재 아키텍처

## Spot은 만능이 아니라, 워크로드에 맞춘 선택

**VARCO 3D의 유리한 특성**

**요청 간 컨텍스트 유지 불필요**

장기 세션 복원 부담이 작아
작업 단위 재실행을 설계하기 유리

# 활용 및 현재 아키텍처

## Spot Tracker (Spot Placement Score)



# 활용 및 현재 아키텍처

## Spot 중단 흐름

### 인프라의 Spot 중단 대응

**Spot 중단 알림**
종료 약 2분 전*

→

**EventBridge / SQS**
중단 이벤트 전달

→

**Karpenter**
신규 Pod 배치 차단·드레인

→

**대체 GPU·Worker**
확보·재배치 시도

# 활용 및 현재 아키텍처

## AI Agent 도입 배경

### 모델 준비 시간 최적화

요청 후 GPU 확보와 모델 로딩으로 대기 발생

수요 흐름과 준비 시간을 고려해 예열 시점 제안

### 운영자의 장애 대응 개입 감소

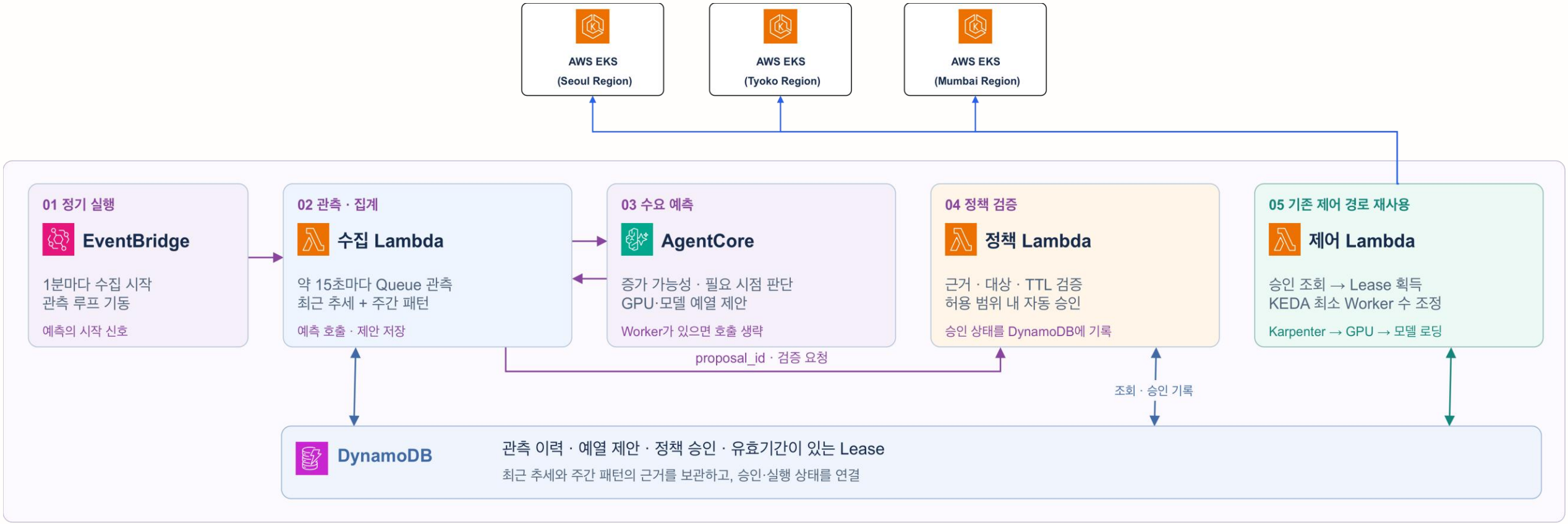
GPU·노드·Worker 상태를 직접 확인하는 부담

장애 근거를 종합해 원인과 대응안 제시



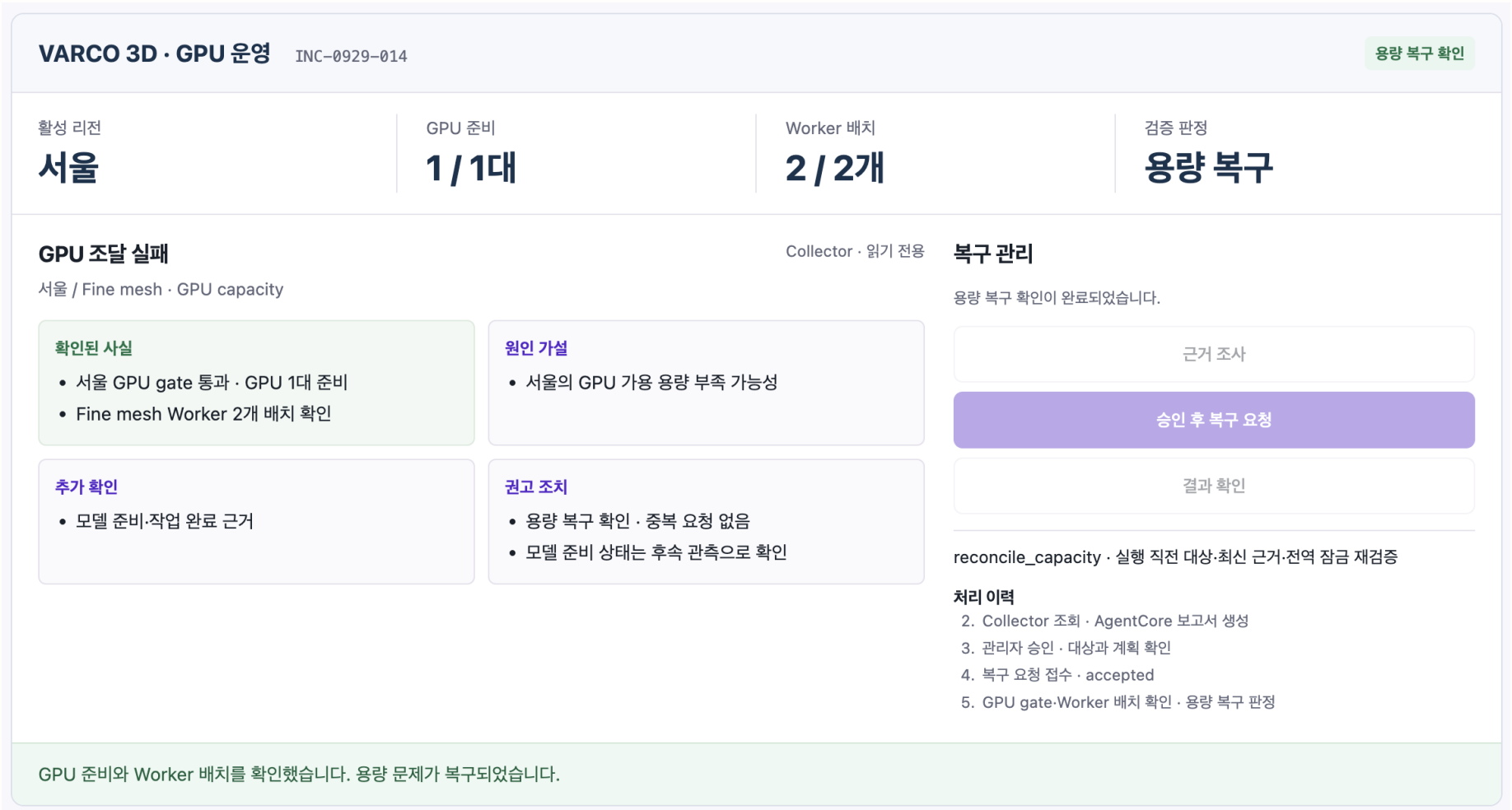
# 활용 및 현재 아키텍처

## Bedrock AgentCore 기반 예측



# 활용 및 현재 아키텍처

## 운영자 장애 대응



# 주요 산출물 및 기대효과

# 주요 산출물 및 기대효과

## GPU 구성별 인스턴스 생성 추이

### g7e 단독 구성 (과거)

| 관찰 항목 | 생성 이력에서 확인한 내용 |
|---|---|
| **생성 패턴** | 리전 사이 이동으로 인한 공백이 긴 패턴 |
| **실제 증설 GPU** | g7e |
| **Spot 활용** | 묶바이 생성 |
| **On-Demand** | 도쿄 or 묶바이 생성 |
| **GPU 선택 범위** | g7e 유형 |

### g7e + g6e 구성 (현재)

| 관찰 항목 | 생성 이력에서 확인한 내용 |
|---|---|
| **생성 패턴** | 관측 구간 전반에서 빠르게 생성 |
| **실제 증설 GPU** | g6e (g7e 간헐적 생성) |
| **Spot 활용** | g6e 서울 생성 / g7e 도쿄 |
| **On-Demand** | g6e 서울 생성 |
| **GPU 선택 범위** | g7e + g6e 유형 |

# 주요 산출물 및 기대효과

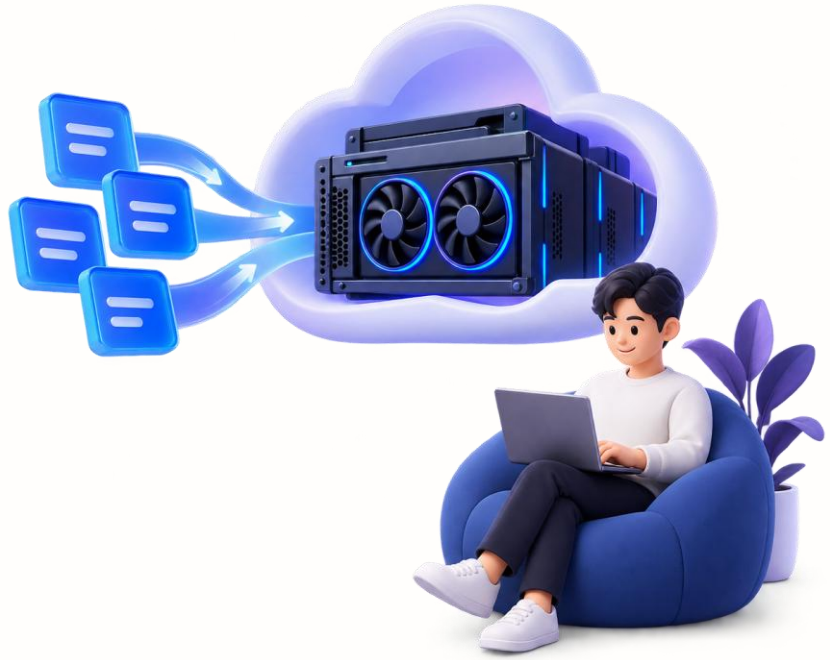
## 서비스 측면

01 대기시간 감소

02 동시 요청 처리 능력 확대

03 사용자 이탈 / 재시도 감소

04 사용성 증가



# 주요 산출물 및 기대효과

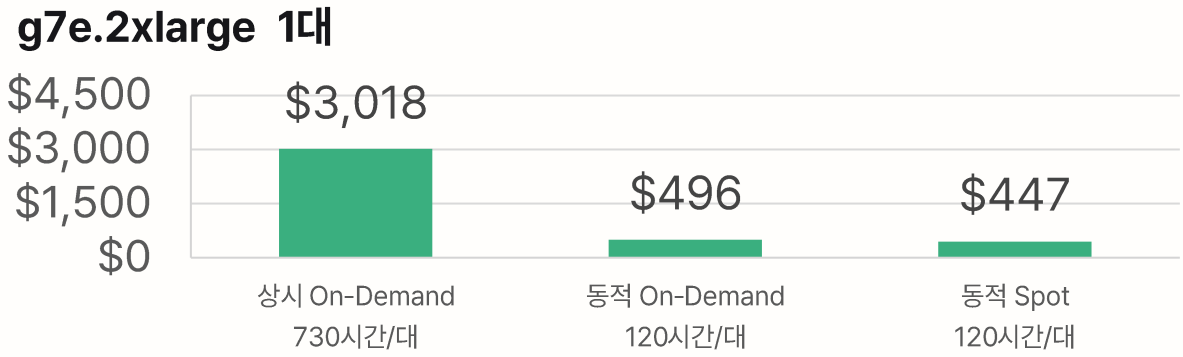
## 인프라 측면

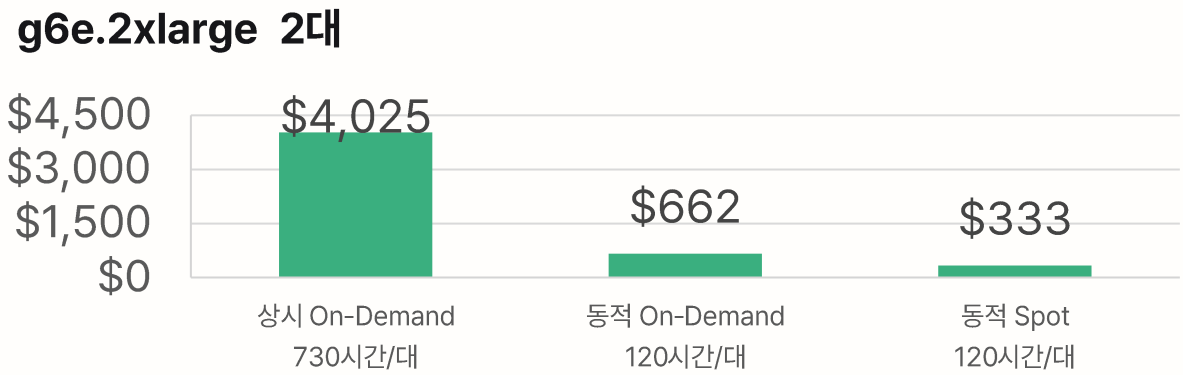
01 장비 구매 / 증설 부담 완화

02 운영자의 수작업 감소

03 GPU 활용도 증가

04 유휴 GPU 비용 감소




**83.6%**

동일 On-Demand 요금 대비 절감

# 요약 및 향후 계획

# 요약 및 향후 계획

## 요약

### 01 제작 공정을 연결하는 VARCO 3D

생성·AI 후처리·Workflow·API로 반복 제작 지원

### 02 EKS 기반 GPU 탄력 운영

KEDA·Karpenter로 증설·회수하고 MIG로 자원 활용
Spot·On-Demand와 멀티리전으로 GPU 수급 대응

### 03 AgentCore로 예측·장애 조사 보조

예열 시점 제안과 근거 기반 원인·대응안 정리

### 04 사용성과 운영 효율 개선 기대

대기·재시도 완화와 동시 처리 능력 확대
유휴 비용·장비 증설·운영 수작업 부담 감소



# 요약 및 향후 계획

## NC AI B2C 서비스 도입

NC AI API Platform

VARCO 3D

VARCO Sound

# 요약 및 향후 계획

## 모두의 AI




X

NC AI

VARCO

출처: https://zdnet.co.kr/view/?no=20260911184448

aws FOR GAMES

# Thank you

GS Neotek | MEGAZONE CLOUD