MEGAZONECLOUD
EKS

MEGAZONE CLOUD

TYPE 1

TABLE OF CONTENTS

TYPE 2

AGENDA

01 Container Orchestration

Container Orchestration 필요성
Container Orchestration AWS EKS

02 EKS Add Open Source

EKS & AddOn Structure
CI/CD, Monitoring, Logging, Tracing, Backup, Integraded
Authentication, Service Mesh, Security

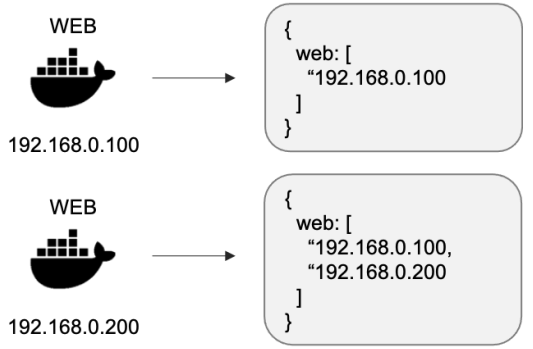
03 API Gateway

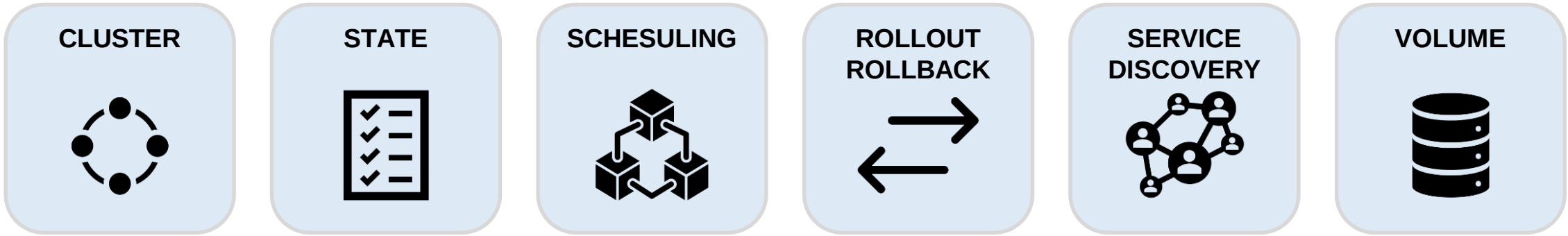
Ingress
API GateWay

1. Container Orchestration

가. Container Orchestration 필요성

- 클러스터(중앙제어-master-node) : 마스트를 두어 노드를 추상화 개념으로 관리한다.
- 상태 : 원하는 상태를 자동으로 관리한다. (replicase: 1 -> 2 -> 3)
- 스케쥴 : 자원의 상태를 확인하여 어디에 배포할 것인가?
- 배포 버전 관리 : 배포 버전을 손쉽게 관리할 수 있다.
- 서비스 등록 : 서비스 등록이 용이하다.
- 볼륨 : 다양한 스토리지 지원 (NFS, AWS EBS, GCE PD, etc)





MEGAZONE CLOUD

1. Container Orchestration

가. Container Orchestration 필요성

“쿠버네티스는 컨테이너화된 워크로드와 서비스를 관리하기 위한 이식성이 있고, 확장가능한 오픈소스 플랫폼이다. “

Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.

- **서비스 디스커버리와 로드 밸런싱:**
  쿠버네티스는 DNS 이름을 사용하거나 자체 IP 주소를 사용하여 컨테이너를 노출할 수 있다. 컨테이너에 대한 트래픽이 많으면, 쿠버네티스는 네트워크 트래픽을 로드밸런싱하고 배포하여 배포가 안정적으로 이루어질 수 있다.

- **스토리지 오케스트레이션:**
  쿠버네티스를 사용하면 로컬 저장소, 공용 클라우드 공급자 등과 같이 원하는 저장소 시스템을 자동으로 탑재할 수 있다

- **자동화된 롤아웃과 롤백:**
  쿠버네티스를 사용하여 배포된 컨테이너의 원하는 상태를 서술할 수 있으며 현재 상태를 원하는 상태로 설정한 속도에 따라 변경할 수 있다.

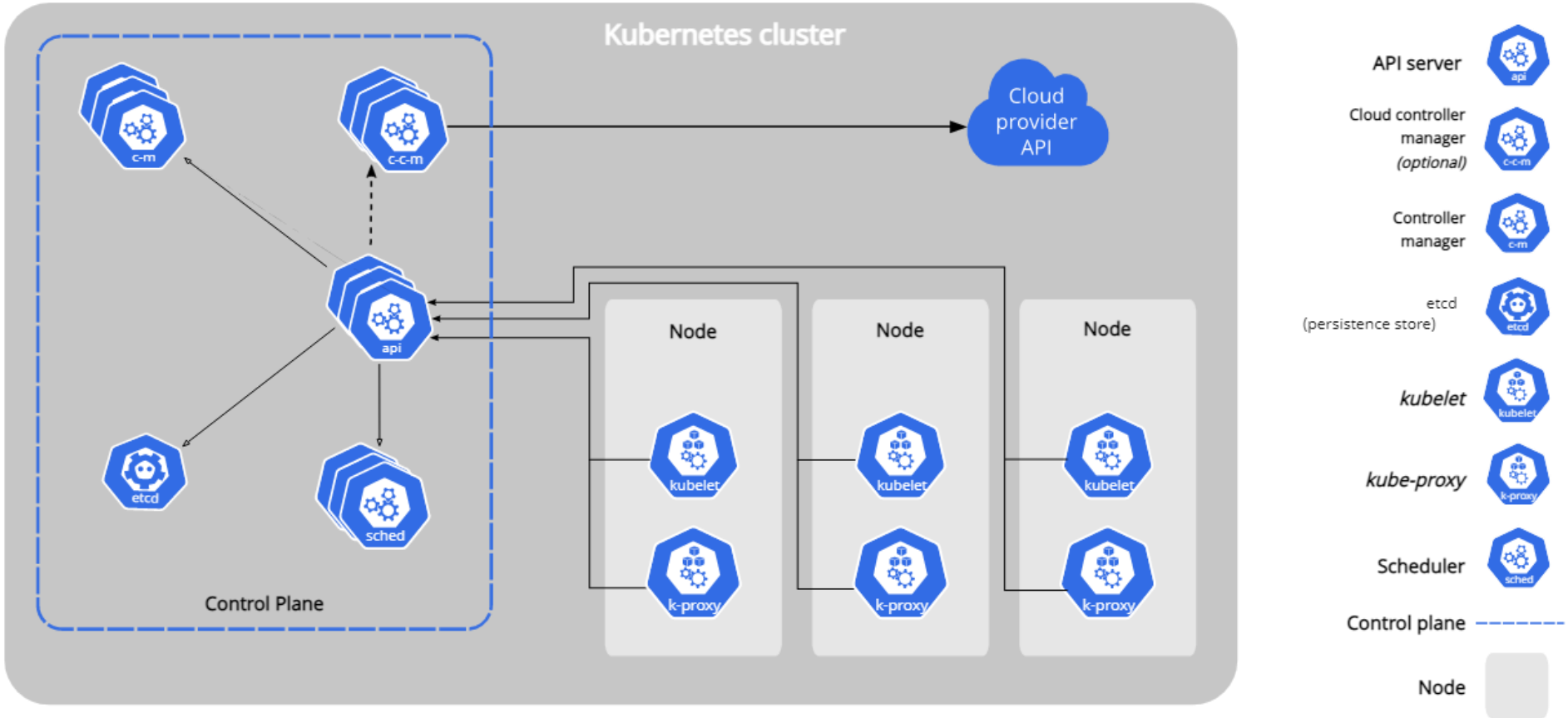
- **자동화된 빈 패킹:**
  컨테이너화된 작업을 실행하는데 사용할 수 있는 쿠버네티스 클러스터 노드를 제공한다. 각 컨테이너가 필요로 하는 CPU와 메모리(RAM)를 쿠버네티스에게 지시한다. 쿠버네티스는 컨테이너를 노드에 맞추어서 리소스를 가장 잘 사용할 수 있도록 해준다.

- **자동화된 복구:**
  쿠버네티스는 실패한 컨테이너를 다시 시작하고, 컨테이너를 교체하며, '사용자 정의 상태 검사'에 응답하지 않는 컨테이너를 죽이고, 서비스 준비가 끝날 때까지 그러한 과정을 클라이언트에 보여주지 않는다.

1. Container Orchestration

가. Container Orchestration 필요성

https://kubernetes.io/ko/docs/concepts/overview/components/



MEGAZONE CLOUD

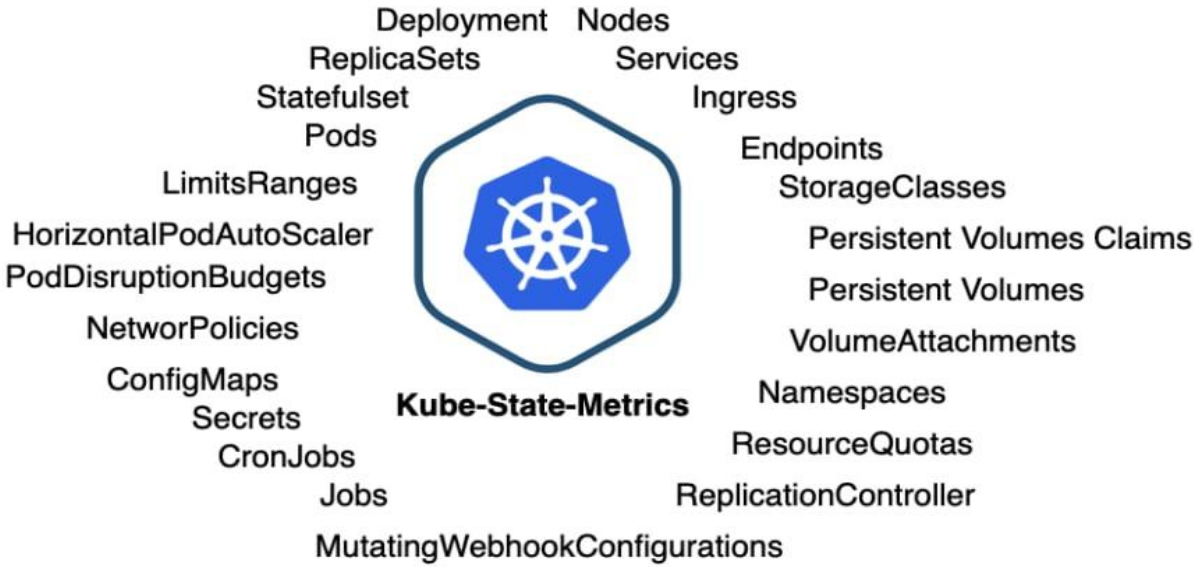
1. Container Orchestration

가. Container Orchestration 필요성

Kubernetes

“쿠버네티스는 컨테이너화된 워크로드와 서비스를 관리하기 위한 이식성이 있고, 확장가능한 오픈소스 플랫폼이다. “

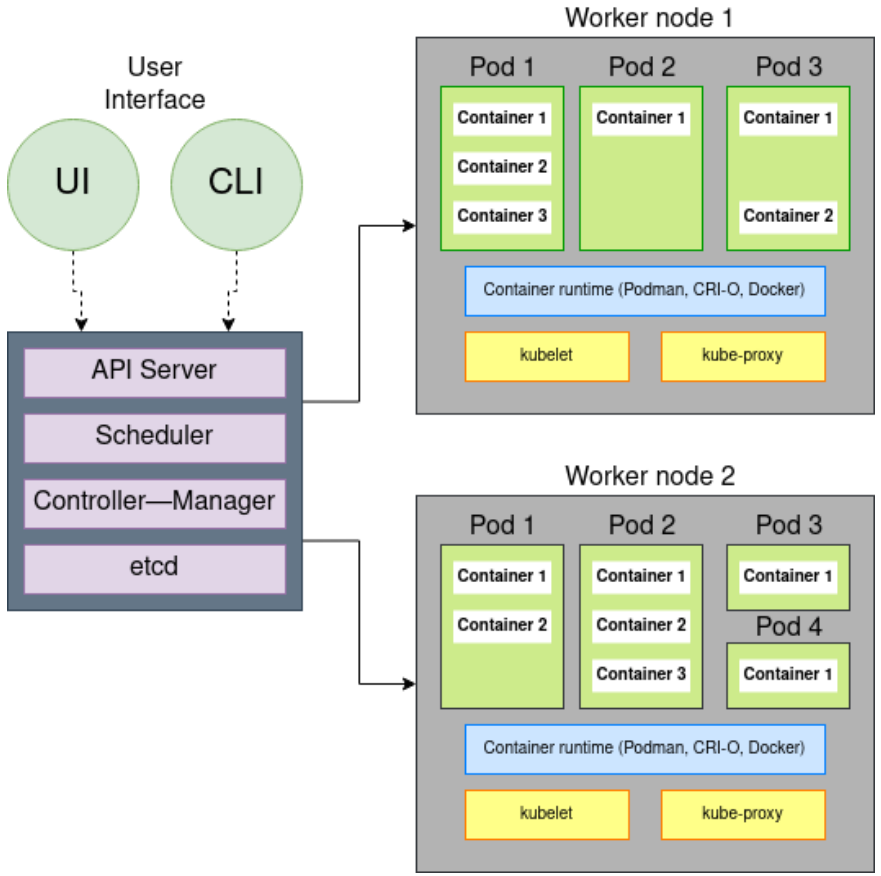
Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.



MEGAZONE CLOUD

1. Container Orchestration

가. Container Orchestration 필요성




```
NAMESPACE       NAME                                                        READY   STATUS    RESTARTS   AGE
argocd          argocd-application-controller-0                             1/1     Running   0          18h
argocd          argocd-applicationset-controller-6d7c9fb4b-qgqw8            1/1     Running   0          18h
argocd          argocd-dex-server-649d4669b8-6l5nk                          1/1     Running   0          18h
argocd          argocd-notifications-controller-765fd9c658-2cm5s            1/1     Running   0          18h
argocd          argocd-redis-74d458dcb8-mv6ch                               1/1     Running   0          18h
argocd          argocd-repo-server-7d9655659b-72vr8                         1/1     Running   0          18h
argocd          argocd-repo-server-7d9655659b-qzzkv                         1/1     Running   0          18h
argocd          argocd-server-68c57b8f97-n8vw6                              1/1     Running   0          18h
devops          apache-5d86cbbd84-vtzmh                                     1/1     Running   0          12h
devops          grafana-759c8b67bb-fmtpn                                    1/1     Running   0          13h
game-2048       deployment-2048-443-768b689799-jgm5v                        1/1     Running   0          23h
game-2048       deployment-2048-443-768b689799-r2f52                        1/1     Running   0          23h
game-2048       kubernetesdashboard-kubernetes-dashboard-6f68bc5587-xhft7   1/1     Running   1          23h
game-2048       tomcat-5475748cd-dttcm                                      1/1     Running   0          17h
ingress-nginx   ingress-nginx-controller-75f6959d4b-6tpvl                    1/1     Running   0          5d12h
ingress-nginx   ingress-nginx-controller-75f6959d4b-mctw8                    1/1     Running   0          5d12h
keycloak        keycloak-0                                                  1/1     Running   0          18h
keycloak        keycloak-postgresql-0                                       1/1     Running   0          18h
kube-system     aws-load-balancer-controller-8444b86c84-bnwpp               1/1     Running   0          3d10h
kube-system     aws-load-balancer-controller-8444b86c84-qcqgc               1/1     Running   0          5d12h
kube-system     aws-node-5m867                                              1/1     Running   0          3d10h
kube-system     aws-node-8b5rt                                              1/1     Running   0          5d12h
kube-system     aws-node-tpd5t                                              1/1     Running   0          5d12h
kube-system     aws-node-zj2lh                                              1/1     Running   0          5d12h
kube-system     cert-manager-8698d7f478-rrlqr                               1/1     Running   0          5d12h
kube-system     cert-manager-8698d7f478-vjmq9                               1/1     Running   0          3d10h
kube-system     cert-manager-cainjector-6bc9d758b-lpxw7                     1/1     Running   0          3d10h
kube-system     cert-manager-webhook-6d48469b7c-bdwmt                       1/1     Running   0          5d12h
kube-system     cluster-autoscaler-aws-cluster-autoscaler-54f84dfcc-lskk9   1/1     Running   0          5d12h
kube-system     cluster-autoscaler-aws-cluster-autoscaler-54f84dfcc-nhn79   1/1     Running   0          5d12h
kube-system     coredns-55b9c7d86b-n5dvl                                    1/1     Running   0          3d10h
kube-system     coredns-55b9c7d86b-sm9tl                                    1/1     Running   0          5d15h
kube-system     ebs-csi-controller-66d59488f5-ctttm                         6/6     Running   0          5d12h
kube-system     ebs-csi-controller-66d59488f5-gzvz6                         6/6     Running   0          5d12h
kube-system     ebs-csi-node-hc47l                                          3/3     Running   0          5d12h
kube-system     ebs-csi-node-rdd8l                                          3/3     Running   0          3d10h
kube-system     ebs-csi-node-tsc2z                                          3/3     Running   0          5d12h
kube-system     ebs-csi-node-x7hbp                                          3/3     Running   0          5d12h
kube-system     external-dns-55c5cf4d6d-2gfbc                               1/1     Running   0          5d12h
kube-system     kube-proxy-8nv9r                                            1/1     Running   0          5d12h
kube-system     kube-proxy-jd8kj                                            1/1     Running   0          5d12h
kube-system     kube-proxy-tf6dm                                            1/1     Running   0          3d10h
kube-system     kube-proxy-tm7pd                                            1/1     Running   0          5d12h
kube-system     metrics-server-694d47d564-4lcmj                             1/1     Running   0          5d12h
monitor         kube-prometheus-stack-kube-state-metrics-58ffc64954-2d8hh   1/1     Running   0          5d12h
monitor         kube-prometheus-stack-operator-86c8b895c9-hl5nb             1/1     Running   0          3d10h
monitor         kube-prometheus-stack-prometheus-node-exporter-28cmn        1/1     Running   0          5d12h
monitor         kube-prometheus-stack-prometheus-node-exporter-9bbqj        1/1     Running   0          5d12h
monitor         kube-prometheus-stack-prometheus-node-exporter-r4zvt        1/1     Running   0          5d12h
monitor         kube-prometheus-stack-prometheus-node-exporter-xsrv2        1/1     Running   0          3d10h
monitor         prometheus-kube-prometheus-stack-prometheus-0               2/2     Running   0          5d12h
service         nginx-9f854d6f5-s47hb                                       1/1     Running   0          17h
[root@su ~]#
```


MEGAZONE CLOUD

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.

- **보안 네트워킹 및 인증:**
  Amazon EKS는 Kubernetes 워크로드를 AWS 네트워킹 및 보안 서비스와 통합합니다. 또한 AWS Identity and Access Management(IAM)와의 통합으로 Kubernetes 클러스터에 대한 인증을 제공
- **간편한 클러스터 규모 조정:**
  Amazon EKS를 사용하면 워크로드 수요에 따라 Kubernetes 클러스터 규모를 쉽게 조정할 수 있습니다. Amazon EKS는 CPU 또는 사용자 지정 지표를 기반으로 수평 Pod 자동 규모 조정, 그리고 전체 워크로드 수요를 기반으로 클러스터 자동 규모 조정을 지원합니다.
- **높은 가용성:**
  Amazon EKS는 여러 가용 영역의 컨트롤 플레인에 대한 고가용성을 제공합니다.
- **AWS 서비스와 통합:**
  Amazon EKS는 다른 AWS 서비스와 통합되어, 컨테이너화된 애플리케이션을 배포하고 관리하기 위한 포괄적인 플랫폼을 제공합니다.

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

Amazon EKS의 Kubernetes 버전 연장 지원 :

Amazon EKS의 Kubernetes 버전 지원이 연장됨에 따라 이제 Kubernetes 버전의 정식 출시 시점부터 최대 26개월 동안 Amazon EKS의 Kubernetes 버전에서 Amazon EKS 클러스터를 실행할 수 있습니다.

연장 지원이 적용되는 Kubernetes 버전에서 실행되는 각 Amazon EKS 클러스터에 시간당 0.60 USD의 요금이 부과됩니다.
클러스터별 시간당 0.60 USD라는 요금에는 Amazon EKS 클러스터에 대해 부과되는 클러스터별 **시간당 0.10 USD**의 요금이 포함된 것입니다

Amazon EKS의 생성 :

- AWS Web Console
- AWS eksctl , AWS CLI 로 제공되는 CLI Command
- AWS CloudFormation, CDK
- Terraform (CDKTF), Boto3 (AWS CLI SDK) , Pulumi 등의 Program Language 혼합
- CrossPlane, Yaml , Helm 배포
- 현재 EKS 1.29 (Kubernetes 1.30)

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

관리형 노드 그룹:

Amazon EKS 관리형 노드 그룹을 사용하면 Kubernetes 애플리케이션을 실행하기 위해 컴퓨팅 용량을 제공하는 Amazon EC2 인스턴스를 별도로 프로비저닝하거나 등록할 필요가 없습니다.

Amazon EC2 Auto Scaling 그룹의 일부로 프로비저닝됩니다. 인스턴스 및 Auto Scaling 그룹을 포함한 모든 리소스는 AWS 계정 내에서 실행됩니다. 각 노드 그룹은 정의한 여러 가용 영역에서 실행됩니다.

Kubernetes Cluster Autoscaler에서 자동 검색할 수 있도록 태그가 자동으로 지정됩니다. 노드 그룹을 사용하여 Kubernetes 레이블을 노드에 적용하고 언제든지 업데이트할 수 있습니다.

Amazon EKS 관리형 노드 그룹을 사용하기 위한 추가 비용은 없으며 프로비저닝한 AWS 리소스에 대해서만 비용을 지불합니다. 여기에는 Amazon EC2 인스턴스, Amazon EBS 볼륨, Amazon EKS 클러스터 시간 및 기타 AWS 인프라가 포함됩니다. 최소 요금 및 선수금은 없습니다.

Amazon EKS는 관리형 노드 그룹 인스턴스에 Kubernetes 레이블을 추가합니다. 이러한 Amazon EKS 제공 레이블에는 eks.amazonaws.com 접두사가 붙습니다.

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

자체 관리형 노드 그룹:

Amazon EKS 클러스터에 자체 관리형 노드를 추가하려면 다음 항목을 참조하세요. 자체 관리형 노드를 수동으로 시작하는 경우 각 노드에 다음 태그를 추가합니다.

키	값
kubernetes.io/cluster/*my-cluster*	owned

```
eksctl create nodegroup          --cluster my-cluster \
  --name al-nodes                --node-type t3.medium \
  --nodes 3                      --nodes-min 1 \
  --nodes-max 4                  -- ssh-access \
  --managed=false                --ssh-public-key my-key
```

자체 관리형 Amazon Linux 노드 시작하기
자체 관리형 Bottlerocket 노드 시작(AWS 제공하는 가벼운 OS , ContainerD 를 포함한 작은 VM , ECS, EKS Fragate, vm-ware, metal)
자체 관리형 Windows 노드 시작
Ubuntu Linux AMI (Canonical은 Amazon EKS와 협력하여 클러스터에서 사용할 수 있는 노드 AMI를 만들었습니다.)

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

VPC CNI Driver:

AWS VPC CNI Driver : CNI 추가 기능은 탄력적 네트워크 인터페이스를 생성하여 Amazon EC2 노드에 연결합니다. 또한 이 추가 기능은 프라이빗 IPv4 또는 IPv6 주소를 VPC에서 각 Pod 및 서비스에 할당합니다.

Calico : EBPF 지원
Cilium (cilium.io) : EBPF 기반의 Overlay Network 을 구성하여 Pod간 통신 역할을 합니다.
EBPF 기반은 Linux 에서 차용한 기술로 커널에서 나온 메시지로 networking, observability, tracing , security 등을 제어

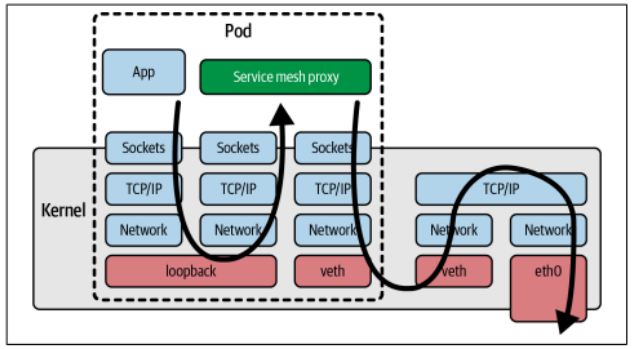
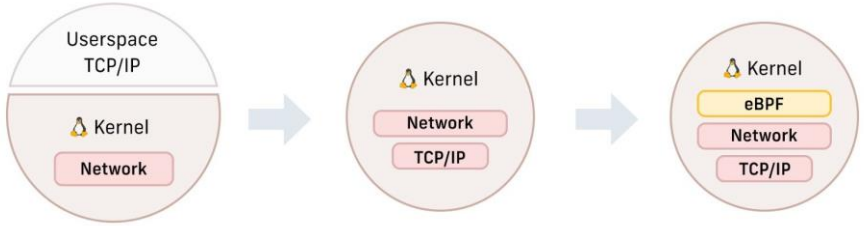


Figure 1-5. Path of a network packet using a service mesh proxy sidecar container



MEGAZONE CLOUD

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

CSI Driver:

AWS EBS CSI Driver : Amazon EBS 볼륨의 수명 주기를 사용자가 생성하는 Kubernetes 볼륨의 스토리지로 관리
~~EC2 (Volume EBS)~~ , **PV (K8S Volume EBS)**
Static Provisioning , Dynamic Provisioning, NVME Volume, Block Volume, Volume SnapShopts, Resizing, Volume Modification(iops, throughput)

Amazon EFS CSI 드라이버
Amazon FSx for Lustre CSI 드라이버, NetApp ONTAP CSI 드라이버, OpenZFS CSI 드라이버
Amazon File Cache CSI 드라이버
Mountpoint for Amazon S3 CSI 드라이버
CSI 스냅샷 컨트롤러

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

```
$ aws ec2 describe-instance-types --filters "Name=instance-type,Values=m5.large" --query "Inst
-------------------------------------------------
|            DescribeInstanceTypes              |
+----------+------------+---------+------------+
| IPv4addr | IPv6addr   | MaxENI  |   Type     |   # of ENI * (# of IPv4 per ENI - 1) + 2
+----------+------------+---------+------------+
|  10      |  10        |  3      |  m5.large  |   3 * (10 - 1) + 2 = 29
+----------+------------+---------+------------+
```

고려사항:

1. Network 설계
   - Network 설계시 AWS EKS 는 CIDR 가용영역이 많으면 많을 수록 좋음.
   - EKS Upgrade 시 Rolling Upgrade, Blue/Green Upgrade 가 발생할 수 있어 여분의 IP 가 필요.
   - 향후 확장시 IP 의 개수가 더 필요
   - K8S 관련 Driver, 보안, 모니터링, 기타 에 많은 ip 들이 필요.
   - Public Subnet (LB 확장 할 가용 IP 정도및 조금의 여분), Private Subnet (EC2, Container) 사용할 여분의 IP

   - EC2 제한의 IP 개수, Pod 당 ENI IP를 가지고 있음.
     “ kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|PrivateIPv4Address “
   - Custom Networking VPC : AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=$subnet_id_1","$subnet_id_2
```

   - EKS 모범 사례 가이드 : ENIConfig Subnet 1개 Zone만 관할

MEGAZONE CLOUD

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “

고려사항:

2. EKS Upgrade
   - Cluter Upgrade , NodeGroup Upgrade, AddOn Upgrade 부분으로 구분
   - Dev -> Stg -> Prd 순으로 테스트 후 단계별 적용

가. EKS Cluster Upgrade
   a. Kubernetes.io : 버전간 차이점, 새로운 환경, 기능에 대한 검토, 현재 적용이 되어 있거나 Deprecate 사항 확인
   b. EKS Cluster Upgrade : 1단계씩 만 업그레이드 (서비스의 영향도가 크지 않다.
      문제가 될수 있는 수준이라면 api version DeployMent, , Configmap, Secret , Role V1 에 따른 api 버전)

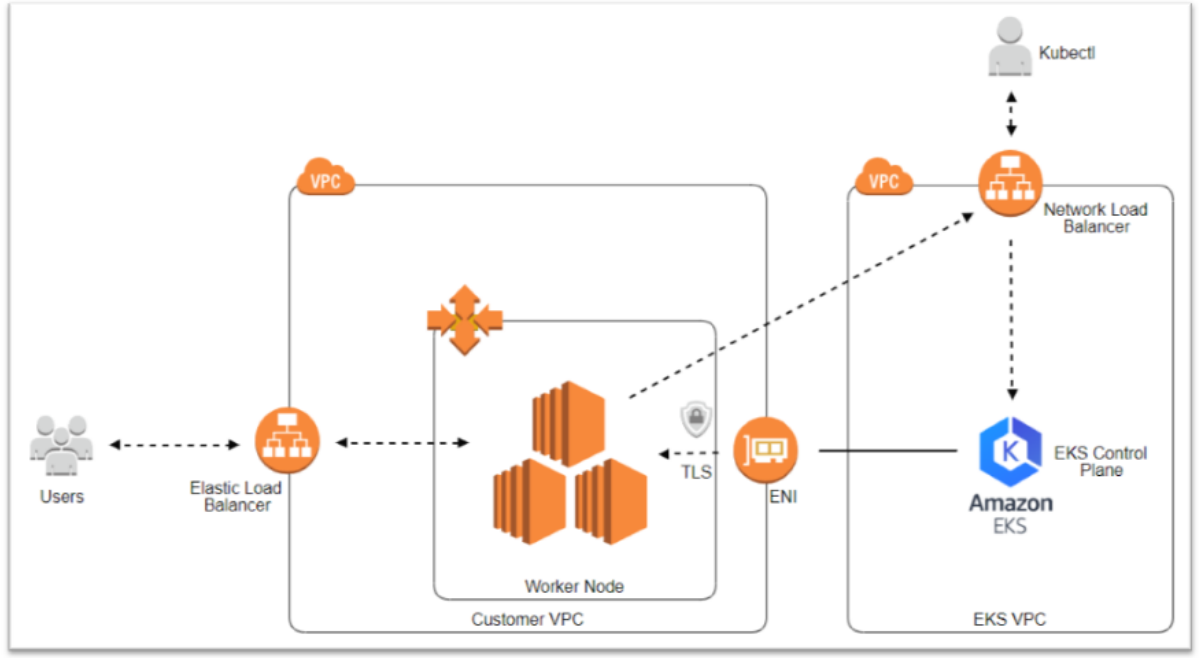
나. NodeGroup Upgrade
   a. Rolling Upgrade (WorkerNode Drain 방식)
   b. Blue / Green Upgrade (WorkerNode Cordon + Pod Rollout 방식)

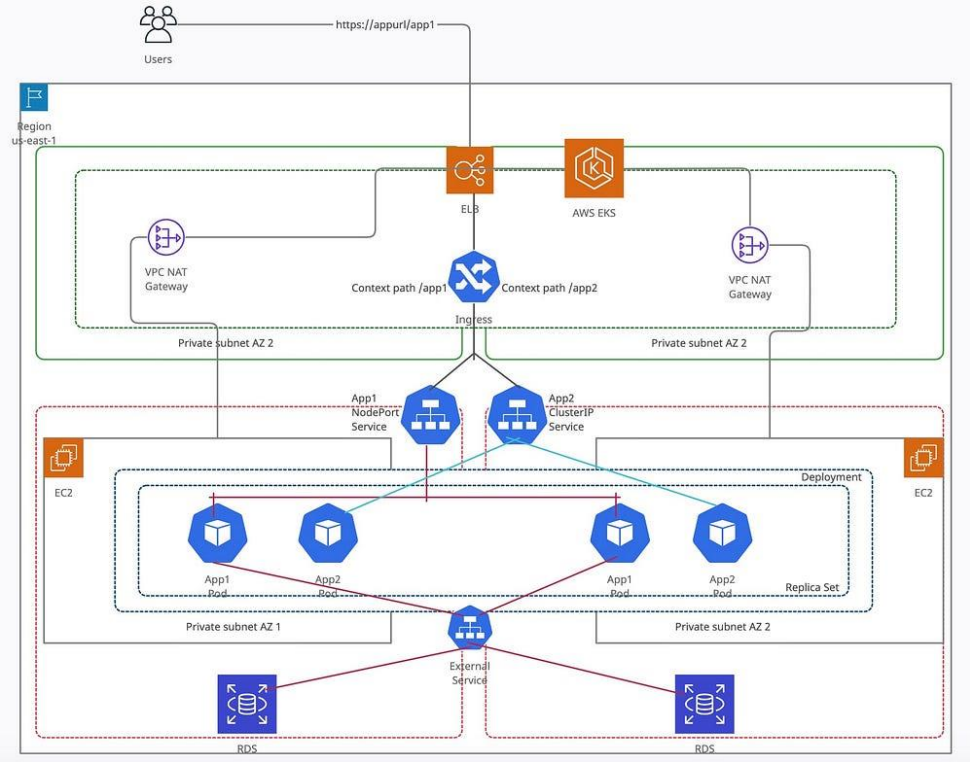
1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

“EKS는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. “





MEGAZONE CLOUD

1. Container Orchestration

나. Container Orchestration AWS EKS

EKS

Elastic Kubernetes Service 와 General Kubernetes

	EKS	Kubernetes
장점	• 관리형 서비스 (ControlPlane 및 노드관리) • AWS EKS는 다른 서비스들 과도 연동이 쉬움. • EBS, EFS, S3, RDS , ALB 등에 접근이 용이 • AWS IAM 과 상호 보완되어 보안에 강화됨 • API Server와 ETCD 관리 운영에 부담이 없다. • EC2 Image, Kubelet, ContainerD 관리 용이 • 컨테이너에 인입되는 트래픽에 제한이 없다.	• 범용성과 유연성, 맞춤화, 다양한 오픈소스 사용가능 • 원하는 방식과 클러스터를 합리적으로 관리 • 숙련된 팀에서는 관리가 용이, 비용절감 (무료)
단점	• AWS 에 종속적이며, 다른 Open Source 사용이 어렵지 않지만, AWS 서비스를 대체하면 지원이 어렵다. • 3 개 버전에 대한 비용이 절감되지만 그이전 버전은 비용이 비싸다.	• 복잡하며, API Server, ETCD 등의 키를 관리 • Kubernetes 학습곡선이 더 필요하다. • 리소스에 대한 수평적 , 수직적 확장이 어렵다. • Open Sourece에 종속적이고, 문제 발생시 관리및 대체가 어려움. • 전문인원이 필요하며, K8S 생태계를 주시 • 운영자의 주관성이 많이 부여 • 기존 VM등과 연계되어 재시작등이 어려움
고려 사항	• 서비스의 크기가 중요하여 소규모 서비스로는 불 합리 함. (리소스 낭비) • 확장성, 대용량의 서비스가 많고 관리해야 하는 서비스가 많을 시 도입 고려 사항 • 많은 인프라를 효율적으로 관리가 가능하며, 적은 인원으로 수많은 서비스를 일괄 적용이 가능	

TYPE 2

AGENDA

01 Container Orchestration

Container Orchestration 필요성
Container Orchestration AWS EKS

02 EKS Add Open Source

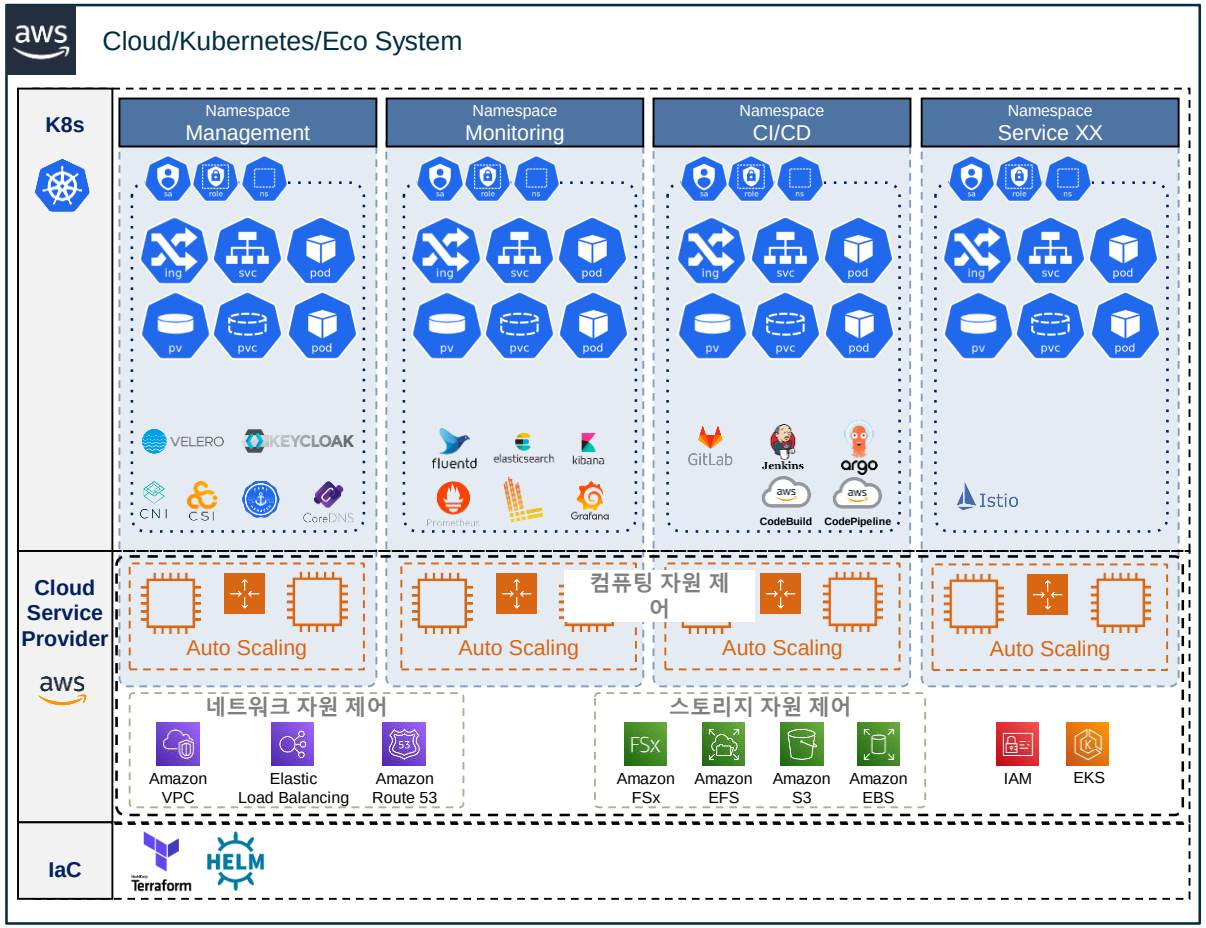
EKS & AddOn Structure
CI/CD, Monitoring, Logging, Tracing, Backup, Integraded
Authentication, Service Mesh, Security

03 API Gateway

Ingress
API GateWay

2. EKS Add Open Source

EKS & AddOn Structure



클라우드 서비스 제공자 기반 K8s 연동

▣ 네트워크, 스토리지, Cluster Auto scaler

- K8s 기반의 솔루션으로 클라우드 서비스 제공자 별 API와 연동하여 네트워크, 스토리지 및 컴퓨팅 자원의 생성 및 K8s 할당 등의 관리 기능을 제공

모니터링

▣ K8s기반 메트릭 모니터링 에코 시스템

- 클라우드 서비스, K8s 및 컨테이너의 메트릭 수집 및 모니터링 기능 제공

▣ K8s기반 로그 모니터링 에코 시스템

- 클라우드 서비스, K8s 및 컨테이너의 로그 수집 및 모니터링 기능 제공

CI/CD

▣ K8s 기반 CI/CD 에코 시스템

- 어플리케이션의 통합, 테스트 및 배포에 이르는 라이프 사이클 전체에 걸친 지속적인 자동화와 모니터링 기능을 제공

보안 및 거버넌스

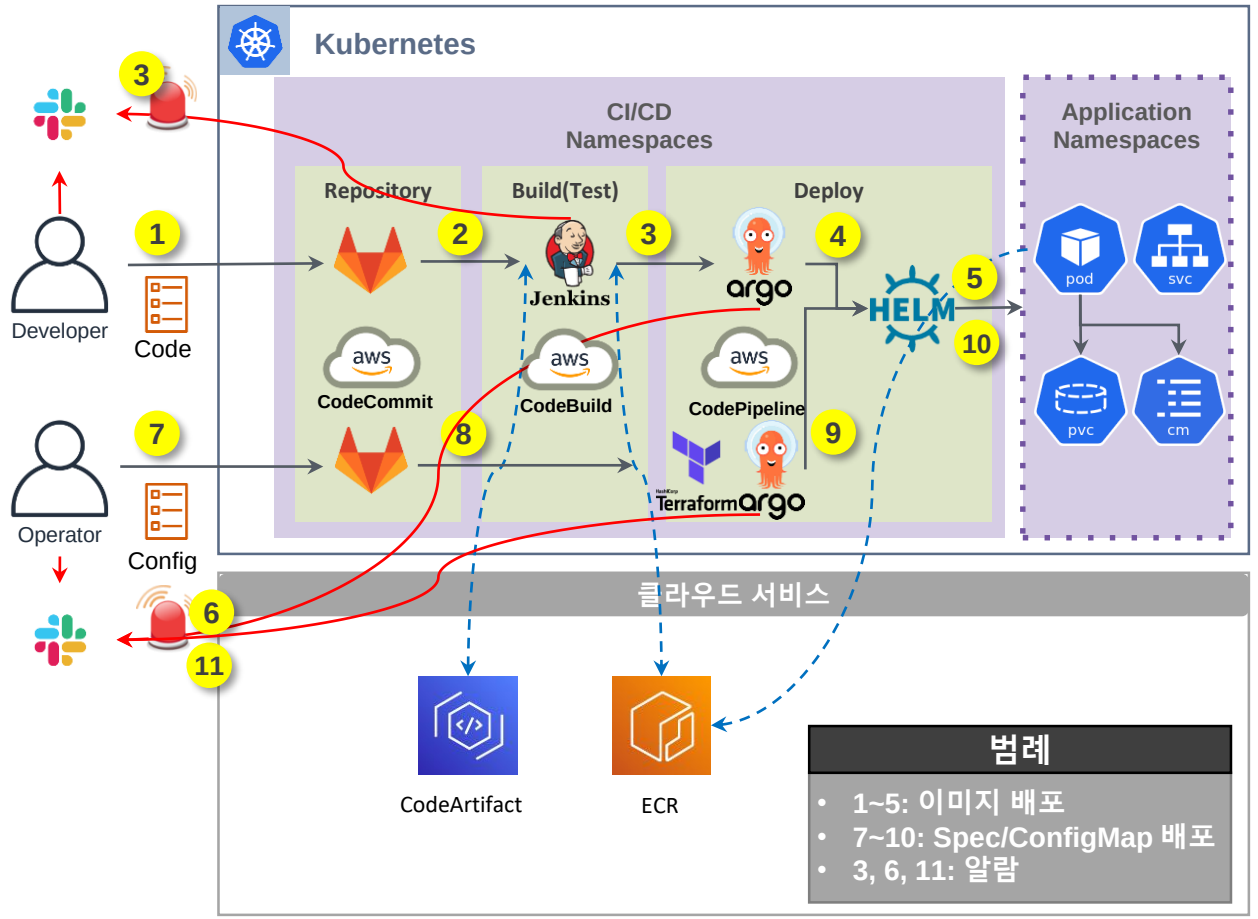
▣ K8s 기반 보안 및 거버넌스 에코 시스템

- 네트워크 정책 관리, 컨테이너 간 커뮤니케이션 관리, 컨테이너 보안 관리 등 컨테이너 플랫폼 전반에 걸친 보안 및 거버넌스 서비스를 제공

MEGAZONE CLOUD

2. EKS Add Open Source

CI/CD



구성 원칙

- 애플리케이션의 코드, 설정 및 인프라 배포를 고려한 배포 파이프라인을 구성
- 배포 장애 발생 시, 빠른 롤백 지원
- 배포 상황에 대한 모니터링 및 알람 지원
- CI/CD 시스템 구성은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

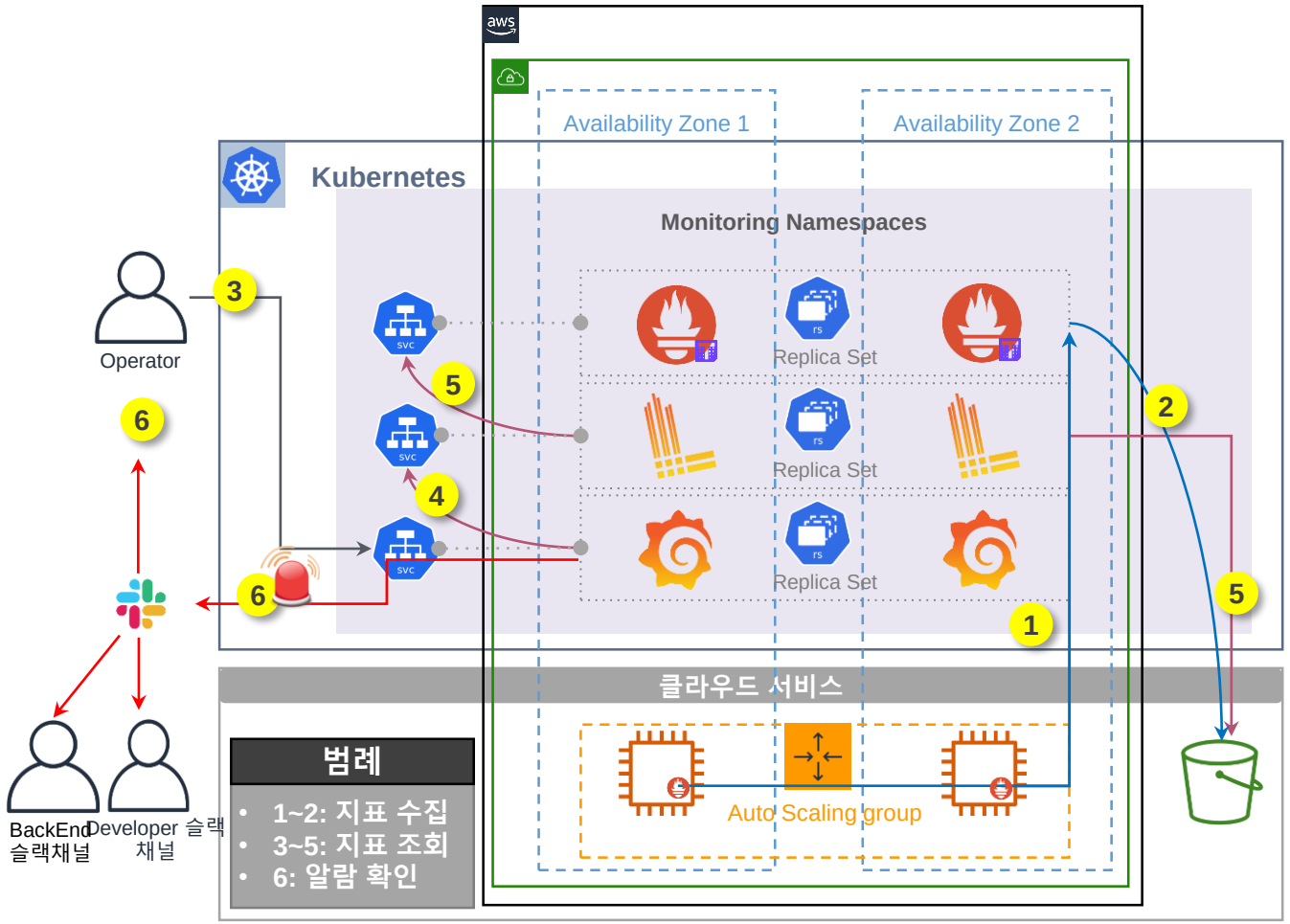
적용 기준

- **적용 대상 : 코드 저장소, 코드 빌드, 테스트, 컨테이너 배포, 이미지 저장소, 라이브러리 저장소, 알람**
- **적용기준**
  - ✔ 애플리케이션의 코드 및 설정 파일에 대해 Branch, Pull Request Merge 및 버전 관리 적용
  - ✔ 배포 코드에 대한 정적 분석, 테스트, 이미지 취약점 스캔 지원
  - ✔ 다양한 배포 방식 (Blue-Green, Canary 등) 지원
  - ✔ 운영 편의성을 고려한 배포 패턴 별 배포 파이프라인 템플릿 지원

MEGAZONE CLOUD

2. EKS Add Open Source

Monitoring



구성 원칙

- 서버, 플랫폼, 애플리케이션에 대한 실시간 지표 수집, 저장 및 시각화 적용
- 핵심 컴포넌트에 대해 고가용성 적용
- 지표의 장기 보관을 위해 클라우드 제공자의 비용 효율적인 스토리지와의 통합 적용
- 모니터링 시스템 구성은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

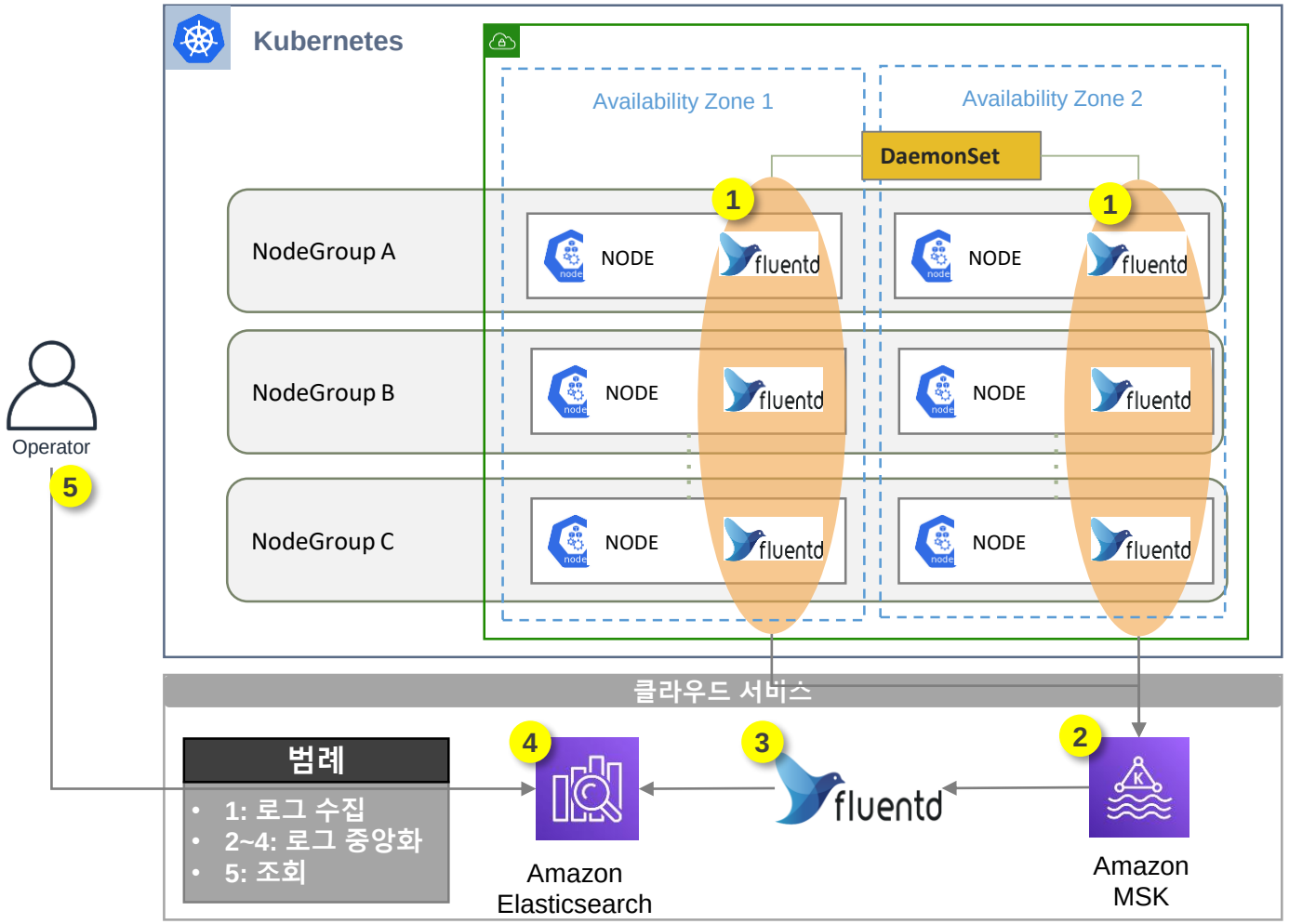
적용 기준

- **적용 대상 : 인프라, 플랫폼, 애플리케이션 지표 수집기, Prometheus, Grafana, Loki (로그)**
- **적용기준**
  - ✔ 에이전트를 통해 서버, 플랫폼 및 컨테이너 별 지표를 수집하여 프로메테우스에 저장
  - ✔ 모니터링 시스템의 고가용성을 위해 주요 컴포넌트에 대해 AZ 별 분산 이중화 구성
  - ✔ 트렌드 분석을 위한 장기 보관용 지표는 클라우드 제공자의 스토리지 서비스로 전송
  - ✔ 운영에 필요한 대시보드 및 알람 구성

MEGAZONE CLOUD

2. EKS Add Open Source

Logging



구성 원칙

- 서버, K8S 컴포넌트, 애플리케이션 로그에 대한 데이터 파이프라인 제공
- 데이터 파이프라인의 핵심 컴포넌트는 Amazon 관리형 서비스를 적용하여 로그 손실 최소화 및 고가용성 제공
- 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

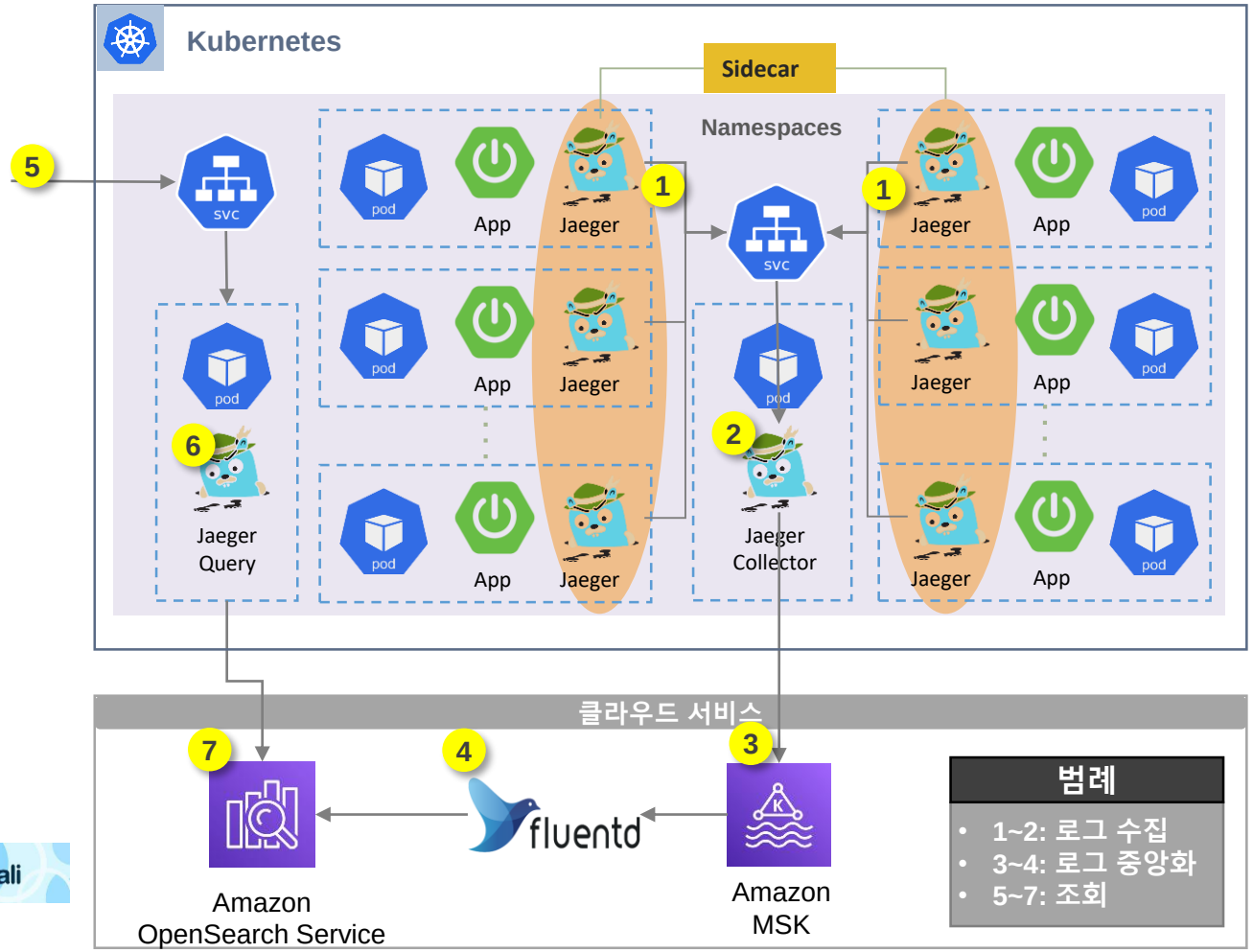
적용 기준

- **적용 대상 : Fluentd, Amazon Kafka, Amazon OpenSearch**
- **적용기준**
  - ✔ 안정적인 운영을 위해서 관리형 서비스 사용을 권장 하나 필요에 따라 고객 관리형 OSS(Apache Kafka, Elasticsearch, Kibana) 또는 상용 솔루션(DataDog 등)으로 대체 가능함
  - ✔ 성능 요건(시스템 규모) 및 유지 비용을 고려하여 일부 기능(Amazon Kafka 등) 제외 가능함

MEGAZONE CLOUD

2. EKS Add Open Source

Tracing



구성 원칙

- 분산 서비스 트랜잭션 로그에 대한 데이터 파이프 라인 제공
- 데이터 파이프라인의 핵심 컴포넌트는 Amazon 관리형 서비스를 적용하여 로그 손실 최소화 및 고가용성 제공
- 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

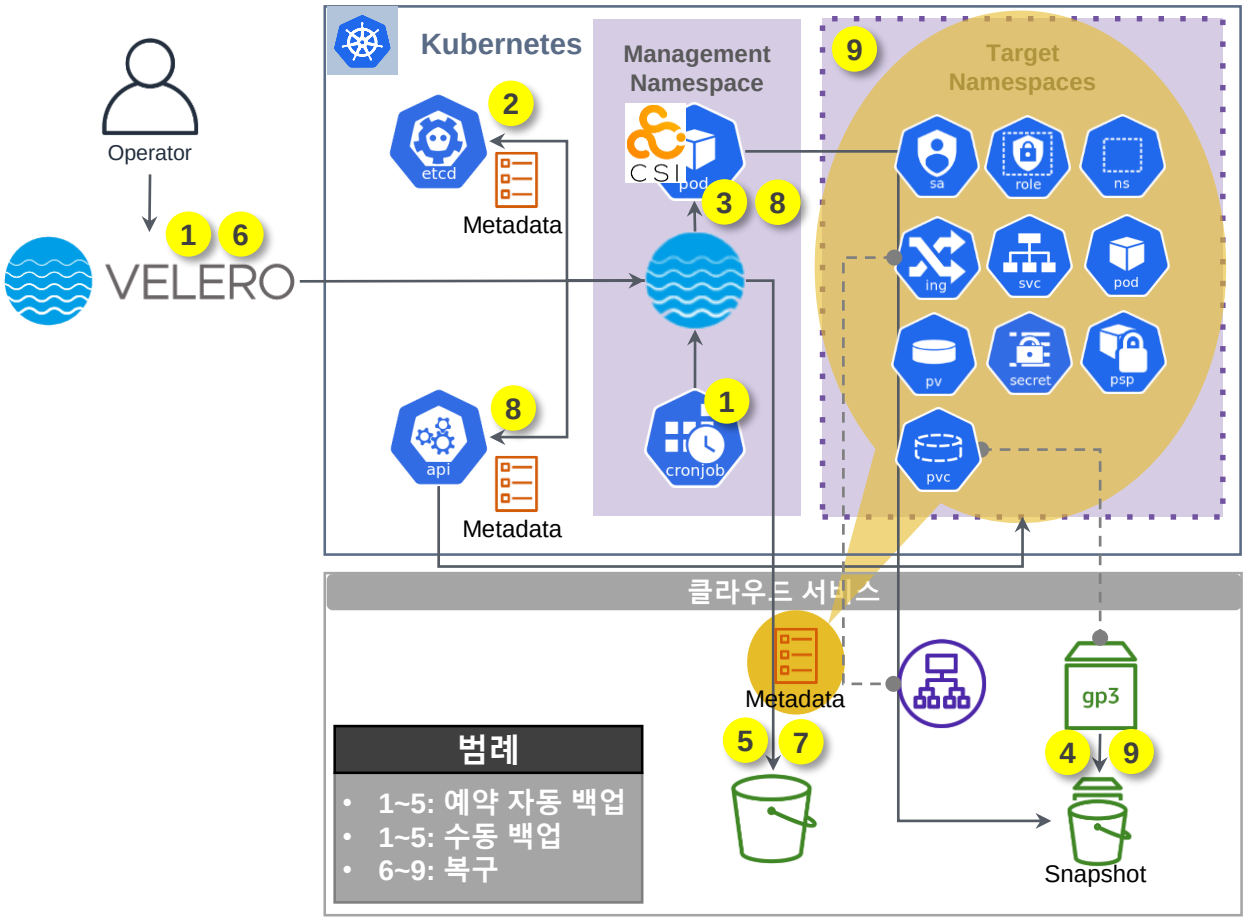
적용 기준

- **적용 대상 : Jaeger(Sidecar), Jaeger Collector, Jaeger Query, Amazon Kafka, Amazon OpenSearch, Fluentd**
- **적용기준**
  - ✔ 안정적인 운영을 위해서 관리형 서비스 사용을 권장하나 필요에 따라 고객 관리형 OSS(Apache Kafka, Elasticsearch, Kibana) 또는 상용 솔루션(DataDog 등)으로 대체 가능함
  - ✔ 성능 요건(시스템 규모) 및 유지 비용을 고려하여 일부 기능(Amazon Kafka 등) 제외 가능함

MEGAZONE CLOUD

2. EKS Add Open Source

Backup & Restore



구성 원칙

- 운영 환경의 모든 애플리케이션 서비스에 대해 네임 스페이스 별 예약 백업을 적용
- 메타 데이터 저장소는 가용성 및 비용 효율적인 클라우드 제공자의 스토리지 서비스를 적용
- 백업 시스템 구성은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원

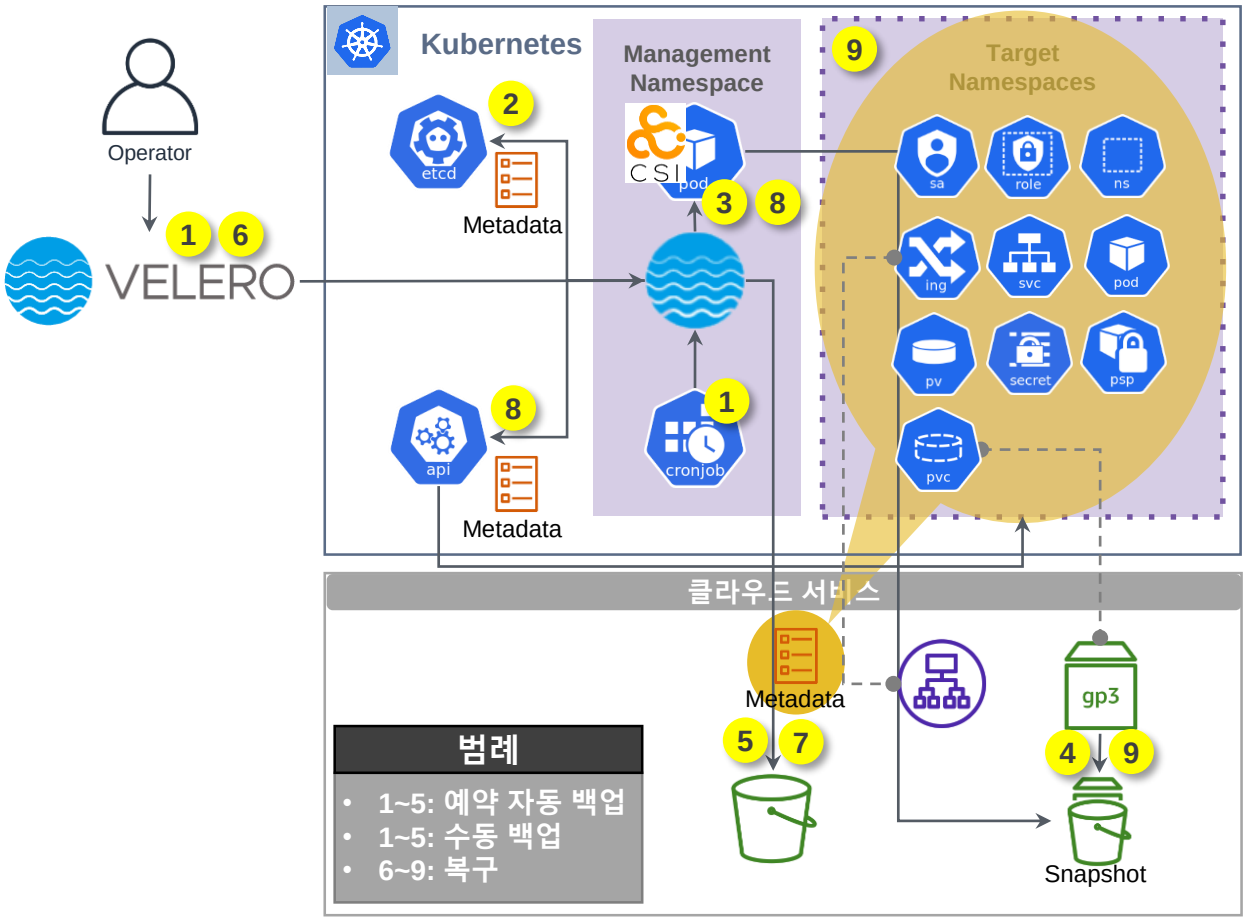
적용 기준

- **적용 대상: 에코 시스템, 애플리케이션**
- **적용기준**
  - ✔ 운영 환경의 네임 스페이스 별 예약 자동 백업
  - ✔ PV(Block)에 대해 백업 스냅샷의 자동 생성 및 비용 효율적인 생명 주기 적용
  - ✔ 백업 메타 데이터 저장소는 고가용성을 보장하는 클라우드 서비스의 스토리지 적용
  - ✔ 장애 복구 상황이 발생 시, 운영자에 의한 수동 복구 실시

MEGAZONE CLOUD

2. EKS Add Open Source

Backup & Restore



구성 원칙

- 운영 환경의 모든 애플리케이션 서비스에 대해 네임 스페이스 별 예약 백업을 적용
- 메타 데이터 저장소는 가용성 및 비용 효율적인 클라우드 제공자의 스토리지 서비스를 적용
- 백업 시스템 구성은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원

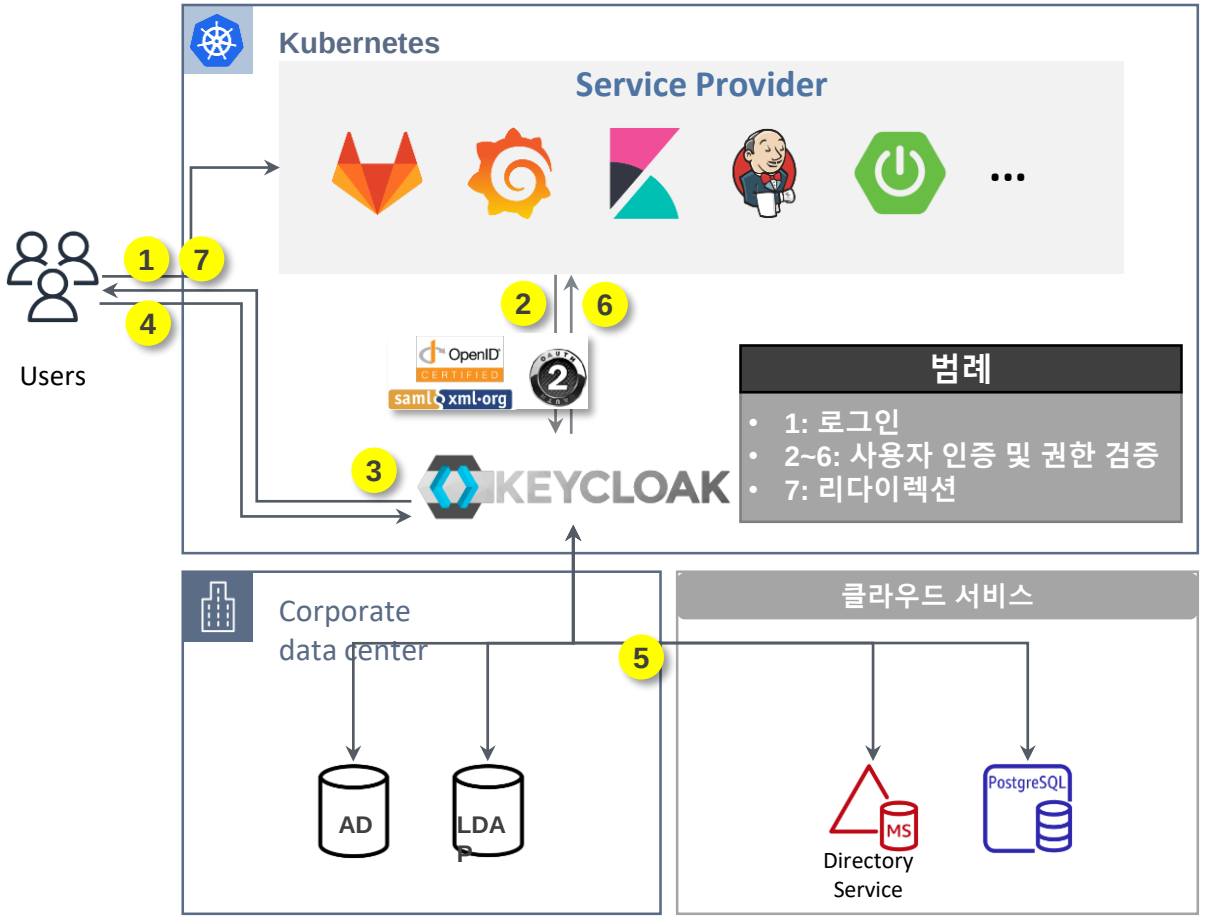
적용 기준

- **적용 대상: 에코 시스템, 애플리케이션**
- **적용기준**
  - ✔ 운영 환경의 네임 스페이스 별 예약 자동 백업
  - ✔ PV(Block)에 대해 백업 스냅샷의 자동 생성 및 비용 효율적인 생명 주기 적용
  - ✔ 백업 메타 데이터 저장소는 고가용성을 보장하는 클라우드 서비스의 스토리지 적용
  - ✔ 장애 복구 상황이 발생 시, 운영자에 의한 수동 복구 실시

MEGAZONE CLOUD

2. EKS Add Open Source

Integrated Authentication



구성 원칙

- 통합 인증 구성에 대해 고가용성을 적용
- 통합 인증 데이터에 대해서는 백업 솔루션을 통한 주기적인 백업을 적용
- 통합 인증 구성은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

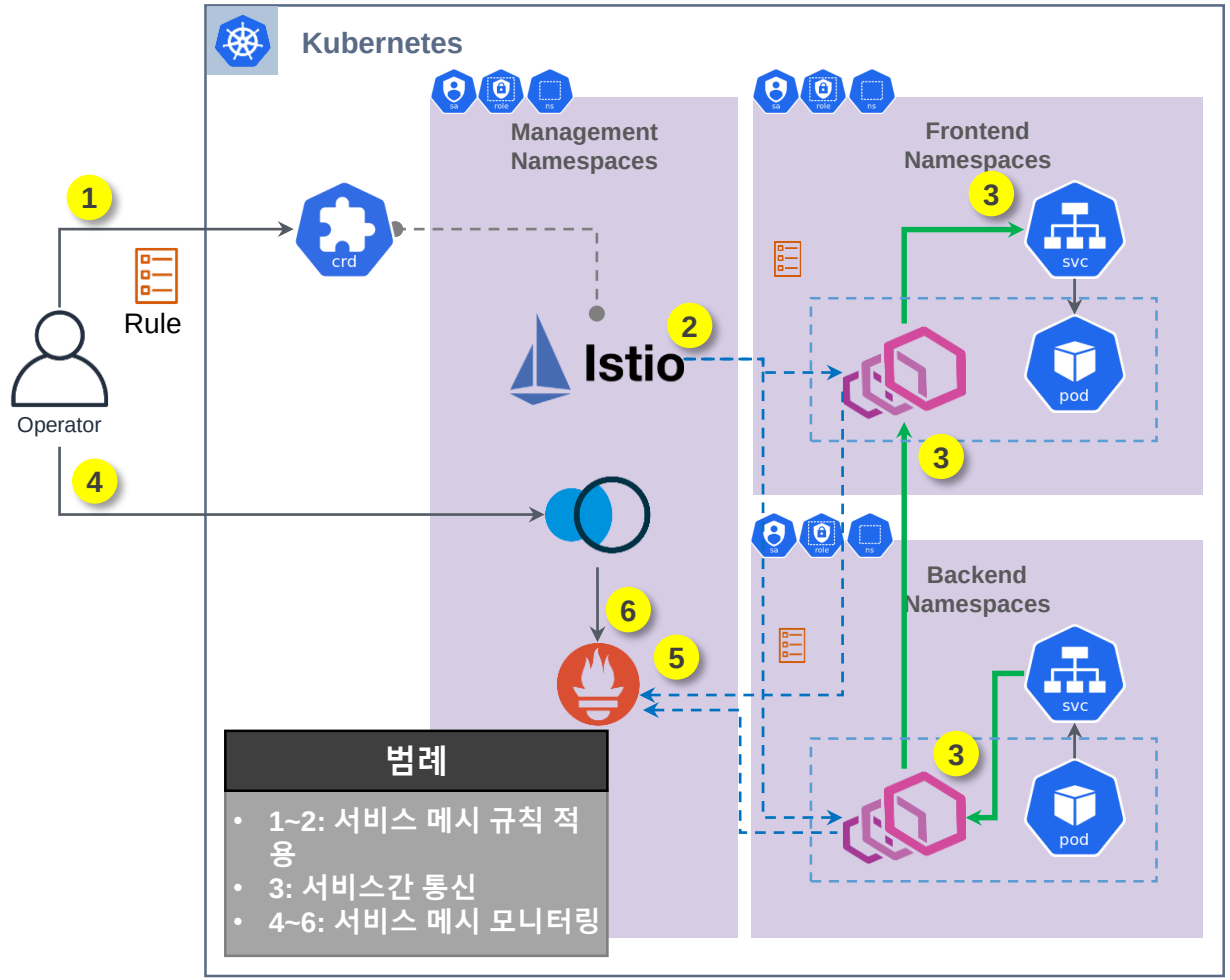
적용 기준

- **적용 대상 : KeyCloak, OpenID/SAML 2.0 지원 서비스, OAuth, AD, LDAP**
- **적용기준**
  - ✔사용자의 인증 및 인가가 필요한 OpenID, SAML를 지원하는 서비스
  - ✔기존의 통합 인증 시스템과의 통합이 필요한 경우 기술 지원 (AD, LDAP)
  - ✔통합 인증 데이터에 대해 예약 작업을 통한 주기적인 백업을 실시

MEGAZONE CLOUD

2. EKS Add Open Source

Service Mesh



구성 원칙

- 마이크로 서비스 간의 라우팅 구성 및 보안 성 강화 등 운영 간소화 지원
- 마이크로 서비스 간의 시각화, 고가용성 및 네트워크에 대한 실시간 모니터링 및 분석 기능 지원
- 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

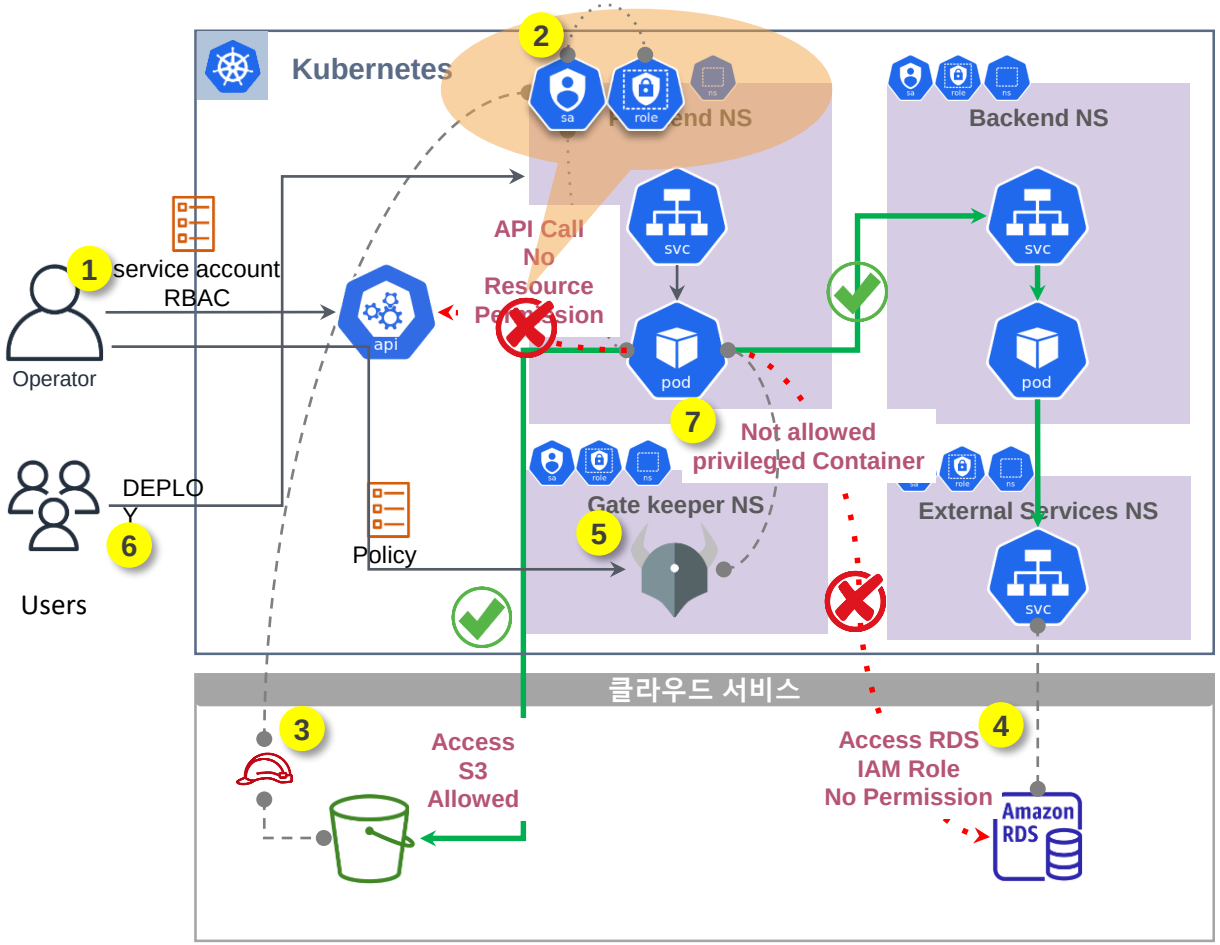
적용 기준

- **적용 대상 : Service Mesh(Istio, Envoy), Kiali**
- **적용기준**
  - ✔서비스 간 트래픽 흐름 방식을 구성하고 표준화하는 제어 기능을 제공
  - ✔서비스 간 사용자 지정 트래픽 라우팅 규칙을 지원하여 배포 중, 오류 발생 후 및 애플리케이션 확장 시 서비스의 고가용성을 보장
  - ✔서비스 간의 모든 요청에 대한 암호화 및 인증 제어 기능 지원
  - ✔모든 애플리케이션의 지표, 로그 및 트레이스를 캡처하여 이를 기반으로 한 시각화 기능 제공

MEGAZONE CLOUD

2. EKS Add Open Source

Security



구성 원칙

- Service Account에 RBAC 및 클라우드 서비스 IAM 역할을 적용하여 컨테이너에 최소 권한만 부여
- Privilege 및 Capability를 제한 등 Open Policy agent 적용을 통한 보안 강화
- 보안 정책은 코드 기반(IaC)으로 제공하여 향상된 운영 관리 환경을 지원함

적용 기준

- **적용 대상 : RBAC, 컨테이너 보안 정책, 클라우드 제공자 IAM 서비스, OIDC, OPA**
- **적용기준**
  - ✔ 컨테이너와 클라우드 서비스 간의 권한 인증을 위해 OIDC 적용
  - ✔ 클라우드 서비스 통합을 위한 최소 권한 IAM 및 Kubernetes RBAC 생성 및 Service Account에 할당
  - ✔ 모든 컨테이너에 Service Account 및 RBAC를 적용
  - ✔ 컨테이너 및 네임스페이스에 'Default' Service Account 사용 제한
  - ✔ OPA를 통한 Container securityContext 제한

MEGAZONE CLOUD

Agenda

1. Container Orchestration

- Container Orchestration 필요성
- Container Orchestration AWS EKS

2. EKS Add Open Source

- EKS & AddOn Structure
- CI/CD, Monitoring, Logging, Tracing, Backup, Integraded Authentication, Service Mesh, Security

3. API Gateway

- Ingress
- API GateWay

TYPE 2

AGENDA

01 Container Orchestration

Container Orchestration 필요성
Container Orchestration AWS EKS

02 EKS Add Open Source

EKS & AddOn Structure
CI/CD, Monitoring, Logging, Tracing, Backup, Integraded
Authentication, Service Mesh, Security

03 API Gateway

Ingress
API GateWay

2. API Gateway

Ingress

Ingress 란 무엇입니까?

“ Ingress 클러스터 외부에서 내부 서비스로 접근하는 HTTP, HTTPS 요청들을 어떻게 처리할지 정의해 둔 규칙들의 모음을 말합니다. “

클러스터 외부에서 접근가능한 URL을 사용할 수 있게 하며, 트래픽을 로드밸런싱도 해주고, SSL 인증서 처리를 해주고, 도메인 기반으로 가상 호스팅을 제공하기도 합니다.

Ingress < Api GateWay (Endpoint -> innser, outer) != ServiceMesh (Inner, outer)

Ingress 는 단일 진입점에 Client – Server 간에 서비스 통신 간소히 하고 , HTTP Path를 이용한 서비스 Route, HTTP Heade 조작

ApiGateway 는 Ingress 기능위에 Session관리, Serkit Bracker, 권한, 보안, 모니터링 등의 추가기능을 넣어 향상

Service Mesh 는 분산 Application 간(MSA) 내부간의 서비스 통신을 관리, 분산추적, 보안 , 로깅, 부한분산 기능을 안전하게 처리

CORS: cross-origin
Rest Api Access-control-Header

MEGAZONE CLOUD

2. EKS Add Open Source

API Gateway

API Gateway 란 무엇입니까?

API Gateway는 트래픽 관리, CORS 지원, 권한 부여 및 액세스 제어, 제한, 모니터링 및 API 버전 관리 등 최대 수십만 개의 동시 API 호출을 수신 및 처리하는 데 관계된 모든 작업을 처리합니다

인증

여러 서비스의 데이터에 액세스해야 하는 경우에도 게이트웨이에서 한 번만 인증하면 됩니다.
이를 통해 대기 시간이 줄어들고 애플리케이션 전체에서 인증 프로세스가 일관됩니다

입력 유효성 검사

고객 요청에 요청을 완료하는 데 필요한 필수 정보가 포함되어 있고 올바른 형식으로 제공되는지 확인하는 API 게이트웨이 역할

의심되는 것으로 보이면 게이트웨이는 요청을 거부합니다

메트릭 수집

API 게이트웨이는 사용자의 요청 수 또는 각 개별 마이크로서비스에 릴레이되는 요청 수를 측정
사용자가 너무 많은 요청을 보내는 경우 게이트웨이는 요청을 마이크로서비스 중 하나로 전달하는 대신 거부하도록 프로그래밍

응답에 대한 전달성

모바일 애플리케이션은 예를 들어 게이트웨이가 요청에 대한 올바른 응답을 제공할 수 있게 일반적으로 웹 애플리케이션보다 적은 데이터를 필요로 합니다

XML로 응답을 포기했지만 요청자가
JSON으로 응답을 필요로 하는 경우 게이트웨이가 이를 자동으로 처리

MEGAZONE CLOUD

2. EKS Add Open Source

API GateWay

API GateWay Session 관리

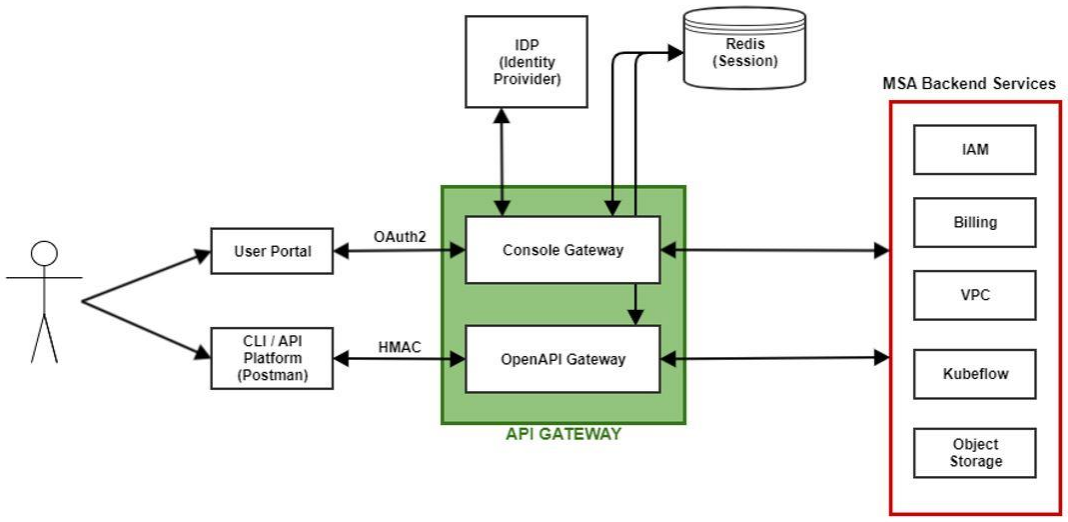


그림. 3 서비스 내의 API 게이트웨이 구성도

1. IaaS API 게이트웨이 :

AWS 람다등을 이용하여 DynamoDB 를 통해 세션유지 ,
AWS Service 종속적

2. 오픈 소스 기반의 API 게이트웨이 SW 사용 :

플러그인 사용 하거나 또는 원활한 기능 지원의 어려움

3. 오픈 소스 기반의 커스터마이즈 된 API 게이트웨이 개발시 :

OAuth2 기반의 Identity Provider(IDP)를 이용한 OAuth2 사용자 인증,

대량의 사용자 인증 요청에 대응하기 위해 session 관리를 Redis 기반의 인증 Session 사용

MEGAZONE CLOUD

2. EKS Add Open Source

Kong VS AWS API Gateway

AWS ApiGateWay VS Kong

분류	**Kong**	**AWS API Gateway**
기능	설치형, UI 오픈소스 konga, 클라우드 제공업체 구애받지 않음 CI/CD 사례지원, 스트리밍 지원안됨. 전용 API 설계 솔루션, k8s RBack 지원, Prometeus , datadog 같은 써드파티 모니터링 솔루션으로 관리및 자체 모니터링, nginx 이라 elk , loki 등으로 수집	AWS 통합, 웹소켓, 서비리스 방식으로 관리포인트 없음. restapi CI/CD 미지원, Lamda 함수통합, 비용 추가발생 Websocket API 제한지원, 안정적으로 사용 amazon cloud watch 로 모니터링, aws api gateway cache지원 DDos 지원 , 사용량 쓰는 만큼 비용 지불
단점	AWS 여러 Cluster 사용하면 전부 설치 해야 함. 엔터프라이즈 도입시 비용 고려, 관리의 단점 (업그레이드) Cache 를 사용하려면 Enterprise 가입	Soap 미지원, 라이선스 비용 없음. 사용량 부과

MEGAZONE CLOUD