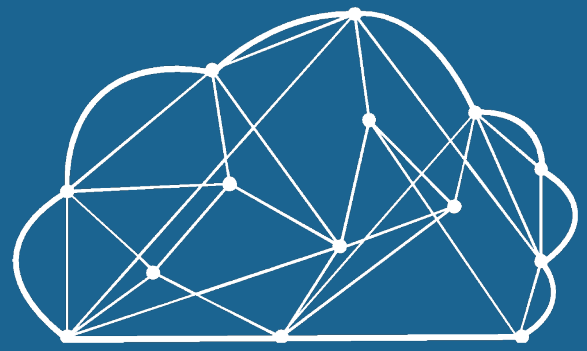
BESPIN GLOBAL



Amazon EKS Deep dive

OpsNow ArchOps 심선보(seonbo.shim@bespinglobal.com)

Version 1.0

CON304

Deep dive on Amazon EKS

Mike Stefaniak (he/him)
Sr. Product Manager, Amazon Elastic Kubernetes Service
AWS

Why EKS

Running and scaling Kubernetes can be difficult and requires significant investment

Securing Kubernetes increases the operational overhead of running applications

Applications need a native way to integrate with other AWS services securely and reliably

aws

EKS tenets

Security first

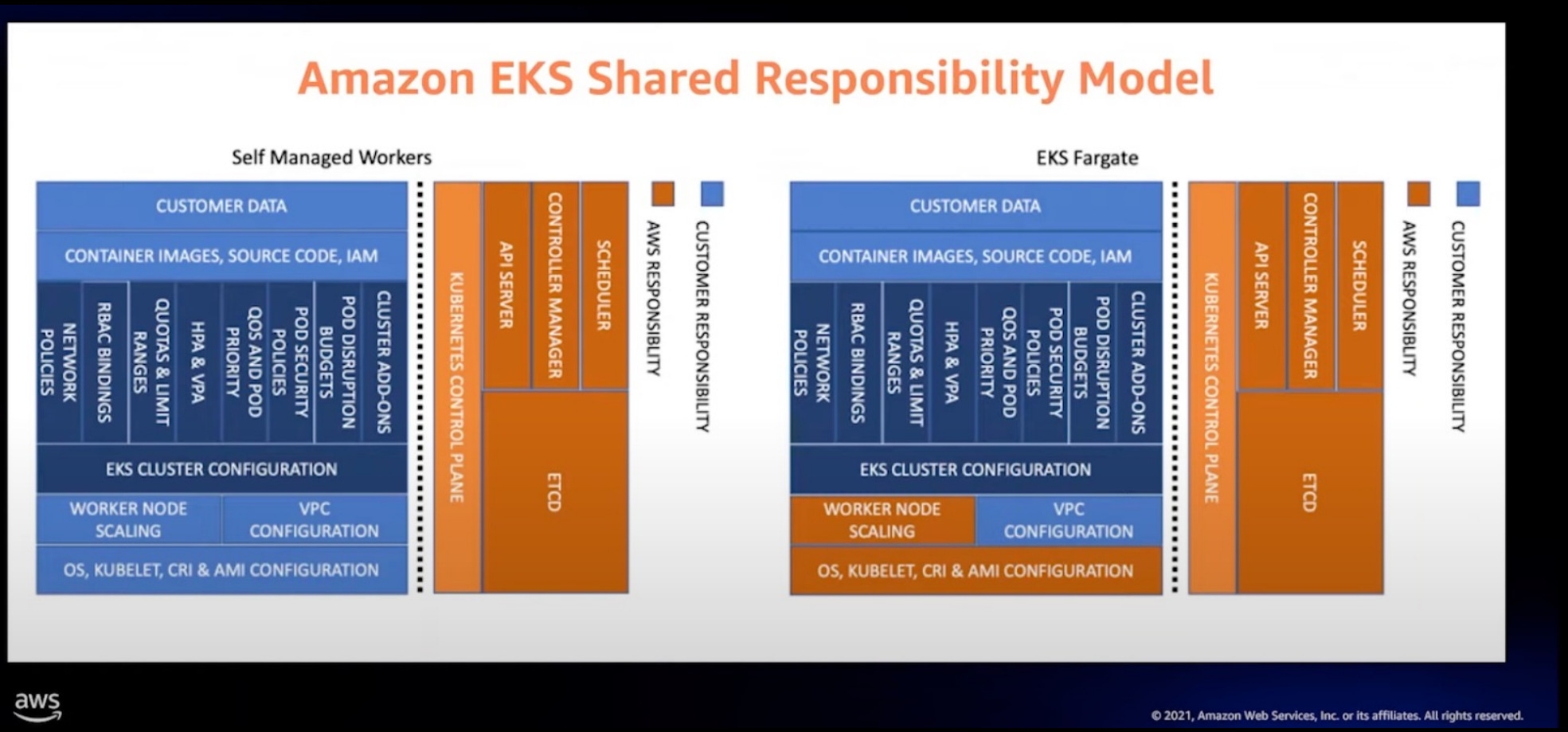
Built for production

Seamless cloud integrations

Native and upstream

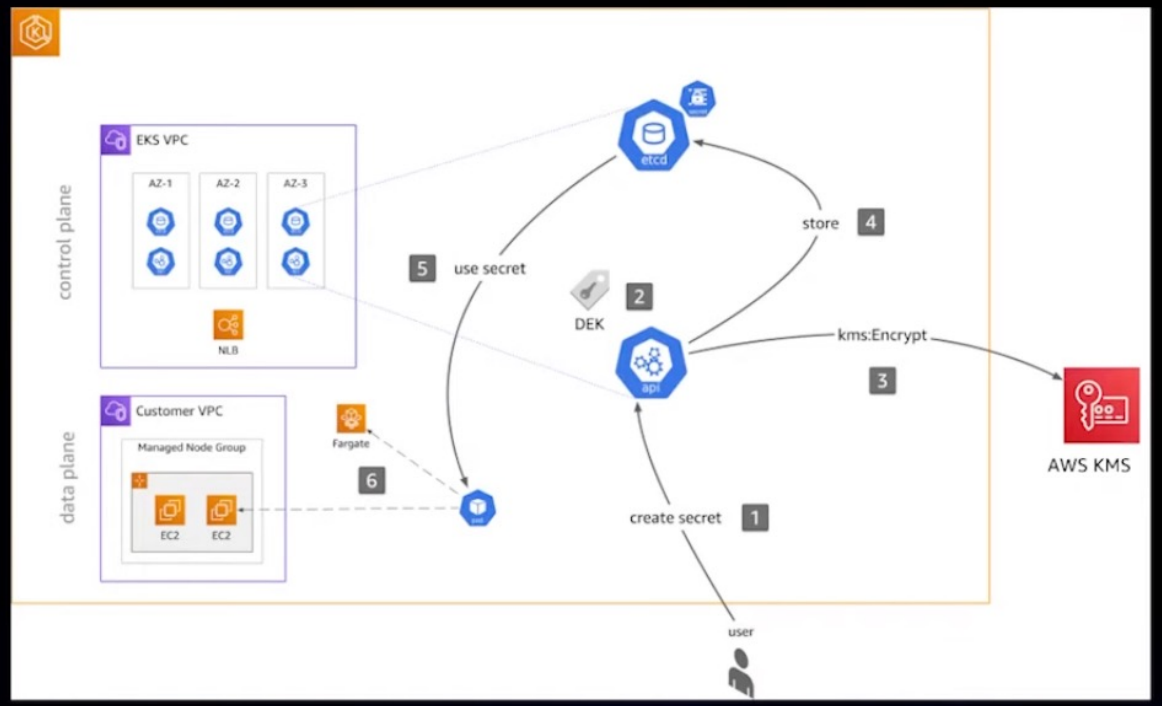
Committed to open source

Security



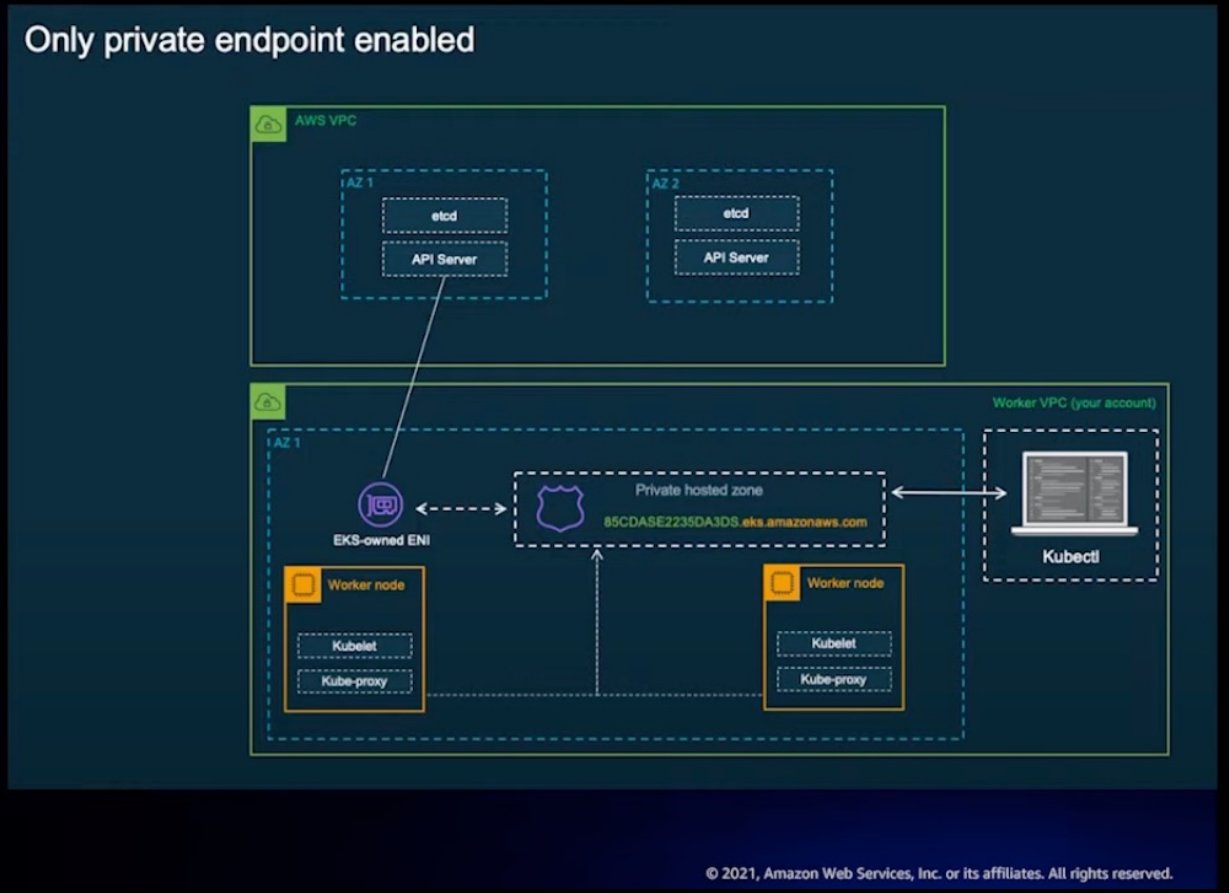
클러스터에서 민감한 데이터의 보호

- Encrypt your secrets with AWS Key Management Service (AWS KMS)
- Store secrets outside the cluster and retrieve using AWS Secrets Manager CSI driver
- Use AWS Certificate Manager Private CA Issuer cert-manager plugin to generate TLS certificates



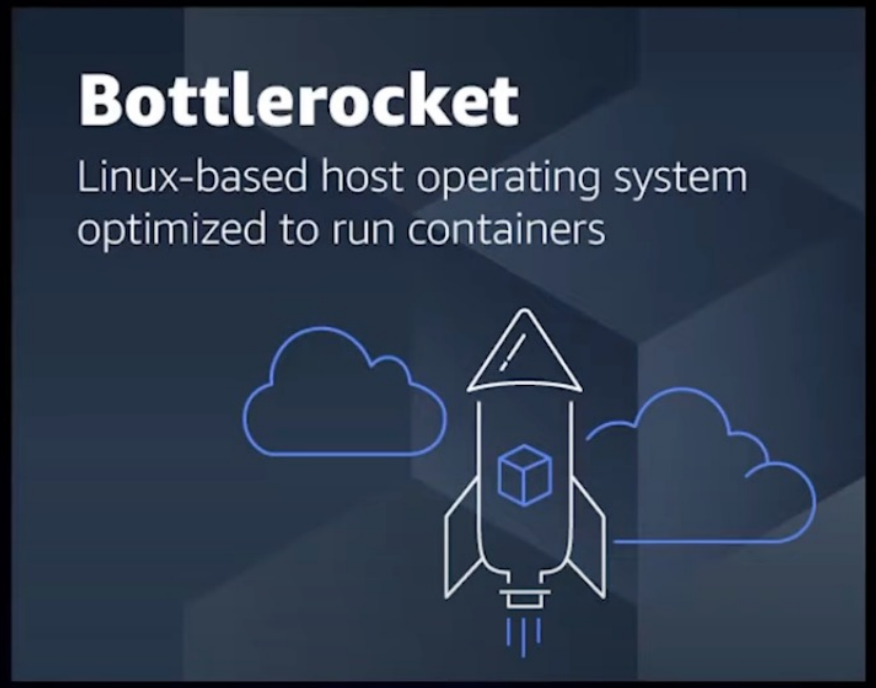
클러스터 액세스를 위한 앤드포인트 제한

- If possible, disable public endpoint
- If not, restrict access to public endpoint with CIDR blocks
- Private endpoint and private only subnets – if no internet communication required



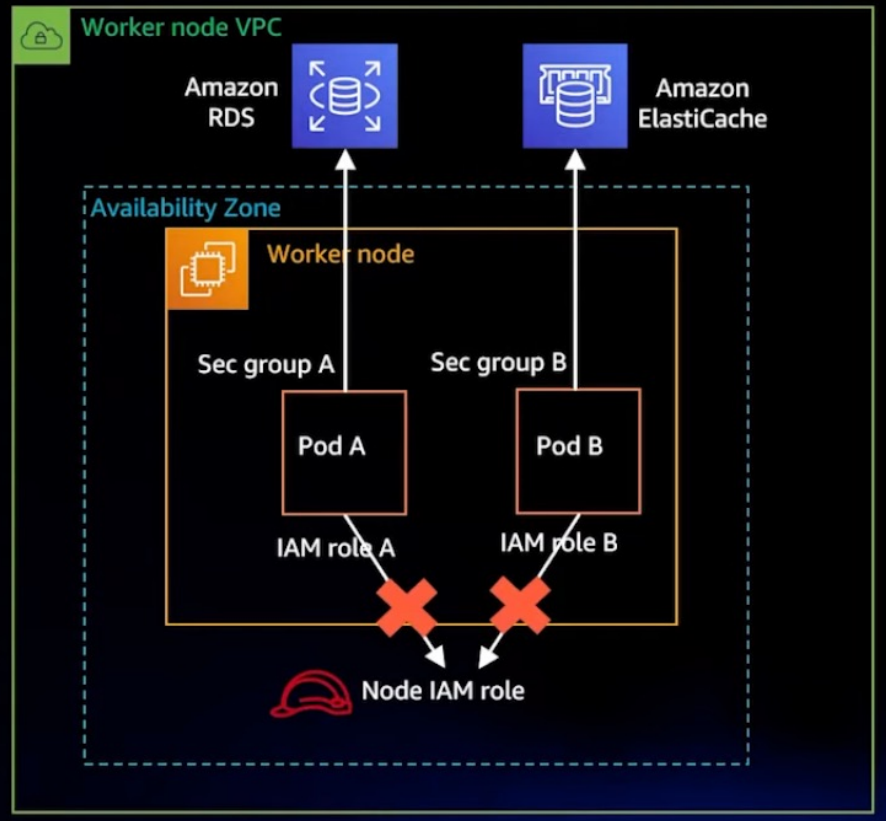
Worker Node를 위한 보안

- Use an OS optimized for containers, such as Bottlerocket
- Treat worker nodes as immutable
- Use AWS Systems Manager instead of SSH
- Validate custom AMIs with CIS Amazon EKS Benchmark
- Or, pass off this responsibility to AWS Fargate



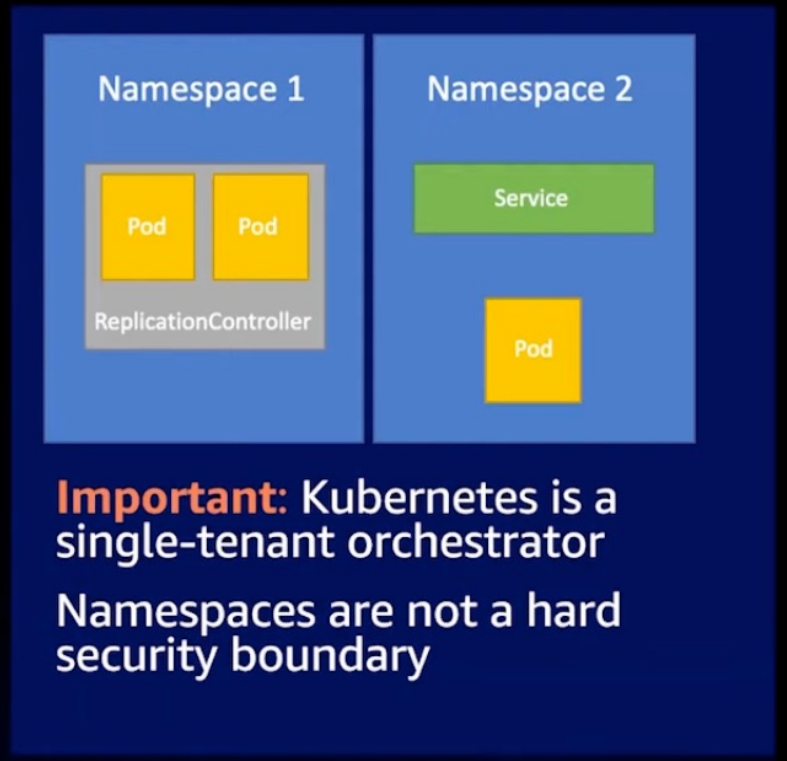
Pod를 위한 보안

- Scope down IAM permissions to pods instead of nodes with IAM roles for service accounts (IRSAs)
- Restrict pod access to instance metadata service (IMDS)
- Use K8S NetworkPolicy to restrict network traffic within cluster
- Use EKS SecurityGroupPolicy to restrict network traffic to AWS services like Amazon RDS



보안을 위한 추가적인 모범 사례

- Namespaces and role-based access controls (RBAC) to logically isolate trusted tenants
- Quotas and limit ranges to control consumption of compute resources
- API priority and fairness (v1.20+) to limit API server requests
- Pod priority and preemption for quality of service (QoS)
- Enforce policies and limit what dev teams can run with agents like Gatekeeper
- Default network policies that deny all cross namespace ingress and egress traffic
- Defense-in-depth with VM-level isolation using Fargate



컴플라이언스 준수

Payment Card Industry (PCI) Security Standard

FedRAMP High

DOD Cloud Security Req's Guide (SRG) – On road map

Criminal Justice Information Service Security Policy (CJIS)

U.S. Health Insurance Portability and Accountability Act (HIPAA)

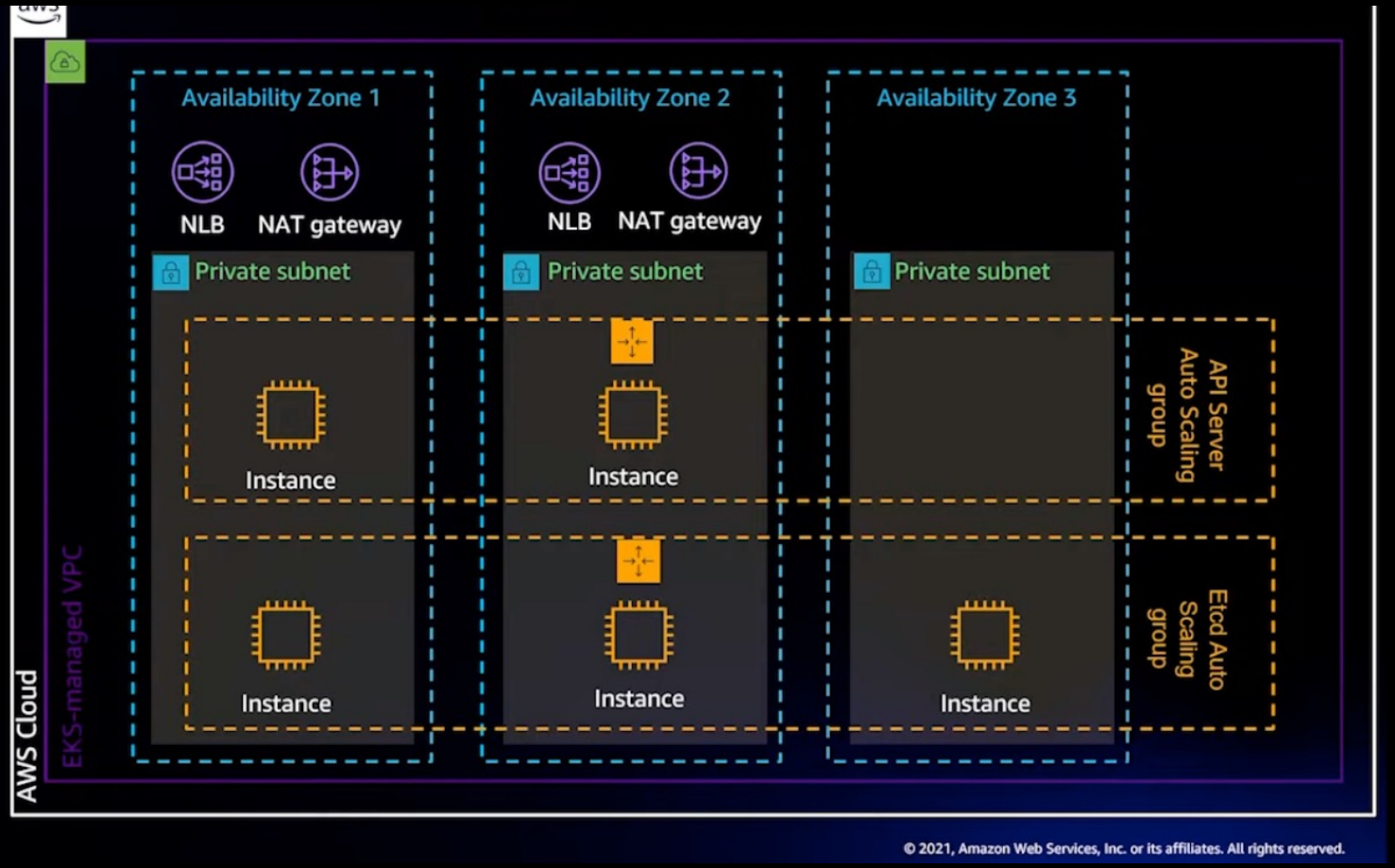
Federal Information Processing Standard Pub (FIPS) 140-2

SP 800-53 (rev 4)
SP 800-171

Health Information Trust Alliance Common Security Framework (HITRUST CSF)

Reliability

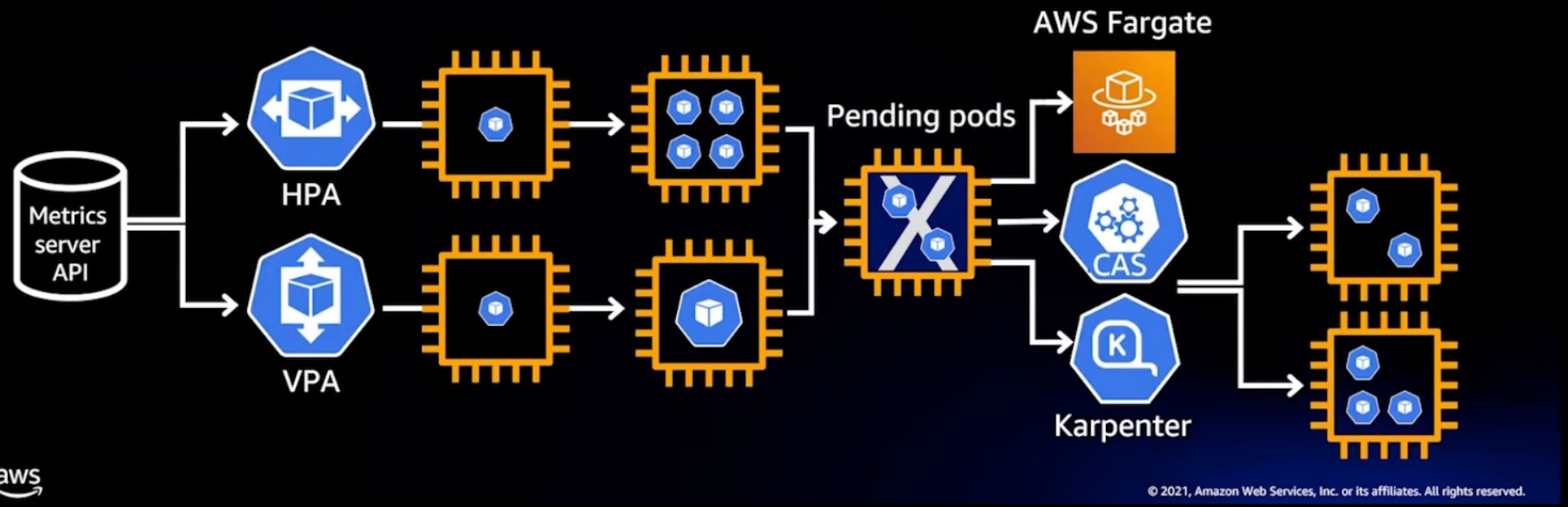
- Survive single-AZ events
- Highly available NLB endpoint
- Rolling control plane upgrades
- Automated etcd snapshots
- 99.95% SLA
- 24x7x365 support



서비스 워크로드를 위한 EKS 운용

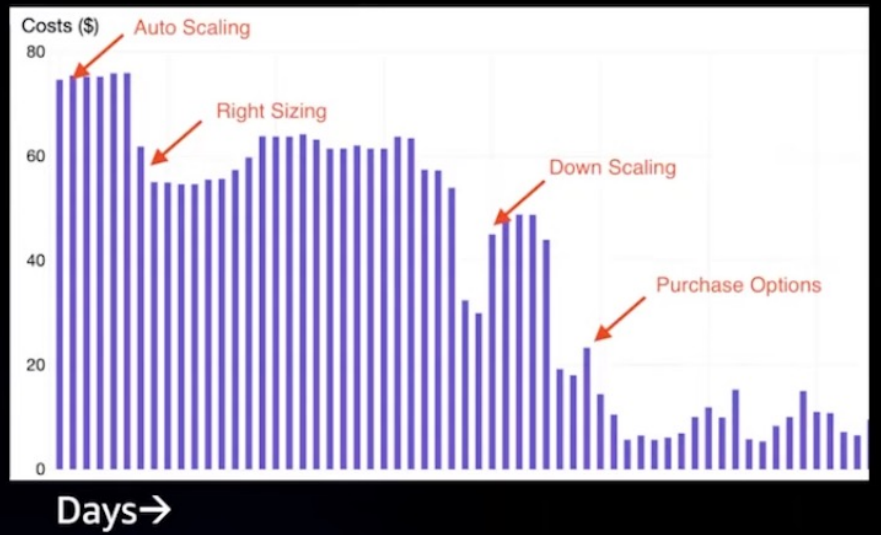
Efficiency - Right Size

- Scale pods horizontally/vertically based on usage
  Set pod resource requests close to actual utilization
- Scale node resources based on pod requirements
  Limit overprovisioning of compute capacity



클러스터 비용 최적화

- Use Spot Instances with managed node groups and save up to 90% off on-demand pricing
- Use Cluster Autoscaler priority expander to prioritize lower-cost node groups
- Utilize Savings Plan for Amazon EC2/Fargate
- Move to latest generation EC2 instances
- Move workloads to EC2 Graviton
- Try out Karpenter in large clusters with fast scaling requirements



IPv6 기반 스케일링 전략

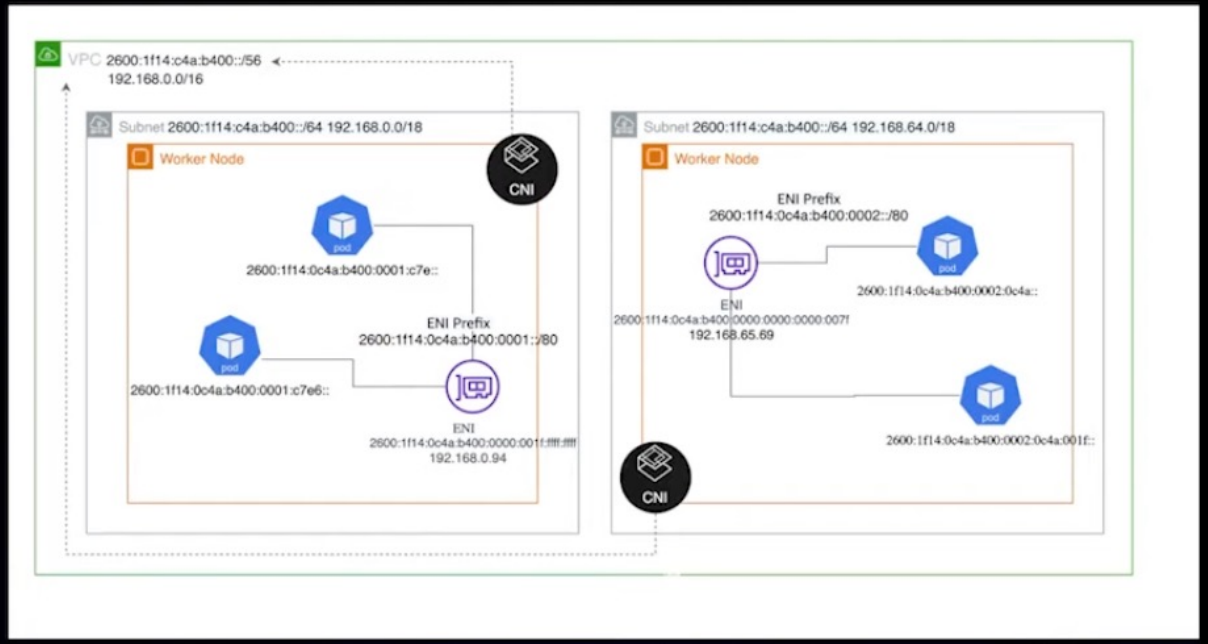
IPv6

- Scale far beyond IPv4 limits with globally unique IPv6 address assigned per pod
- Simplified pod-to-internet routing without network address translation
- Faster pod launch times
- Egress IPv4 traffic support

VPC CNI custom networking

- Add additional IPv4 CIDR blocks to existing VPC
- Run pods in subnets separate from primary ENI of worker nodes

EKS IPv6 networking setup



Important: Worker nodes can run in subnets separate from subnets registered during EKS cluster creation

적은 수의 큰 클러스터 운용 vs 다수의 작은 클러스터 운용

SPLIT WHEN YOU HAVE TO, MERGE WHEN YOU CAN

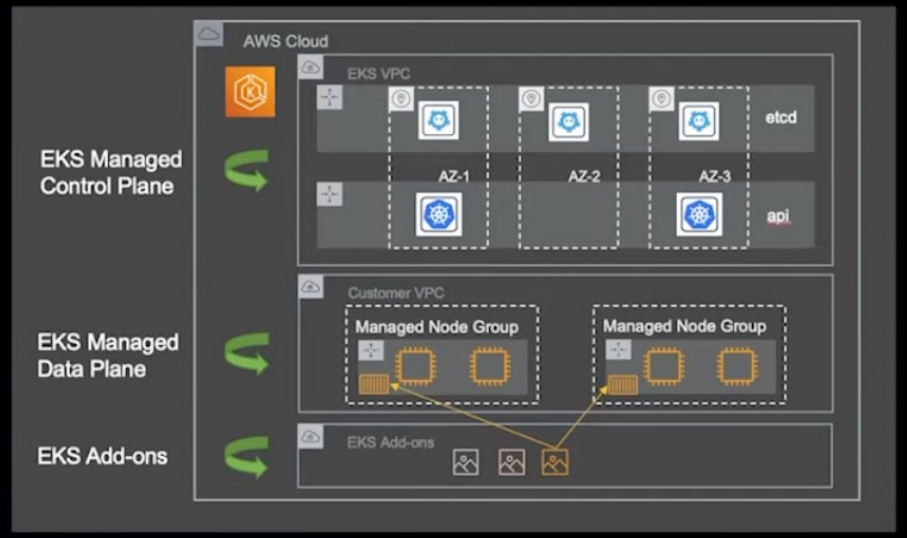
- At a minimum, separate clusters per environment
- Step 1 – Analyze constraints: untrusted tenants, organizational boundaries, area of impact concerns
- Step 2 – Optimize within constraints; fewer clusters leads to more efficient bin packing and use of resources

Source: https://learnk8s.io/how-many-clusters

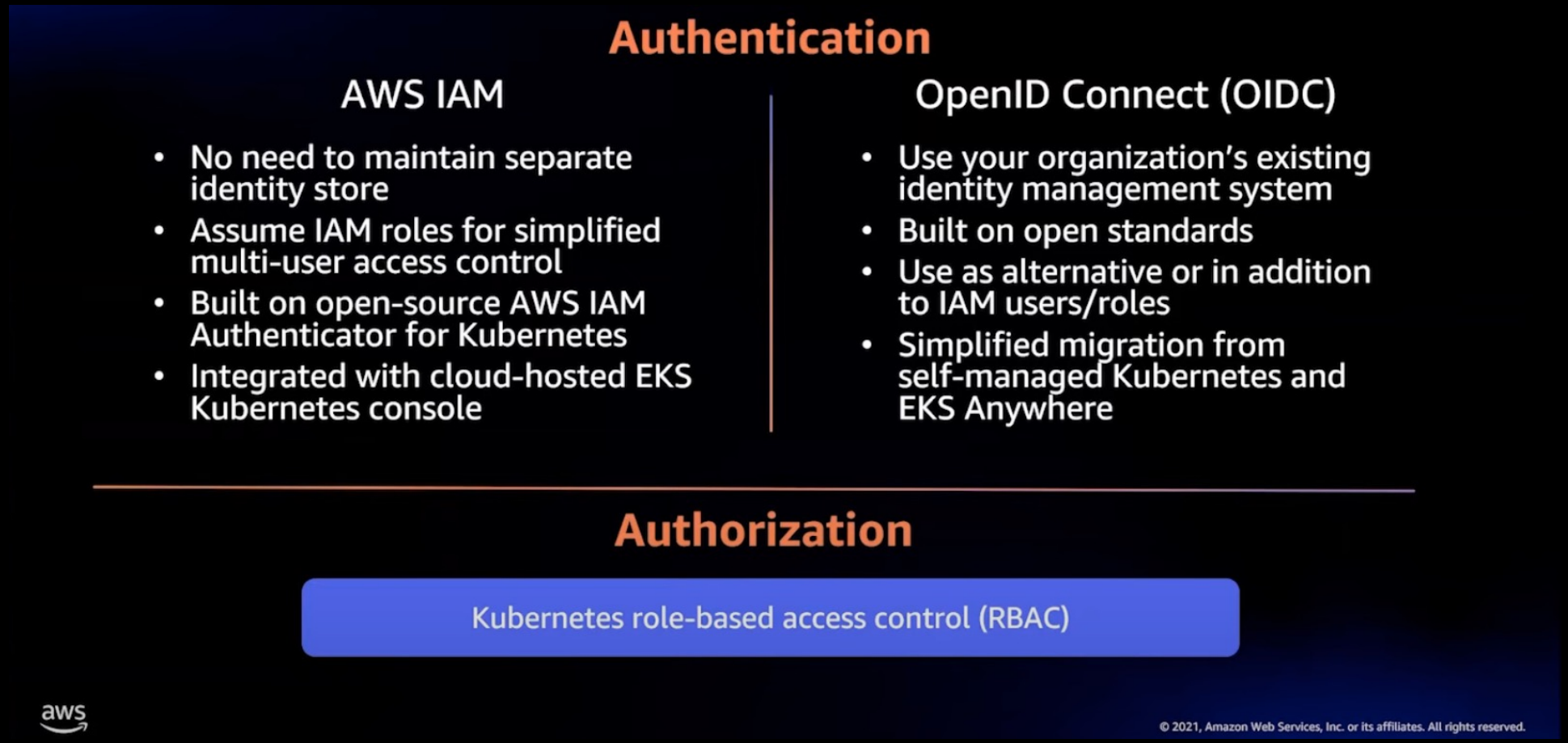
EKS 운용 - 버전 업그레이드

WHICH VERSION OF KUBERNETES DO I USE?

- Amazon EKS aims to be ~100–150 days behind upstream release
- You must plan for version upgrade at least yearly, ideally 2x-3x/yr
- Use EKS managed capabilities to simplify upgrades
- Run upgrade process in test env. first
- Check EKS and Kubernetes release notes
- Test your application manifests against deprecated/removed APIs



EKS 클러스터 액세스 제어 관리



EKS 클러스터 상태 모니터링

Control plane side

- Enable control plane logging for audit, API server, and authentication
- Collect and monitor control plane Prometheus metrics for API request latency

Recommendation: Run kube-state-metrics to easily surface cluster state information

Application side

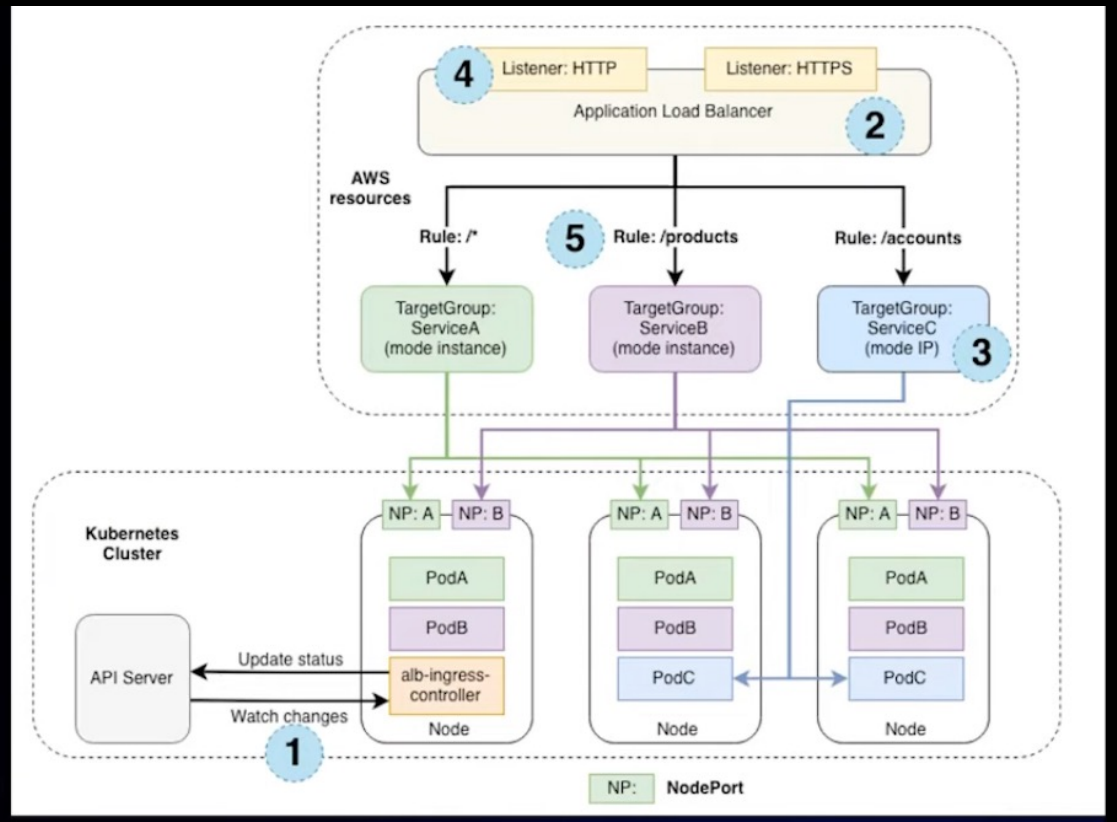
- EKS add-ons support for AWS Distro for OpenTelemetry
  - Output metrics/traces to CloudWatch, Amazon Managed Service for Prometheus, or partner destinations
- AWS Distro for Fluent Bit logging
  - EKS/Fargate built-in Fluent Bit
- CloudWatch Container Insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices

인터넷 서비스를 위한 네트워크 트래픽 라우팅

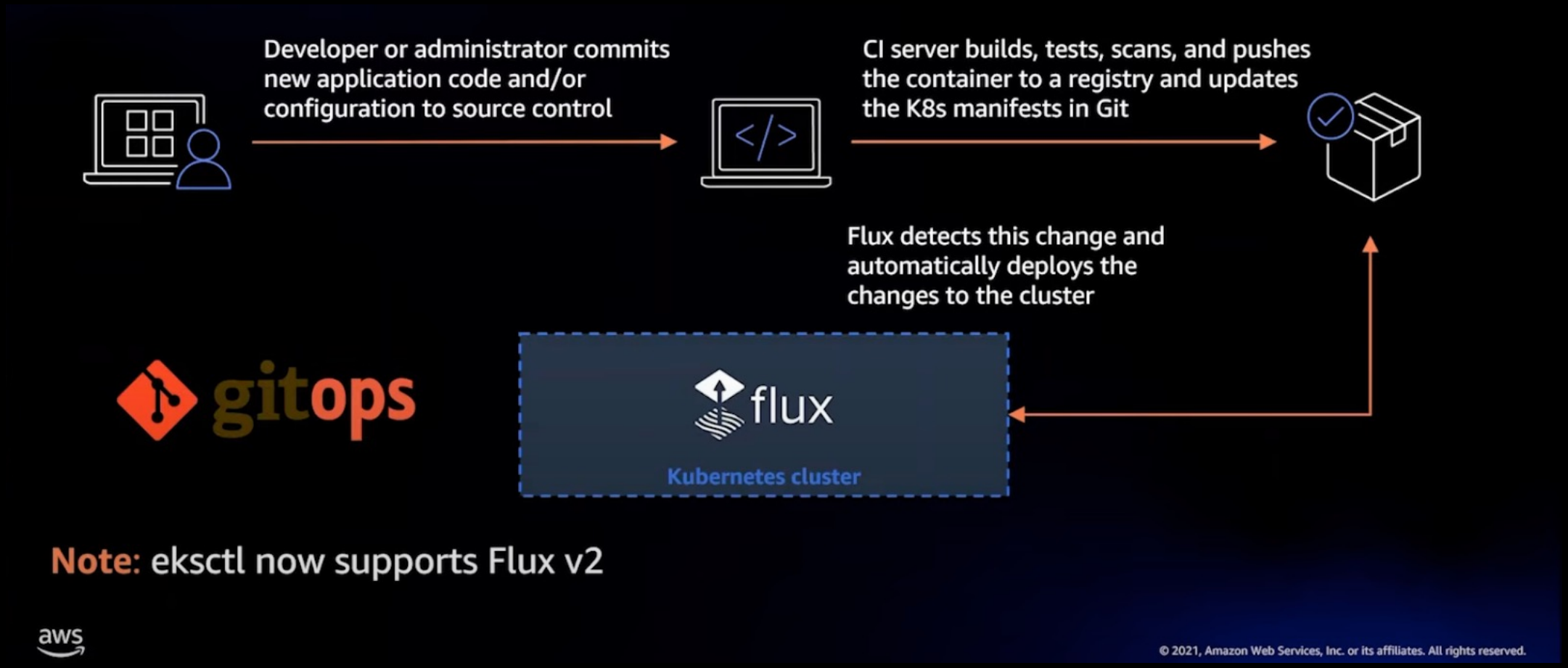
AWS Load Balancer Controller

- Provision AWS Network Load Balancers in response to Kubernetes services
- Provision AWS Application Load Balancers (ALBs) in response to Kubernetes ingress
- Skip kube-proxy and route directly to pods with IP targeting and VPC CNI
- Group multiple Ingresses under a single ALB

Note: The in-tree Kubernetes service controller is deprecated



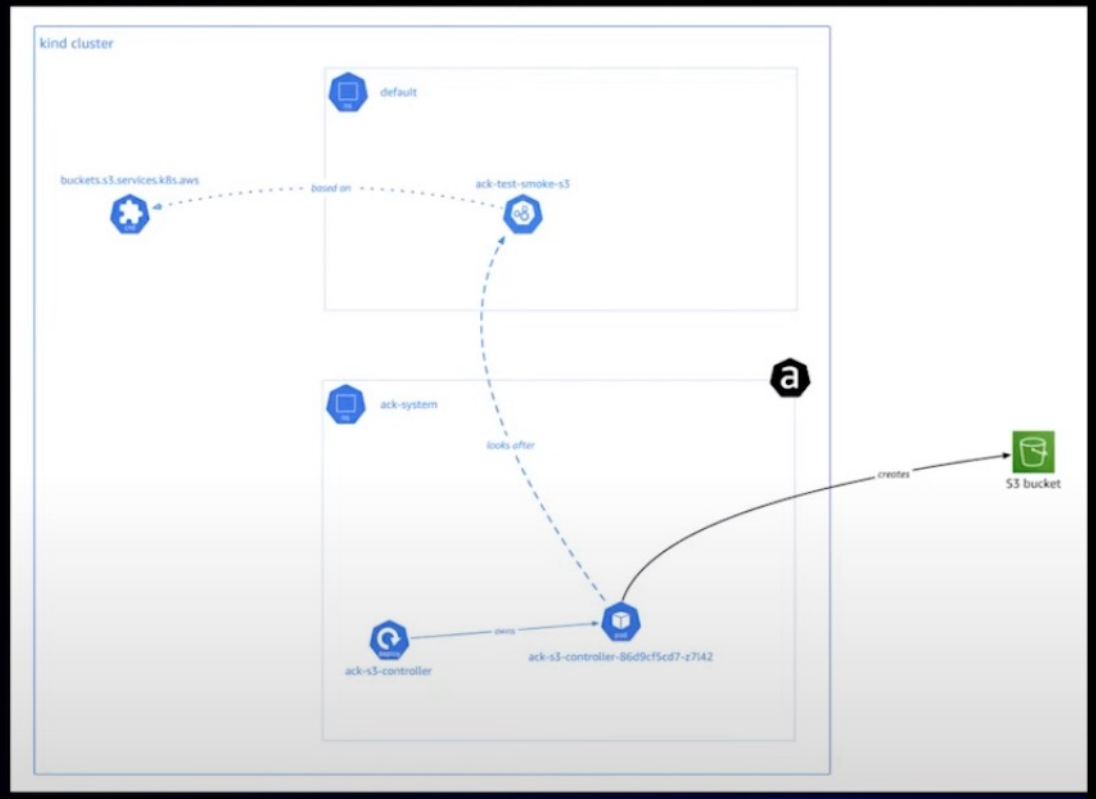
일관된 설정으로 많은 EKS 클러스터 관리



EKS 내부에서 AWS 서비스를 프로비저닝하고 액세스

AWS Controllers for Kubernetes (ACK)

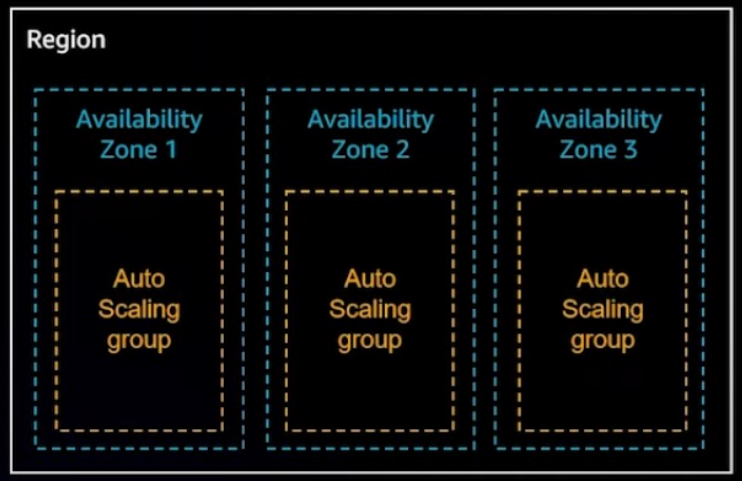
- Application developers create Kubernetes custom resources representing AWS services along their deployment manifests
- ACK service controller is then responsible for reconciling the desired AWS resources
- Controller code is automatically generated from AWS Go SDK, enabling fast support for new AWS service features



Stateful 워크로드 관리

- Amazon EBS CSI driver for high-performance use cases (ex. databases)
- Amazon EFS CSI driver for storage that needs to be shared across pods (ex. content management)
- Single-AZ Auto Scaling groups for workloads that require Amazon EBS volumes
- Dedicated node groups for stateful workloads – Disable Amazon EBS CSI tolerateAllTaints
- Lower max. Amazon EBS attachments per node, or use VPC CNI prefix assignment – to solve for ENI slot contention

Recommended Auto Scaling group setup for EBS stateful workloads

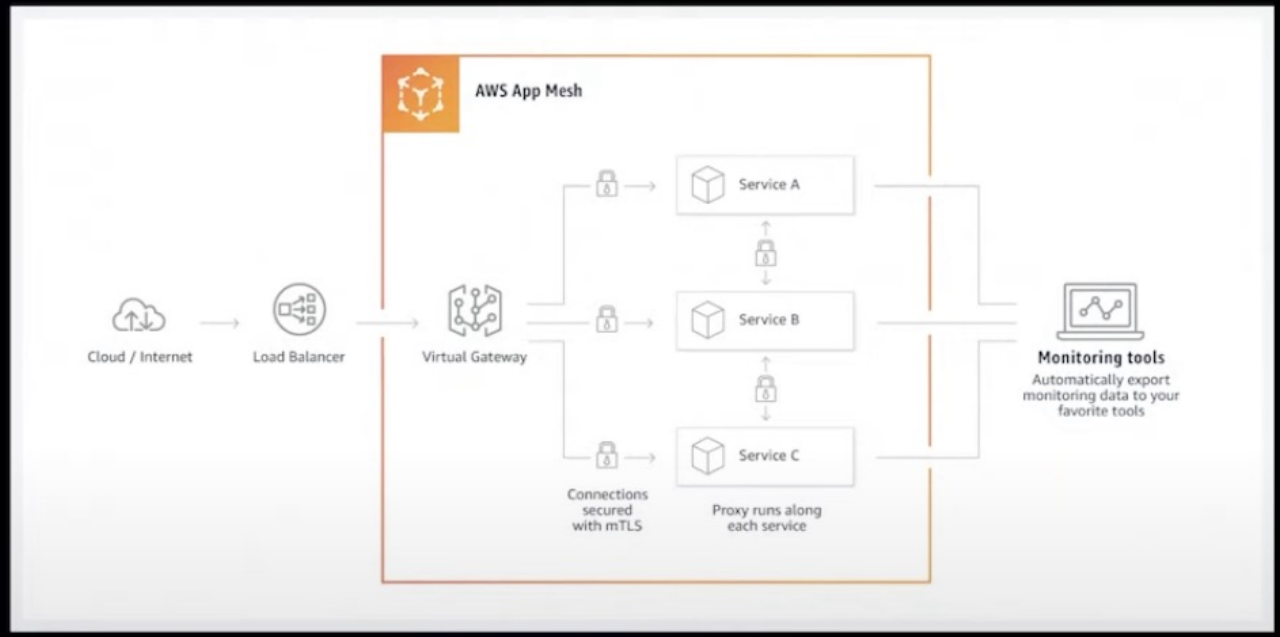


Recommendation: Use AWS managed services (through ACK) when possible

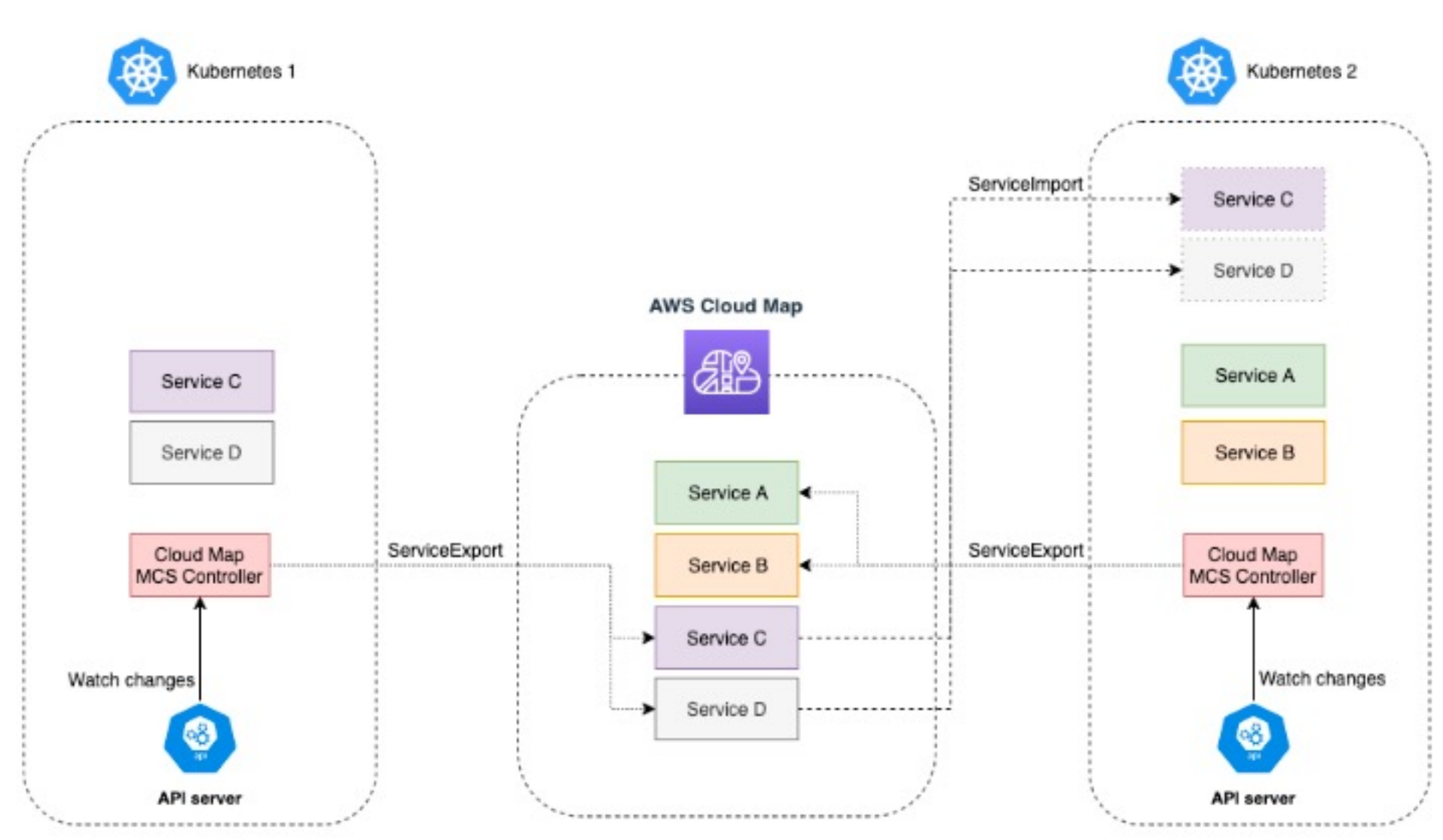
aws

대규모 클러스터 운용에서 서비스 식별

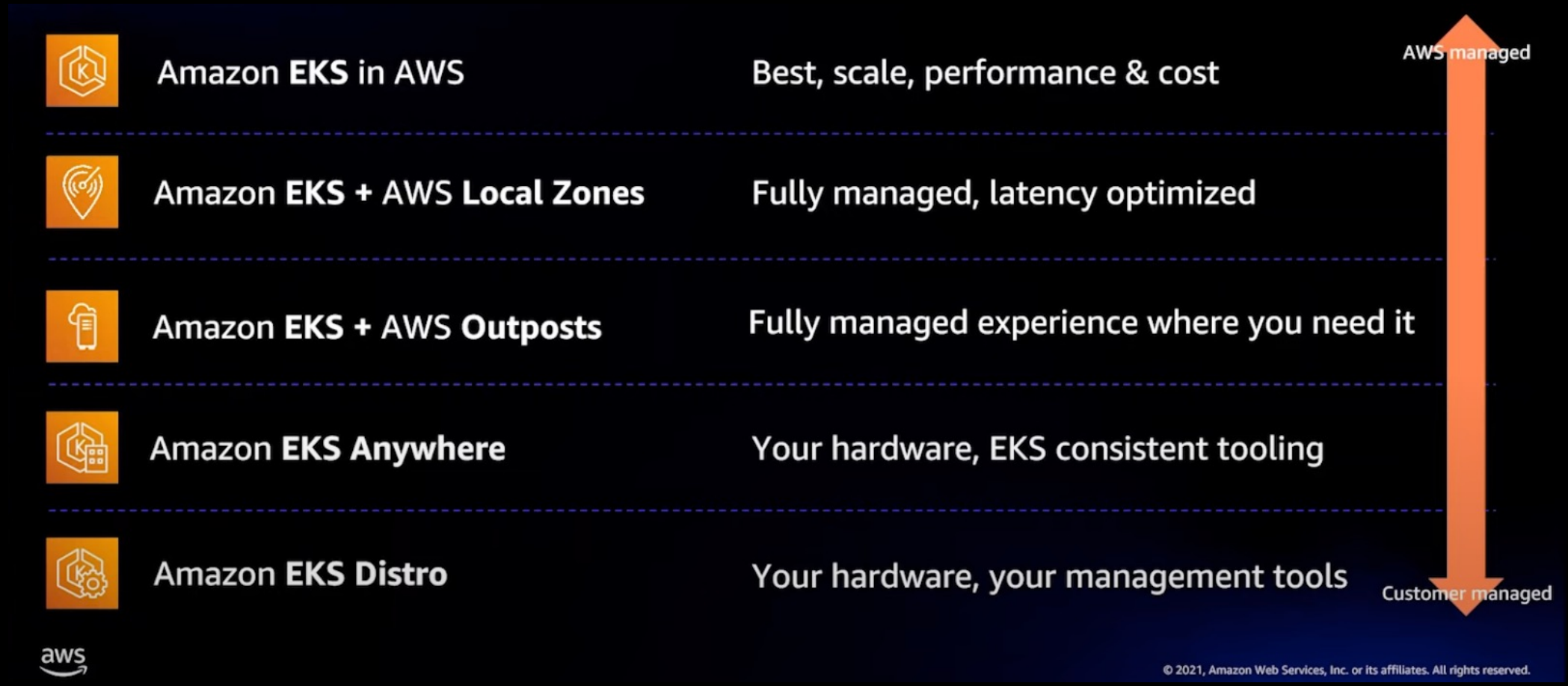
- AWS Cloud Map Multi-Cluster Services Controller
- AWS App Mesh Kubernetes controller for fully managed service mesh
  - Encryption in transit
  - Traffic shaping
  - Canary deployments
  - Service discovery
  - Service authentication
  - Observability
- Bring your own service mesh



대규모 클러스터 운용에서 서비스 식별



Portability - 일관된 방식으로 EKS 를 운용



Conclusion

Deep dive on Amazon EKS를 통해 EKS를 왜? 사용해야 하는지,

AWS가 EKS 서비스를 릴리즈하는데 어떤 철학을가지고 무엇에 주력하는지?

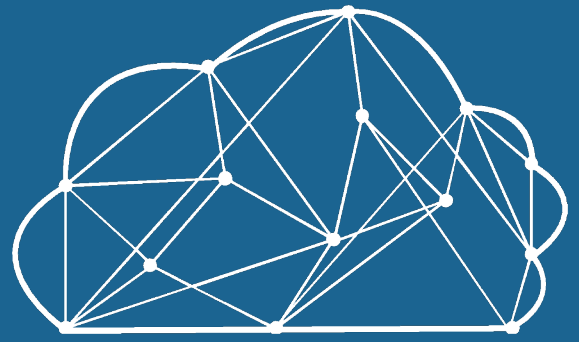
를 알 수 있었습니다.

사용자는 맨 처음 EKS와 같은 Cloud의 서비스를 생각할 때 비교적 높은 운영 비용과 기술 스택에

부담을 느끼지만 EKS가 빙산의 수면 아래에서 운영되는 Kubernetes 기술들의 복잡함과 통합을

위해 얼마나 큰 노력이 녹아 있음을 이해하면

왜 EKS를 사용하는지 Built for Production의 가치를 생각하게 됩니다.



감사합니다

BESPIN GLOBAL