
AWS
re:Invent
DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

PUBG STUDIOS | KRAFTON

GAM311

The evolution story of game architecture: PUBG: Battlegrounds, KRAFTON

JungHun Kim
DevOps Team Lead
KRAFTON

Minsuk Kim
Solutions Architect
Amazon Web Services

Minwook Chun
Technical Account Manager
Amazon Web Services

aws

Agenda

01 Modernizing gaming architecture

02 PUBG’s evolution story

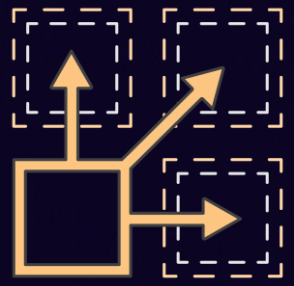
03 Lesson’s learned

04 Summary

Modernizing gaming architecture

Modernization

1



Technology & architecture

Independent business functions

2

People, process, & culture

Organized for value

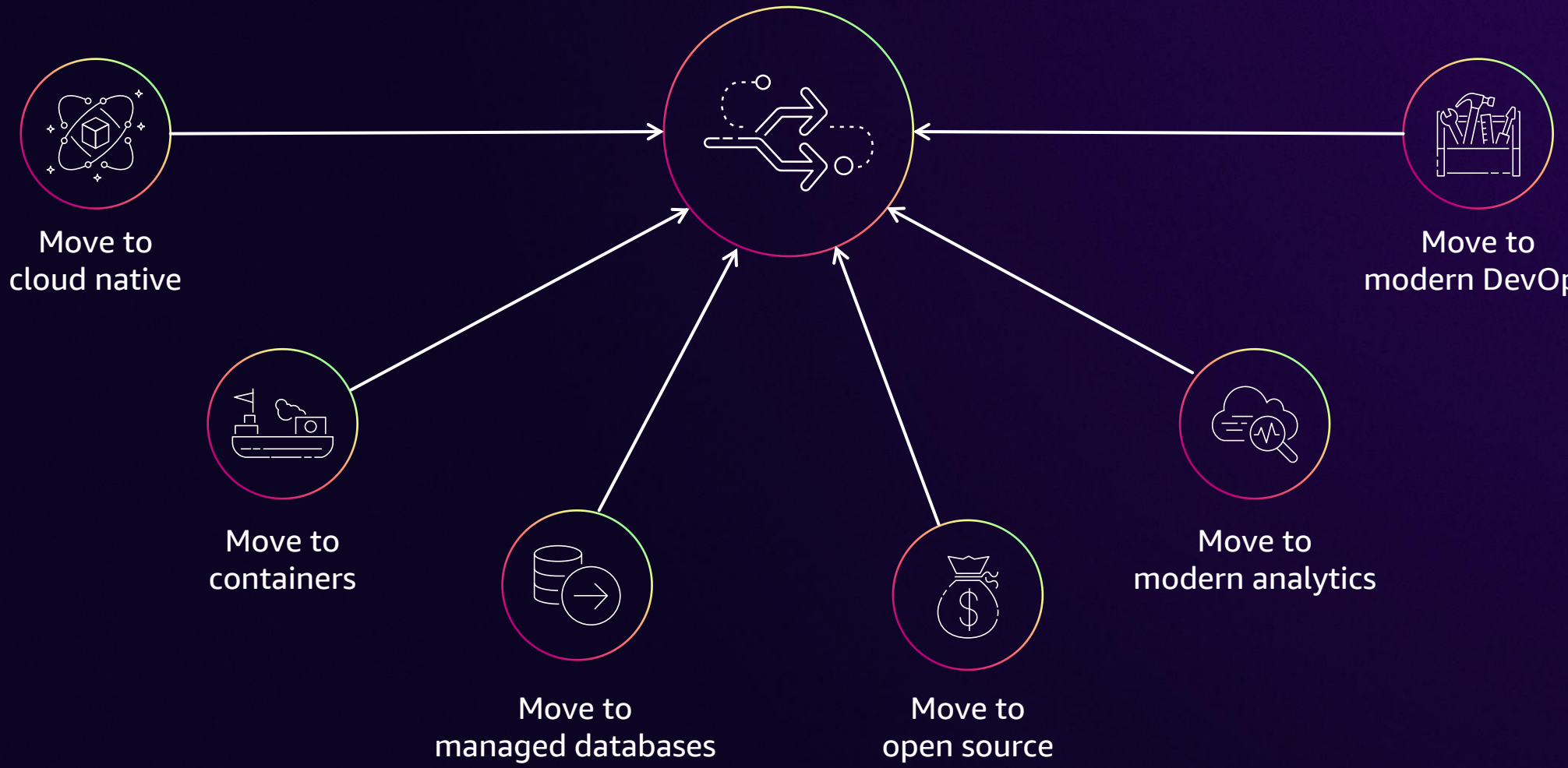
3

Ops & governance at scale

Automate, enable, & self-service

Modernization is the refactoring of legacy technology by combining modern infrastructure, architecture, organization patterns together to maximize resiliency, engineering efficiency, and business agility

Modernization pathways



Move to purpose-built chipsets – Adopt Graviton, Inferentia, Trainium

PUBG’s modernization journey

November 2018

Start

October 2019

Moved ‘Lobby’ servers to Amazon Elastic Kubernetes Service (EKS)

July 2021

Moved ‘Session’ servers to Amazon Elastic Kubernetes Services (EKS)

June 2023

Moved ‘Lobby’ servers to AWS Graviton

PUBG’s evolution story: Overview

PUBG: BATTLEGROUNDS

A first-person shooter (FPS)/
third-person shooter (TPS) battle royale
that pits 100 players against each other

Top 3 games by current players in STEAM

Major components

- Lobby
- Session gameplay


TEEN
T
ESRB

Components: User's perspective

Lobby

Entry point for entire users

Matchmaking for gunplay session

Store, inventory, customize

Achievements and rewards

Components: User's perspective

Session

100 players gunplaying session

Farming, gunplay, survival, and win

Training room for practice

Arcade mode (e.g. zombie mode)

Components: Engineer's perspective

Lobby server

Single region (us-east-1)

Microservices (.NET webservers)

Stateless servers
(with storage backends)

Session server

Distributed across multiple regions

Unreal dedicated server

Stateful server
(without persistent storage)

Common challenge areas

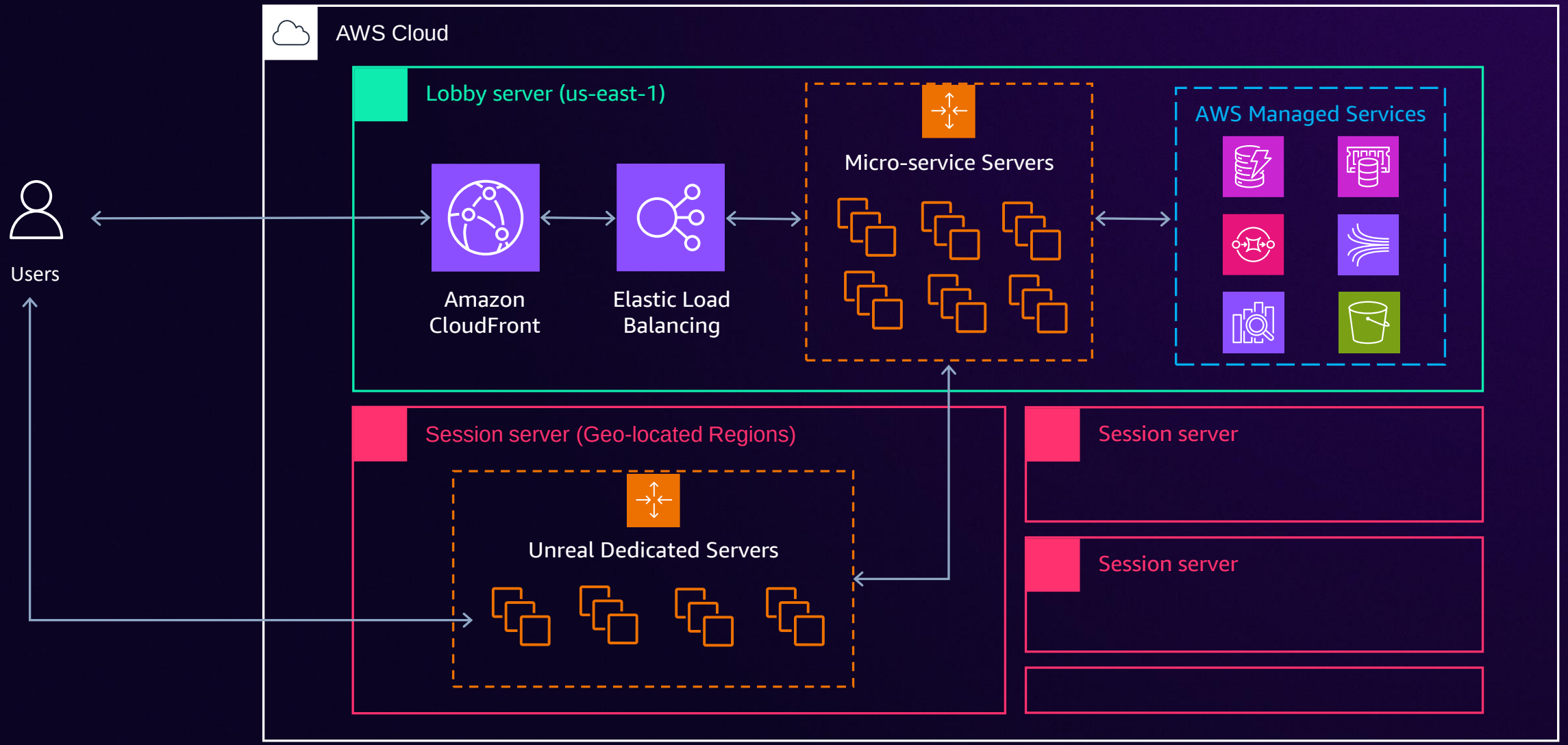
Efficiency

Staff productivity

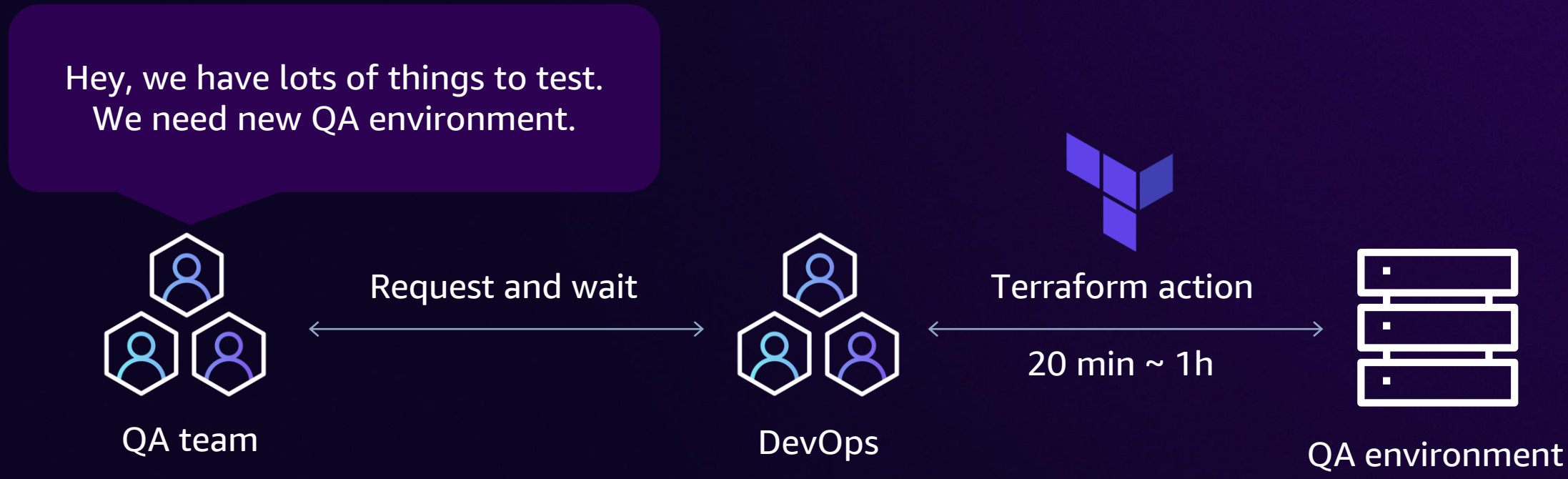
Agility

PUBG's evolution story: Moving lobby servers to containers

Legacy architecture overview



QA environment creation workflow

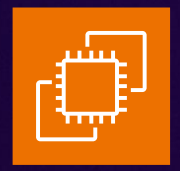


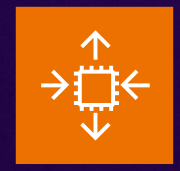
AWS services used in each QA environment


Amazon VPC

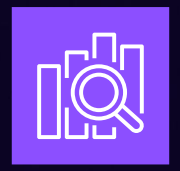

Amazon Route 53


Amazon CloudFront

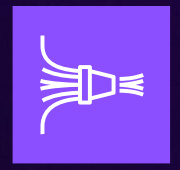

Amazon EC2


Amazon EC2 Auto Scaling


AWS IAM


Amazon OpenSearch Service (Amazon ES)


Amazon Kinesis Data Streams


Amazon Data Firehose


Elastic Load Balancing


Amazon SQS


Amazon S3


AWS CodeDeploy


Amazon DynamoDB


Amazon ElastiCache

Requirements for new workflow

Efficiency

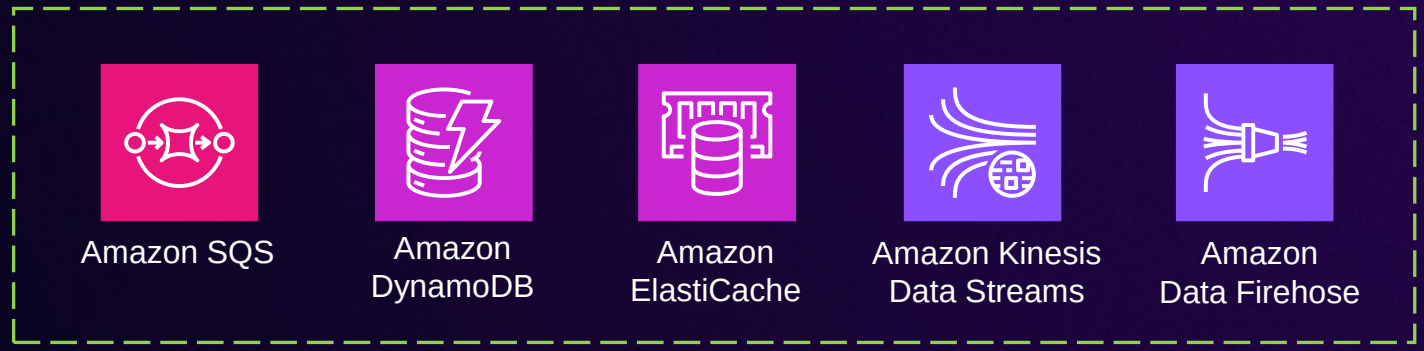
Standardization

Automation

Moving QA environment to Kubernetes

Dedicated resources

Hard to share

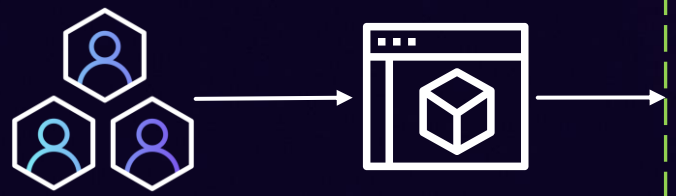


Shared resources

Easy to share

Moving QA environment to Kubernetes

Dedicated resources



Web UI

✓ QA ✓ Designer
✓ PM ✓ Developer

Redis Server (Pod)

DynamoDB Local (Pod)

Kubernetes resources for microservices

- ✓ Self service for QA Infra
- ✓ Short creation time (< 5min)
- ✓ Fully configurable

Shared resources

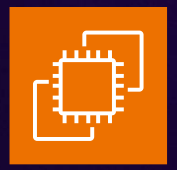
DevOps

Terraform

Amazon VPC

Amazon Route 53

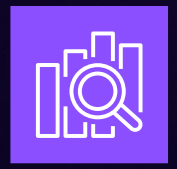
Elastic Load Balancing



Amazon EC2

Amazon EC2 Auto Scaling

AWS IAM



Amazon OpenSearch Service

Amazon S3

Amazon CloudFront

AWS EKS

Migration strategy differences

QA environment

Allow database flush

Allow big bang migration

Allow downtime

Production environment

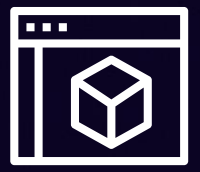
NEVER flush database

ALWAYS migrate gracefully

ALWAYS prepare rollback plan

Downtimes are usually NOT ALLOWED

Moving production environment to Kubernetes



Internal deploy tool
(Extend from QA SaaS tool)

Terraform

Dynamic resources

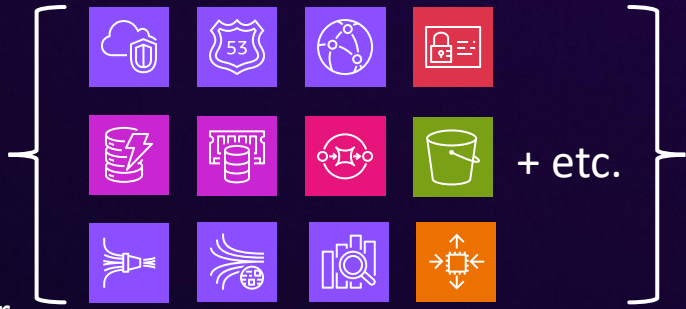
Deployment for
microservices

Virtual resources like
- ✓ Service
- ✓ Ingress
- ✓ Horizontal Pod Autoscaler
- ✓ Pod disruption budget
- ✓ Etc.

Static resources (partially shared with EC2)

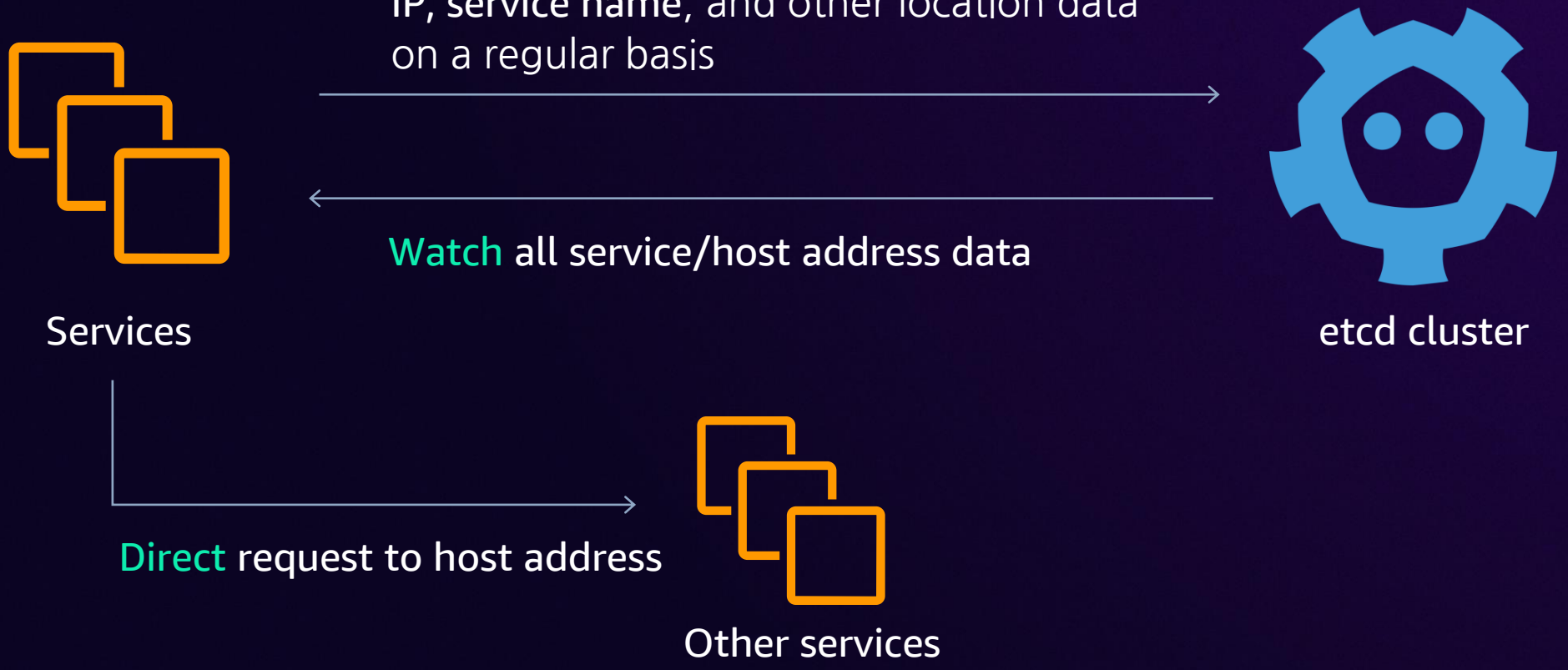
Amazon EKS

Helm charts for
Kubernetes controller



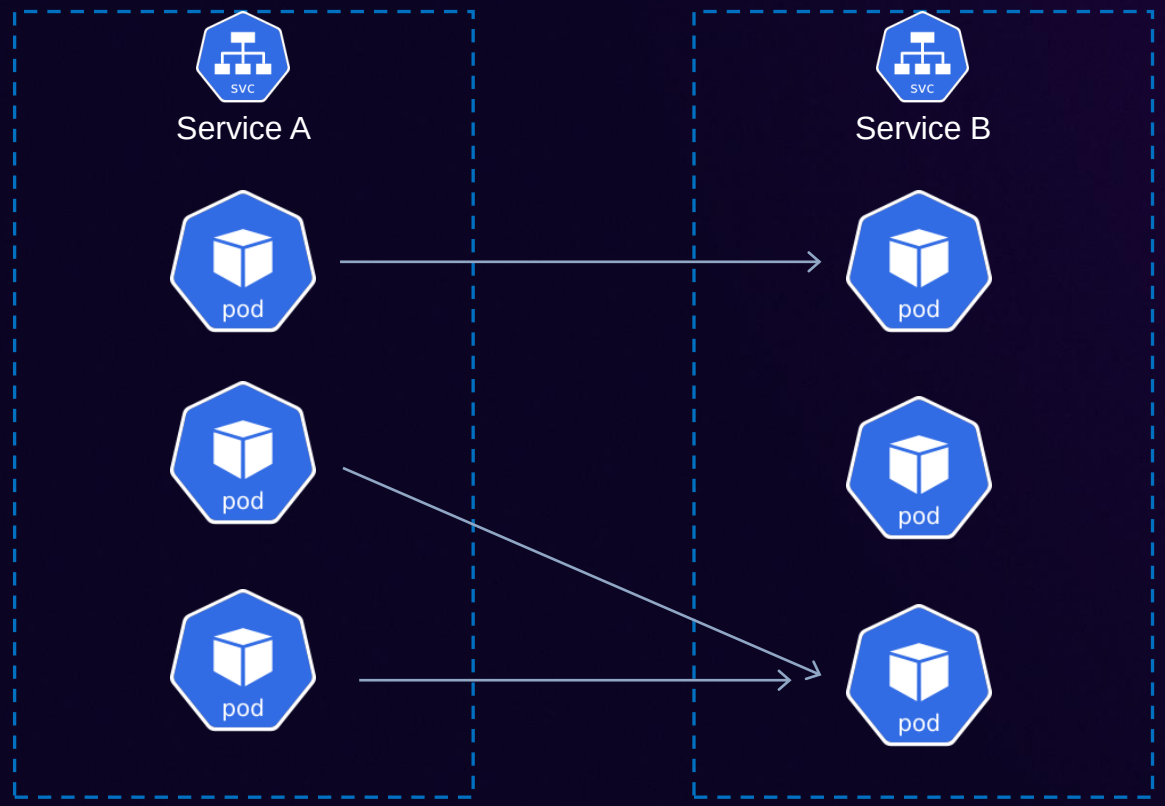
AWS Resources

Migration: Service discovery

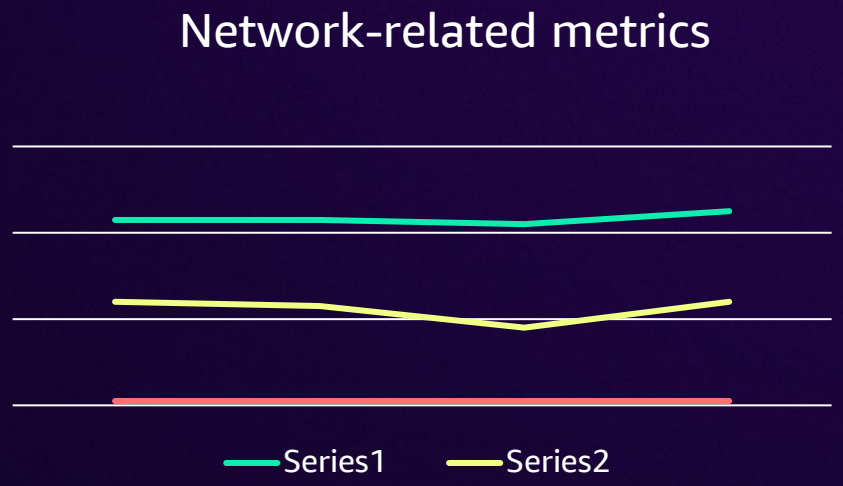


Migration: Traffic shifting

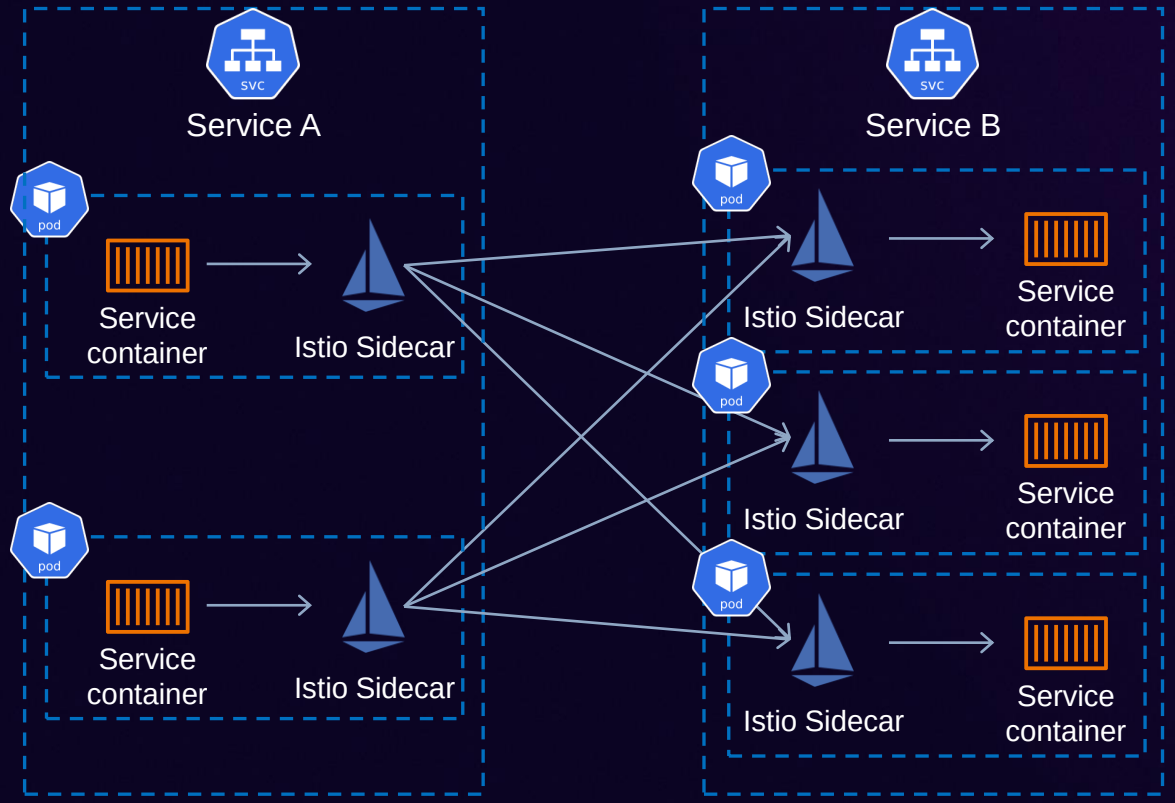
Problem: Unbalanced traffics



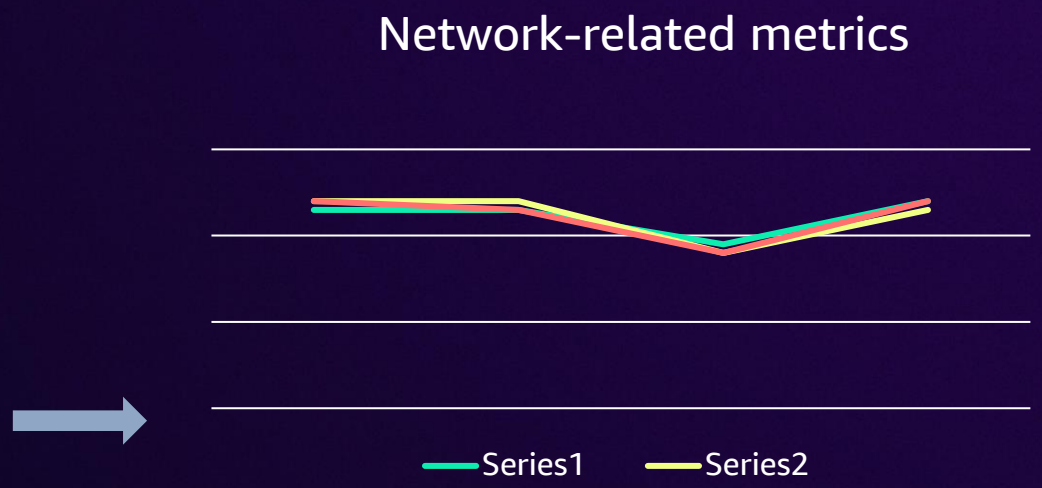
With connection pulling



Moving to service mesh



With service mesh



Additional benefits

- ✓ Dynamic traffic management
- ✓ Security
- ✓ Observability

Benefits from migrating to Kubernetes

Improved
operation excellency

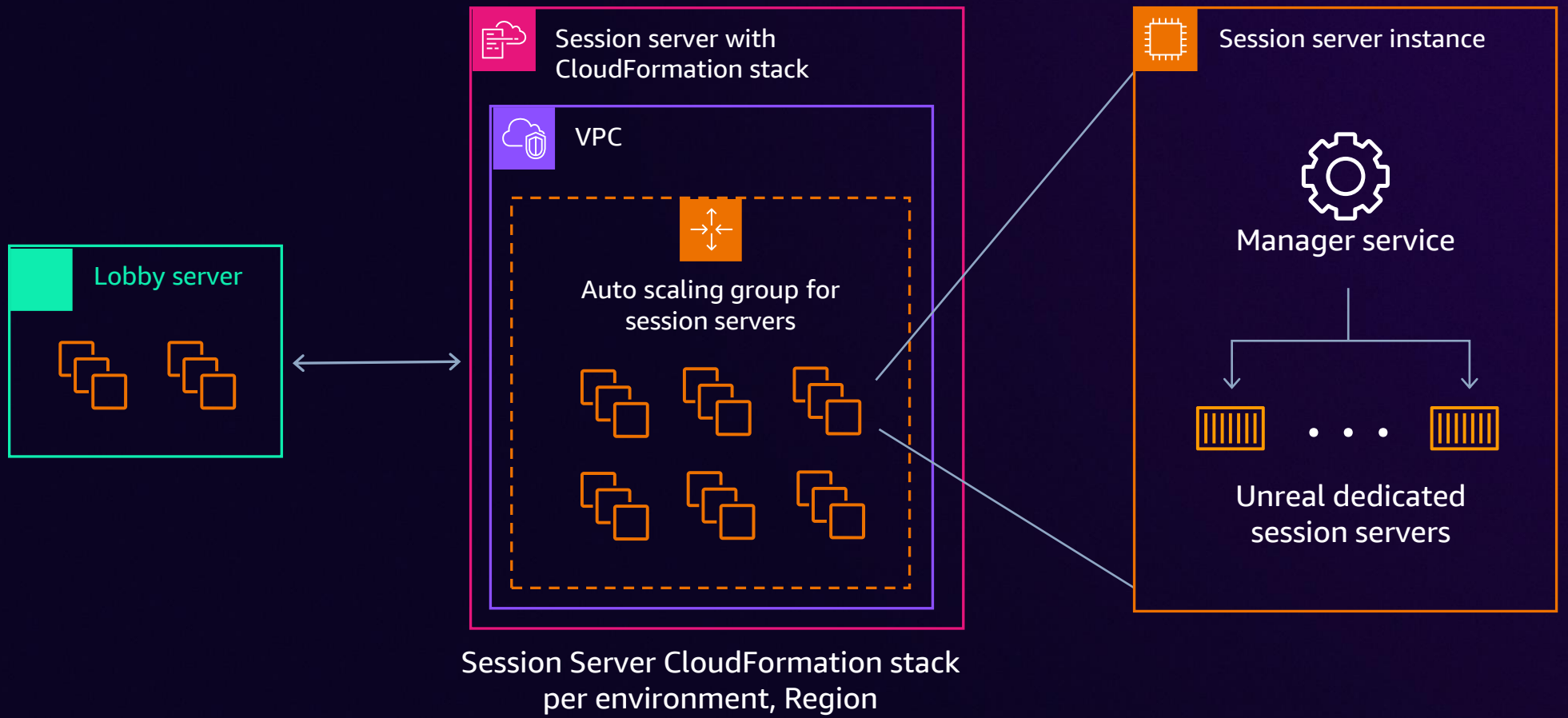
Enhanced
traffic management

Pre-requisite

Knowledge of Kubernetes and its ecosystems

PUBG's evolution story: Moving Session servers to Containers

Legacy architecture



- ✓ Manage lifecycle
- ✓ Process logs
- ✓ Capacity mgmt.

- ✓ Handle user traffic

Requirements

Standardization

Eliminate legacy stack and standardize to reduce overhead

Use suitable OSS to simplify session server management

Scalability

Easily scalable to hundreds of thousands of sessions

Relatively consistent response times at large scale

Cost efficiency

Resource scaling and buffering to match demands

Maximizes VM efficiency through resource bin-packing

Standardization and scalability: Agones

Open source multiplayer dedicated game server scaling and orchestration platform on top of Kubernetes

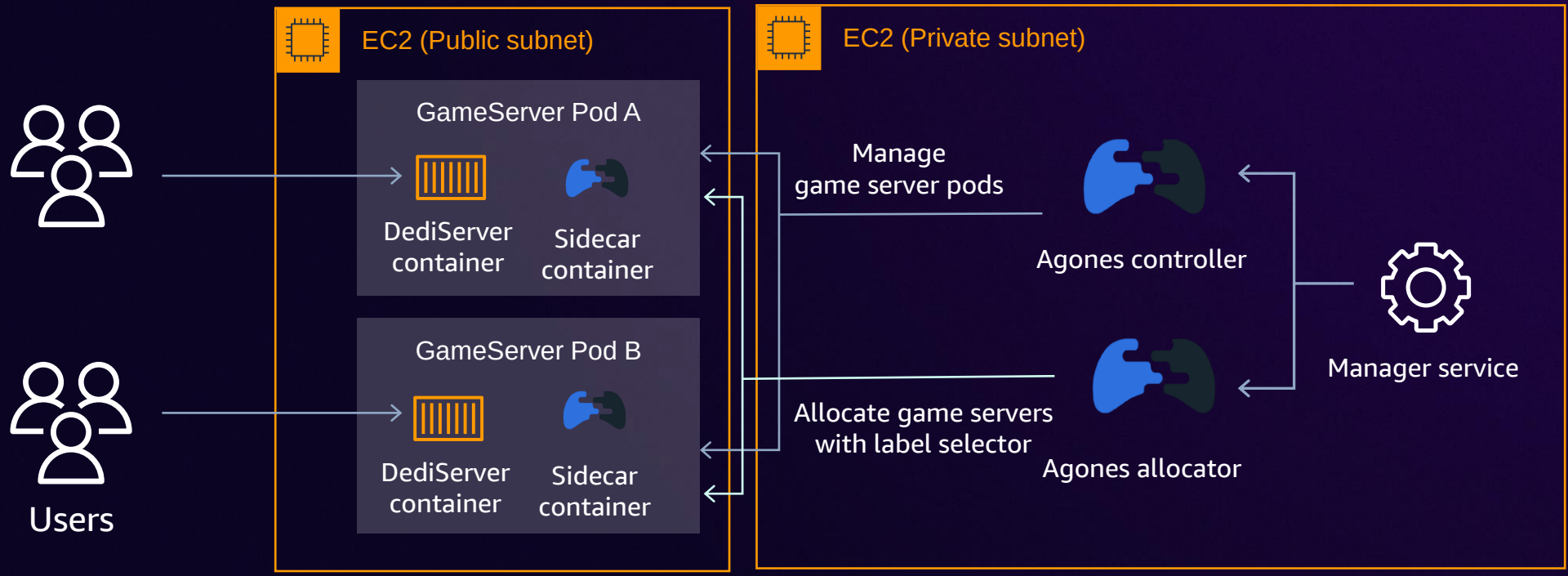
Manage game servers like deployment

- 1 GameServer = 1 Pod
- Manage GameServers using Fleet: Set of GameServers
- Support FleetAutoscaler to manage capacity

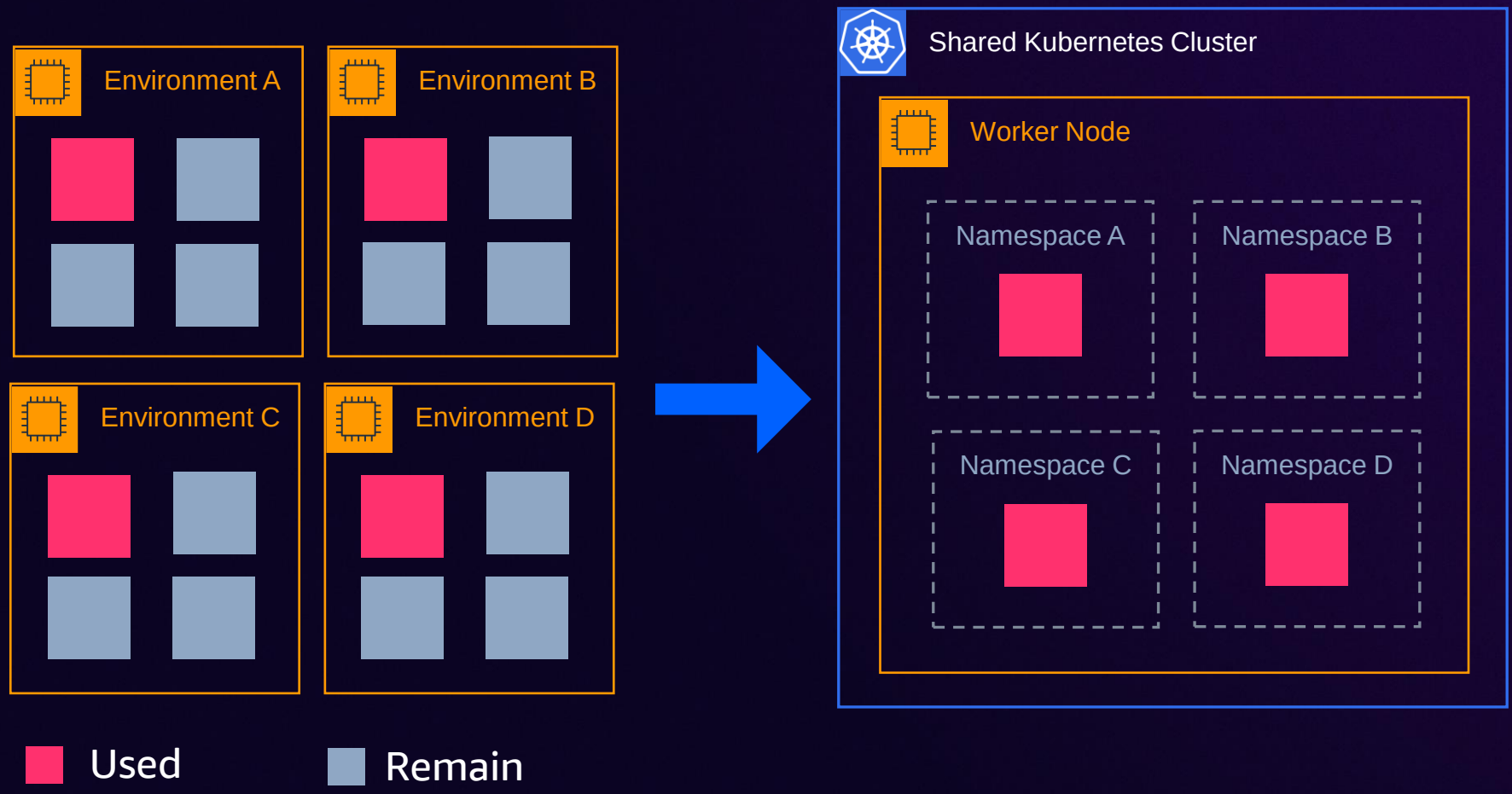
Easily integrated with K8s ecosystem

- Manage GameServer Pod as Kubernetes way

Architecture with Agones



Cost efficiency: Bin-packing and capacity management



GameServer scaling

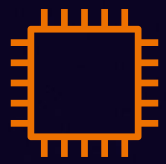
✓ FleetAutoscaler

Instance scaling

✓ ClusterAutoscaler

✓ Karpenter

Challenges: Bootstrapping time



Instance provisioning

(1~3 minutes)

Instance bootstrapping

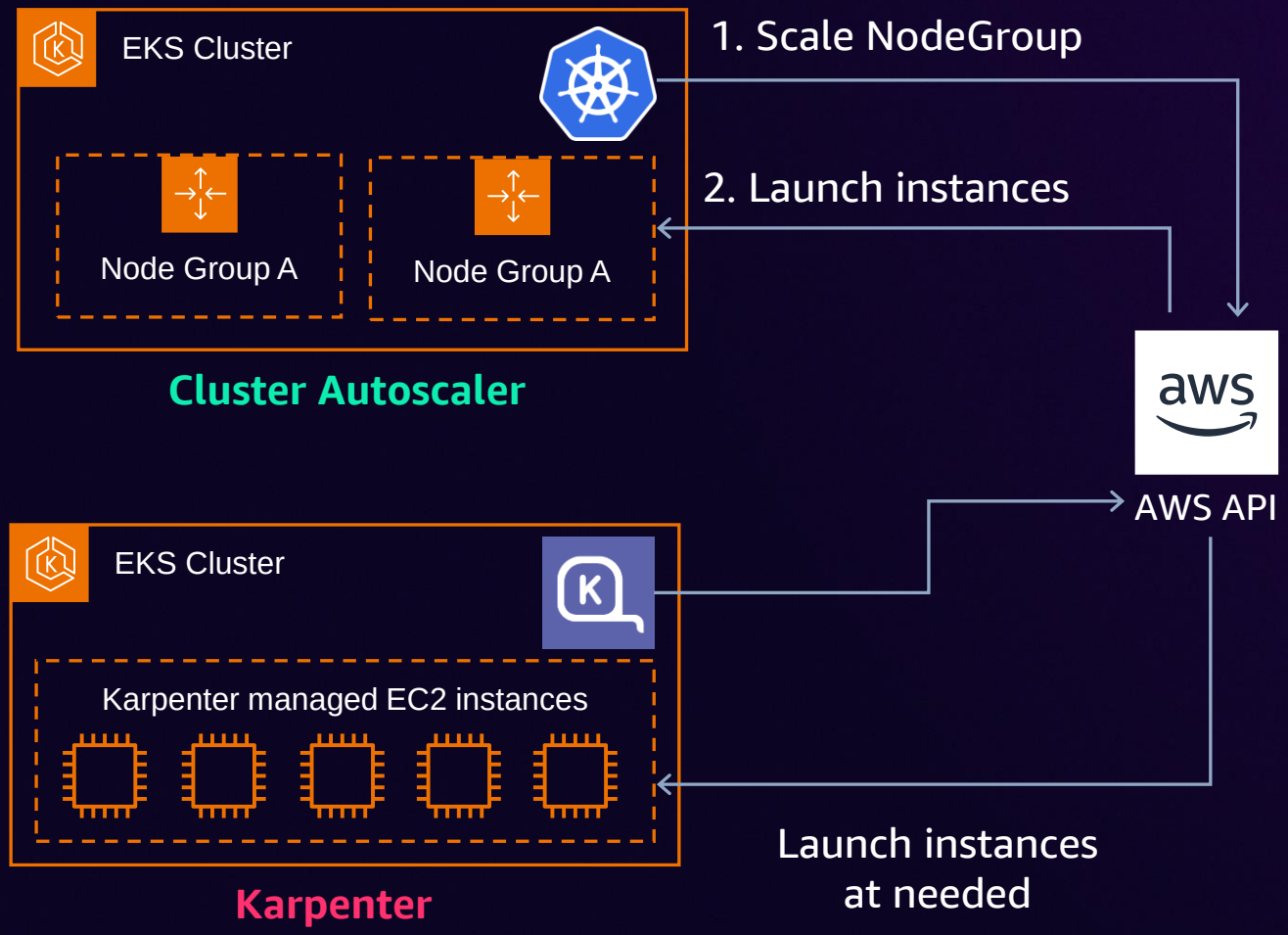
(2~3 minutes)

Pod provisioning

(5~10 minutes)

10~15+ minutes

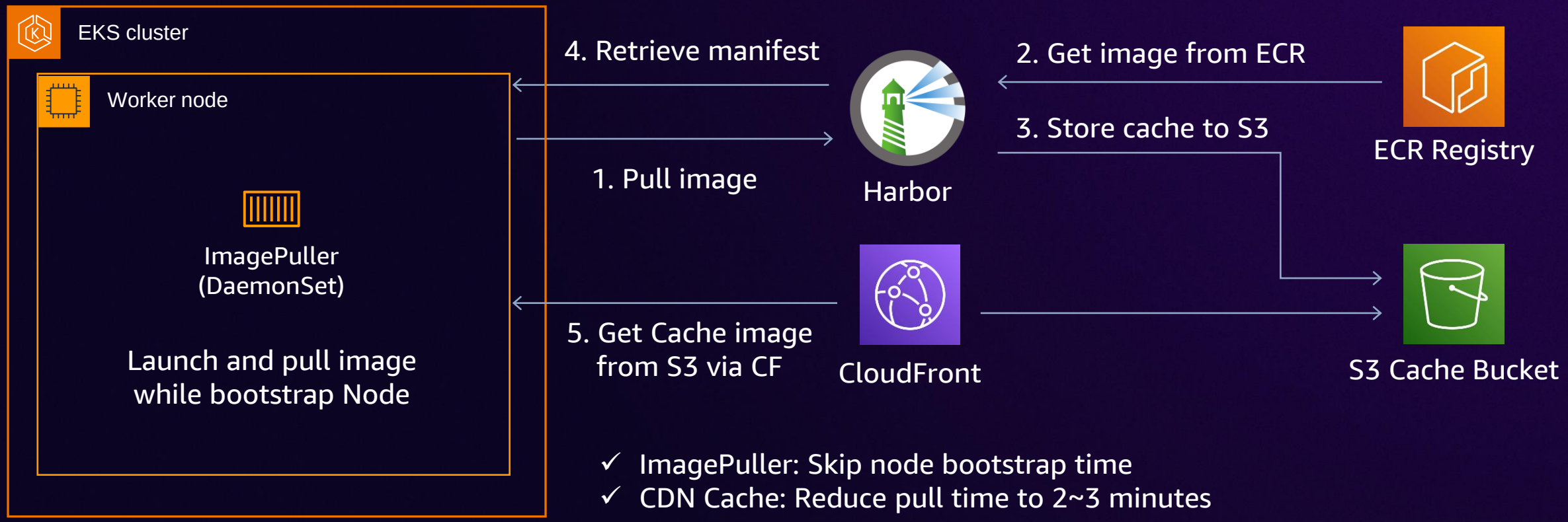
Our approach #1: Adopting Karpenter



Benefits

- ✓ Reduce EC2 launch delay
  - Remove EKS NodeGroup and Auto Scaling Group overhead
- ✓ Easy node management
  - Dynamic, groupless node provisioning
  - Automatic node sizing & bin-packing
  - Drift detection, expiration

Our approach #2: Container registry proxy



- ✓ ImagePuller: Skip node bootstrap time
- ✓ CDN Cache: Reduce pull time to 2~3 minutes

15+ minutes ➔ 3~4 minutes

Benefits and trade-offs

Benefits

1. Reducing QA environment operating costs through resource sharing
2. Enhancing operation efficiency by integrating the infra stack with Kubernetes

Trade-offs

1. Difficult to manage provisioning concurrency
2. Unable to adjust resource dynamically (<1.32 on EKS)

PUBG’s evolution story: Moving to AWS Graviton

Why AWS Graviton?

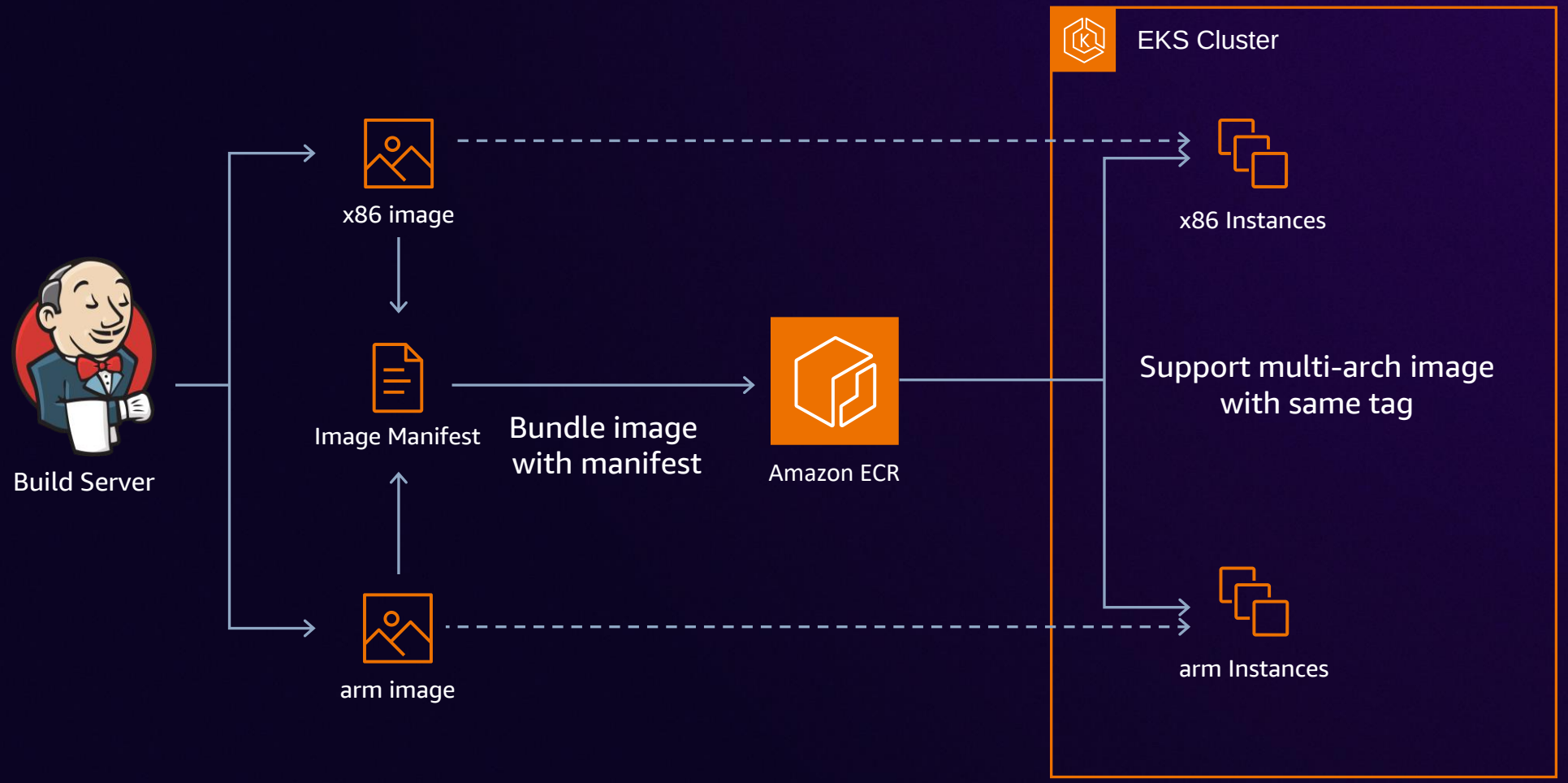
Up to 40% better price-performance for a broad spectrum of workloads

Up to 20% less expensive than comparable x86-based instances*

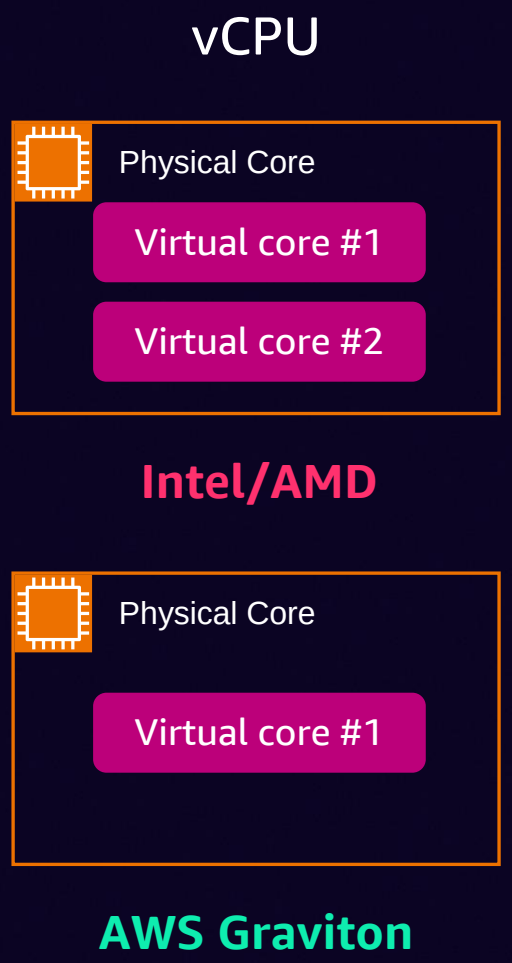
Up to 60% more energy efficient vs. comparable x86-based instances

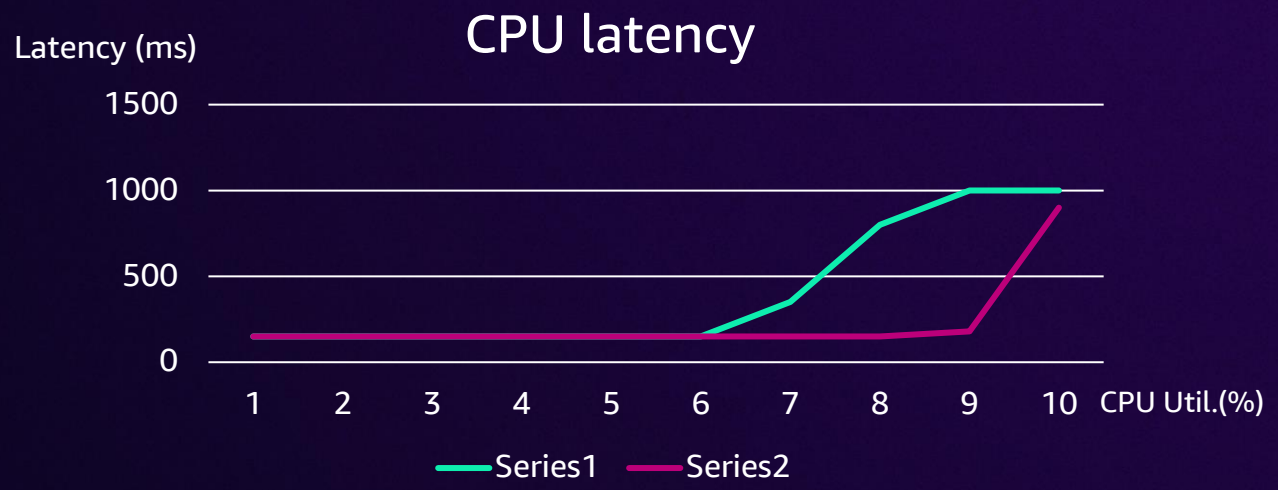
* Based on on-demand public pricing

Multi-architecture container images



Benefits





Lower CPU latency under high CPU loads

- Higher target CPU utilization
- More density for server pods

➔ +35% price performance

Remaining Challenge: Session Server on Graviton

Unreal Dedicated Server

+

Lots of Libraries
Internal & 3rd Party

Lessons learned

Lessons learned

Focus on what matters

Be aware of side effects

Prepare for rollbacks

Summary

Challenges and approaches

Efficiency

Staff productivity

Agility

Moving to Amazon EKS with a Karpenter / Agones / Istio

Reduce operational burden with small number of DevOps engineer

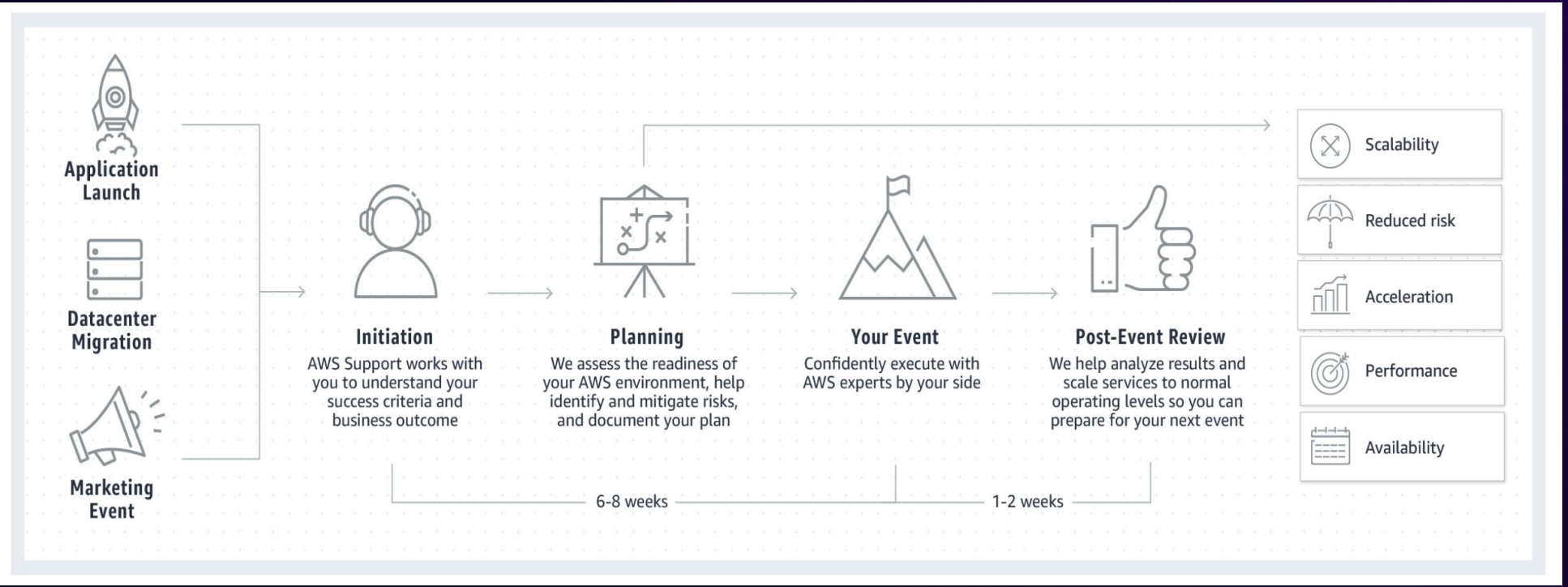
Moving to Graviton instances

Reduced 35% more EC2 instance cost

How AWS helped with the migration process

AWS Countdown (formerly Infrastructure Event Management)

- **From optimized planning to the major events with confidence**



How AWS helped with the migration process

Trusted advisor for customers to operate global gaming services in the cloud

- **Operation review with proactive services**

- If you have an **enterprise support** plan, we encourage you to explore our proactive services, including workload diagnostics and reviews, and operational workshops and deep dives.

How AWS helped with the migration process

Fast engagement on pain points

- **Transparent roadmap meeting – e.g., HostPort in EKS Windows networking**

We welcome your candid feedback and feature requests. Please feel free to share them with your **AWS account team**.

Thank you!

Please complete the session survey in the mobile app

JungHun Kim
DevOps Team Lead
KRAFTON

Minsuk Kim
Solutions Architect
Amazon Web Services

Minwook Chun
Technical Account Manager
Amazon Web Services