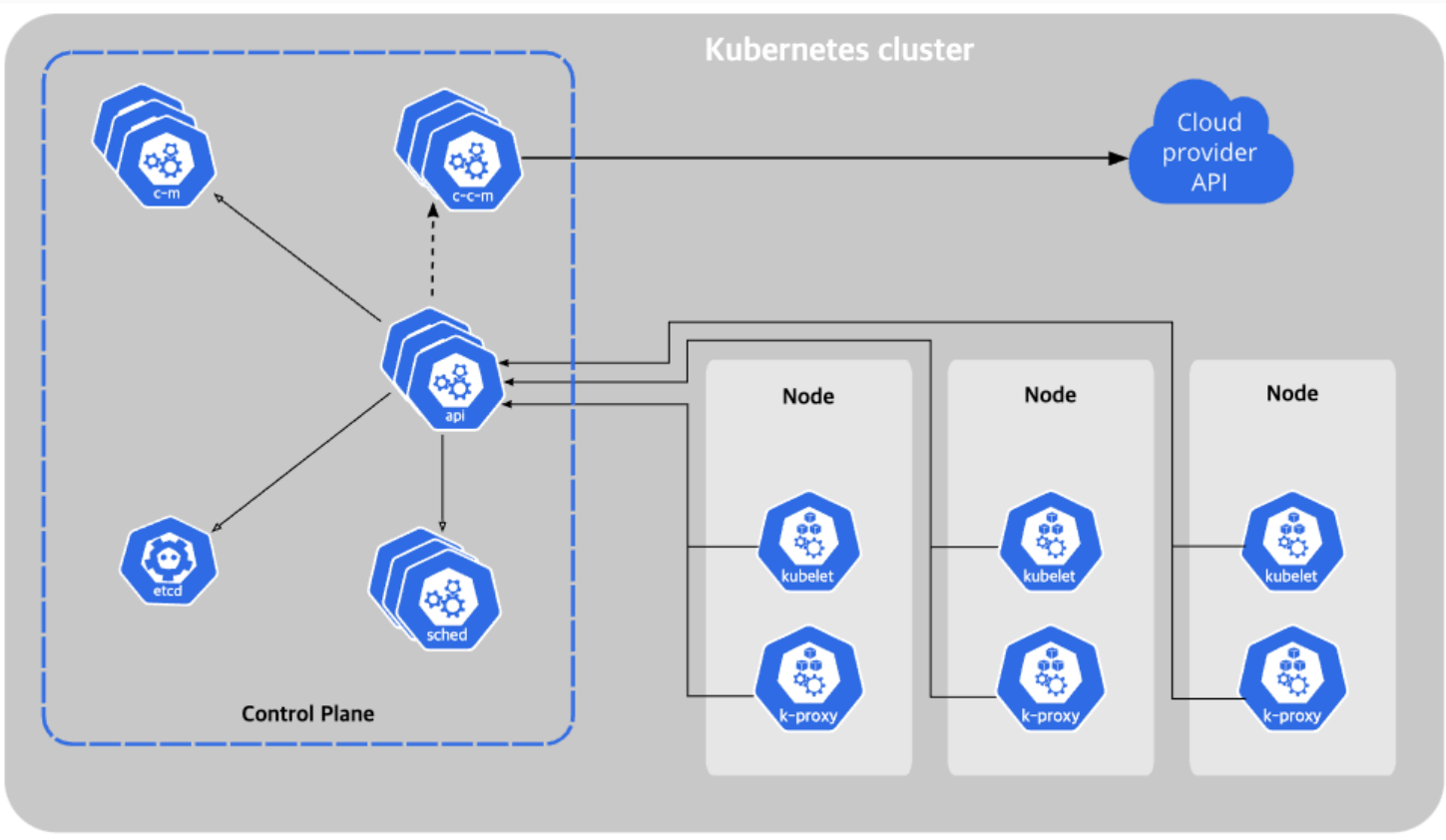
Amazon EKS 란?

Amazon EKS 관리형 서비스는 **컨트롤 플레인을 직접 구성하지 않고서 k8s를 손쉽게 사용할 수 있도록 편리함을 제공**

AWS에서 제공하는 **VPC, ELB, IAM 등 특정 기능들을 같이 활용하고자 할 때 유용**



Amazon EKS 특징

VPC(Virtual Private Cloud) 와 통합

일반적으로 쿠버네티스 클러스터에서는 파드 네트워크로 워커 노드의 네트워크와는 다른 자체 네트워크 체계를 배치.

클러스터 자체 네트워크를 사용하기 때문에 명시적으로 엔드 포인트를 설정하지 않으면 **클러스터 외부에서 파드에 통신이 불가능**

Amazon EKS에서는 Amazon VPC 통합 네트워킹을 지원하고 있어 파드에서 VPC 내부 주소 대역을 사용할 수 있고 **클러스터 외부와의 통신이 가능**하고 심리스(Seamless)하게 구현할 수 있다.

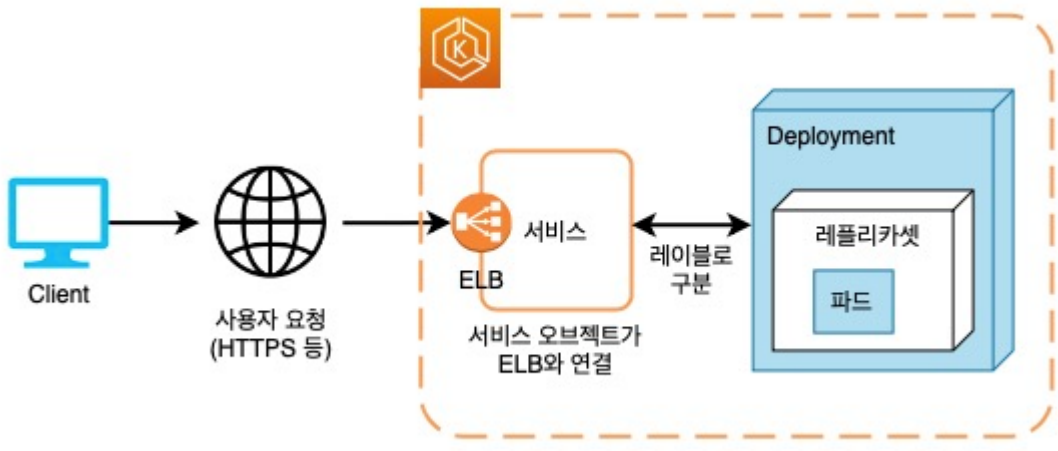
cf) 심리스(Seamless): 서비스 접근을 단순하게 하는 것 혹은 복잡한 기술이나 기능을 설명하지 않아도 서비스 기능을 직관적으로 구현하는 뜻

ELB와의 연계

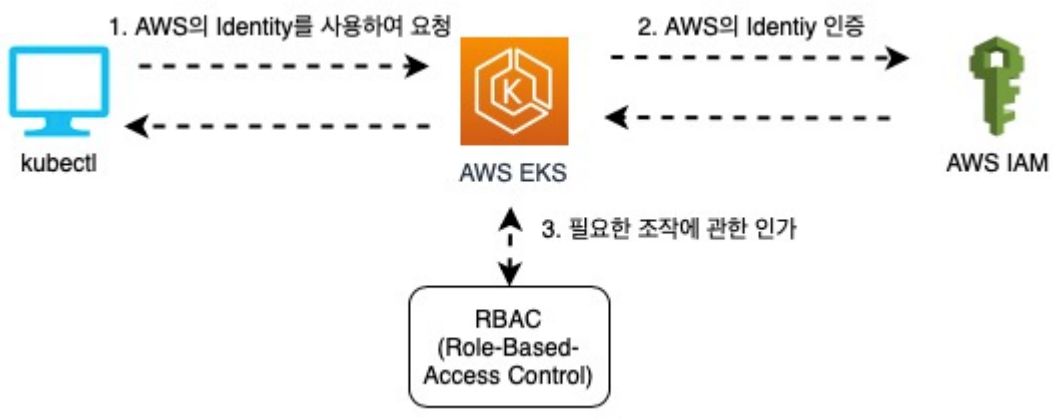
k8s 클러스터 외부에서 접속할 때는 서비스를 사용해 엔드포인트를 생성할 필요가 있음

가장 전형적인 엔드포인드가 로드밸런서

EKS 에서는 k8s의 서비스 타입 중 하나인 LoadBalancer를 설정하면 **자동적으로 AWS의 로드밸런서 서비스인 ELB가 생성**
이것으로 HTTPS나 경로 기반 라우팅 등 L7 로드밸런서 기능을 AWS 서비스로 구현 가능



IAM 을 통한 인증과 인가



쿠버네티스 클러스터는 kubectl 이라는 명령줄 도구를 사용하여 조작.

이때 **해당 조작이 허가된 사용자에 의한 것음을 올바르게 인증**해야 함.

또 인증된 사용자에게 **어떤 조작을 허가할지에 대한 인가 구조**도 필요

EKS 에서는 IAM과 연결한 인증 및 인가 구조를 제공

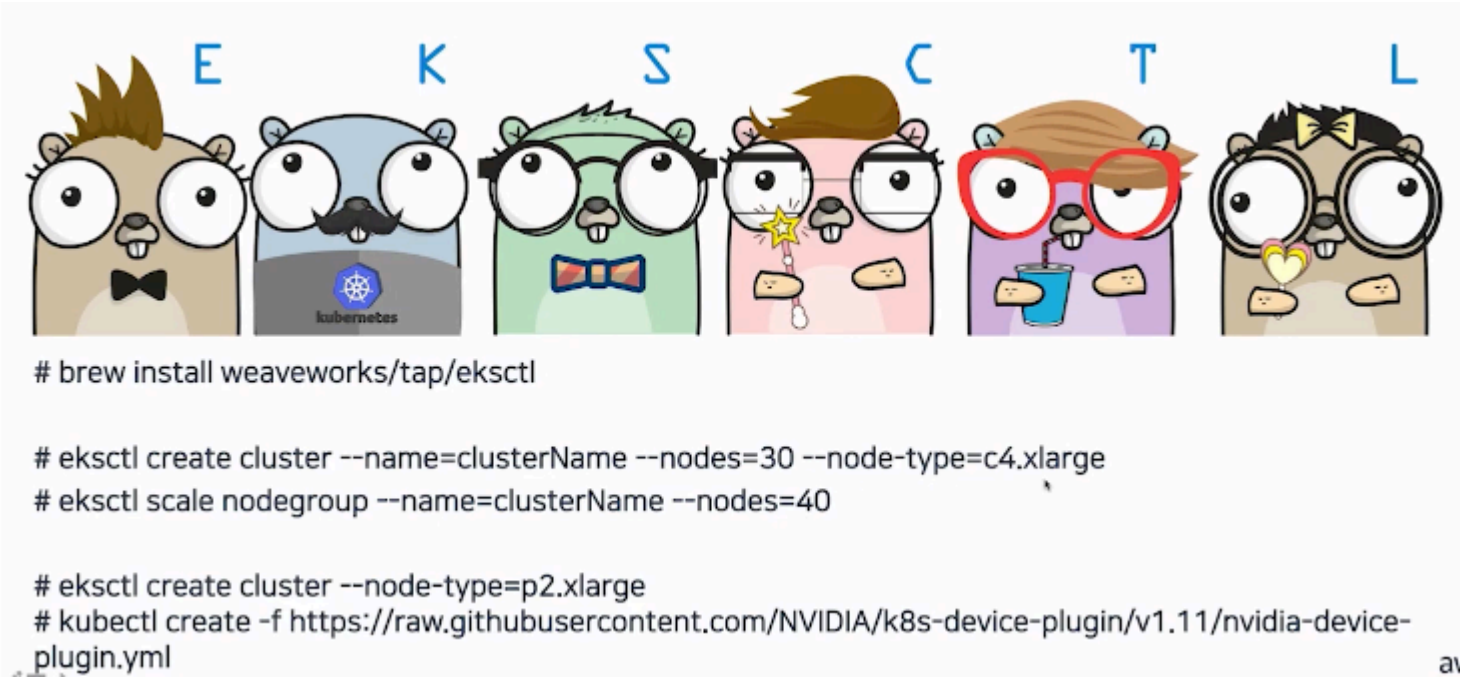
워커 노드

EKS workflow 를 살펴보자

EKS 클러스터 컨트롤 플레인 생성 부분은 EKS에서 관리형 서비스로 제공하고 있다.

워커 노드에 대한 서비스도 제공. ex) AWS에서 제공하는 서비스 **eksctl**

반대로, 워커노드를 Customize 해서 원하는 서비스를 생성할 수 있음



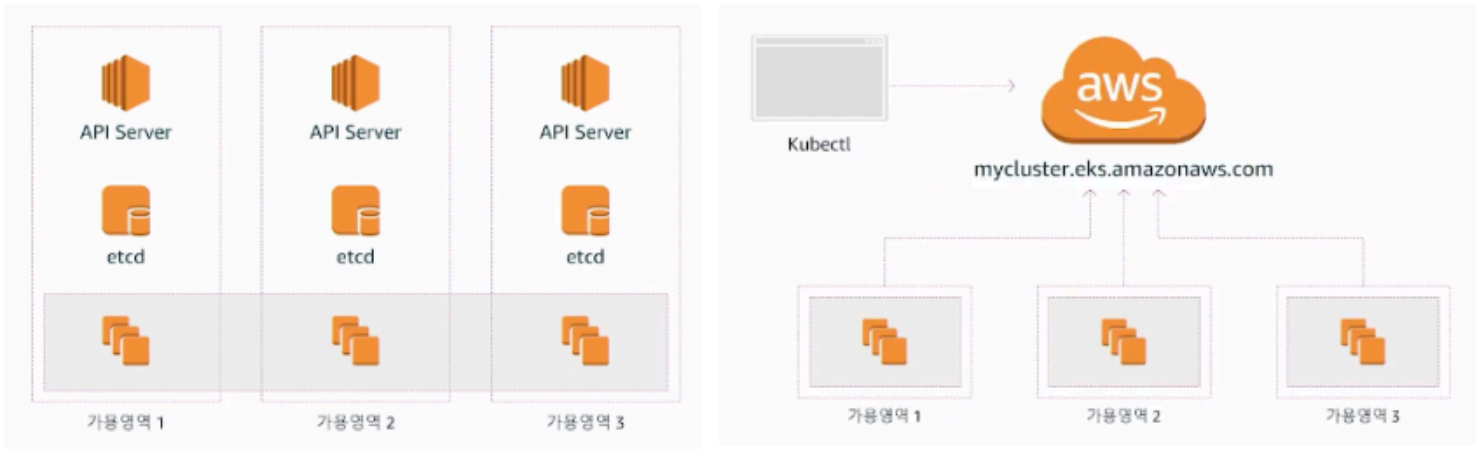
EKS 클러스터의 유지 관리나 버전 업그레이드할 때 필요한 가상 머신 설정을 쉽게 해주는 관리형 노드 그룹 구조와 가상 머신을 의식하지 않고 파드를 배포할 수 있는 **파게이트(Fargate)**

하지만, 파게이트는 워커 노드 관리가 필요 없는 만큼 파드가 배포되는 호스트에 사용자 접근도 제한되며 거기에 따른 제약도 있다.

	Amazone EC2	Amazon Fargate
관리 부하	높음	낮음
제약	적음	많음

특징	1. Amazon EC2 KES와 별도로 EC2 관리 2. 관리형 노드 그룹 EC2임에는 변함이 없지만 EC2에 통합된 워커노드로 동작. 유지보수나 버전을 업데이트할 때 필요한 조작이나 고려 사항들을 자동적으로 실행	워커 노드도 완전히 AWS가 관리하며 사용자는 EC2를 관리하지 않아도 됨

Amazon EKS 아키텍쳐

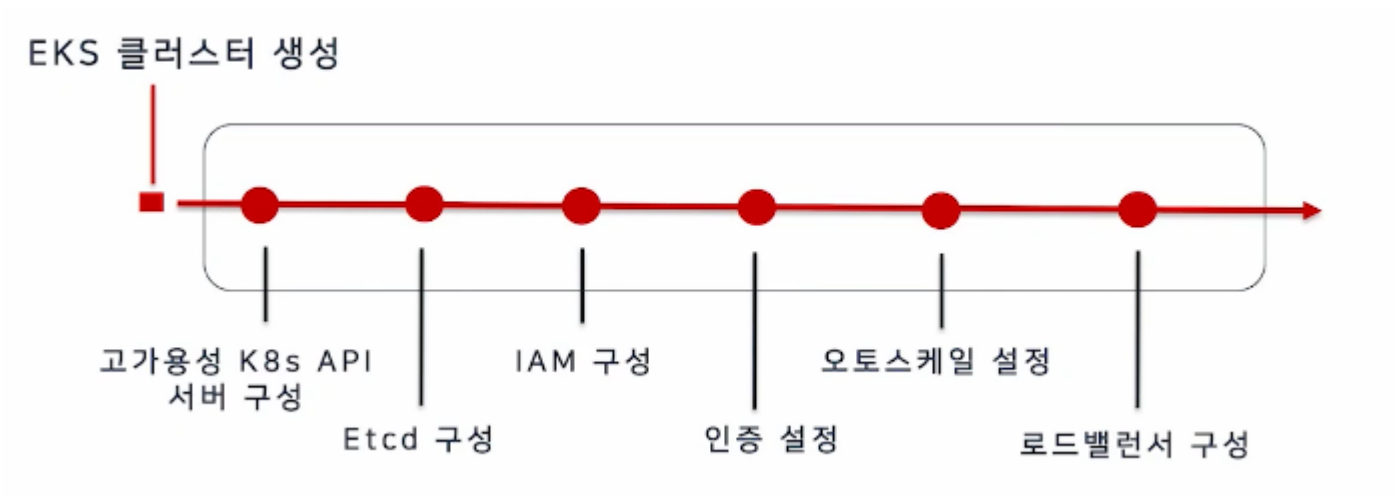


EKS 를 배포하게 되면

왼쪽 그림처럼 가용영역별로 API Server와 etcd 와 컨테이너가 배포되게 됩니다. **(고가용성 보장)**

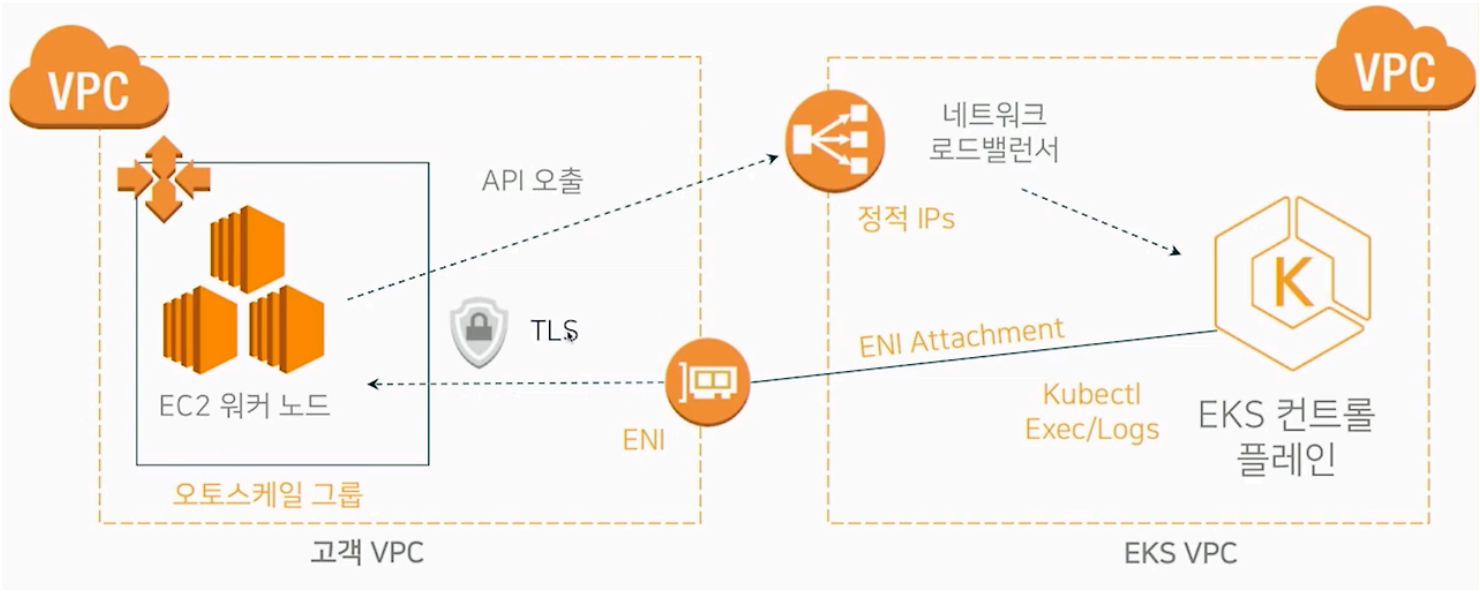
하지만, 사용자 입장에서는 하나의 마스터 엔드포인트로 보입니다.

AWS Side Workflow

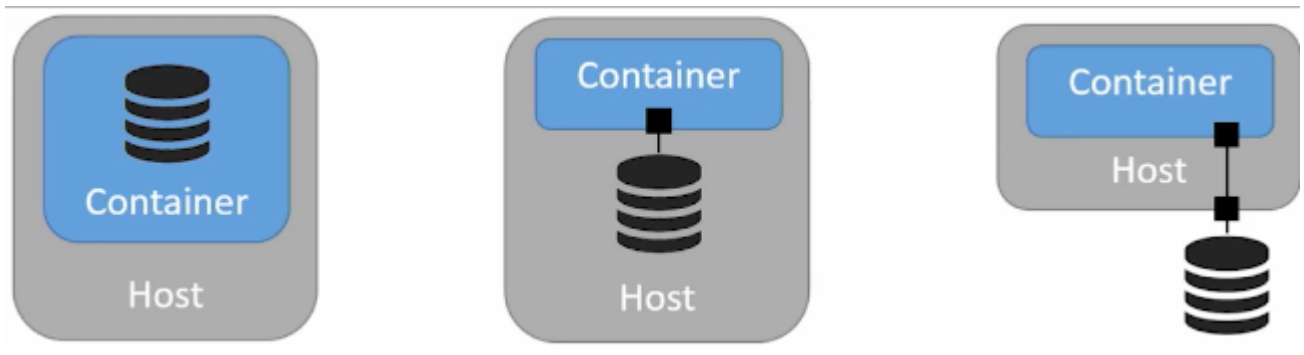


EKS 클러스터를 사용자가 생성하면 AWS side에서 쿠버네티스 클러스터를 생성하기 위해 위와 같은 workflow를 거침.

- 쿠버네티스의 api 서버를 각 가용영역에 배포
- api 데이터, 쿠버네티스의 상태 데이터를 확인하기 위해 etcd를 같이 배포
- 쿠버네티스에서 오는 call 에 대한 IAM 구성
- 쿠버네티스 마스터 노드의 오토스케일링 설정
- 클러스터가 안정적으로 구현하도록 여기에 연결할 수 있는 로드밸런서를 구성



스토리지



3가지의 스토리지 케이스가 있음.

1. 컨테이너의 볼륨

- 컨테이너가 종료되면 손실
- 호스트 노드의 메모리 활용 가능 (임시 캐시)

2. 호스트의 볼륨

- 컨테이너가 다른 호스트 재배치되면 접근 불가
- 컨테이너 이미지를 경령화 하기 위해 사용

3. 네트워크 볼륨

- 호스트에 독립적
- 컨테이너가 다른 쪽으로 스케줄링 되도 문제없음

1,2는 호스트의 가용성에 의존되어 있음.

Stateless 애플리케이션에 적합

Stateful 애플리케이션은 Persistent 볼륨이 필요

PersistentVolume(PV)

독립적인 라이프사이클이 있는 볼륨 플러그인

NFS, iSCSI 혹은 클라우드 제공 스토리지

PersistentVolumeClaim(PVC)

PV를 요청하는 오브젝트

용량과 access mode 정의

Provisioning

Static: 미리 PV를 생성하고 PVC Claim 기반으로 특정 PV를 요청

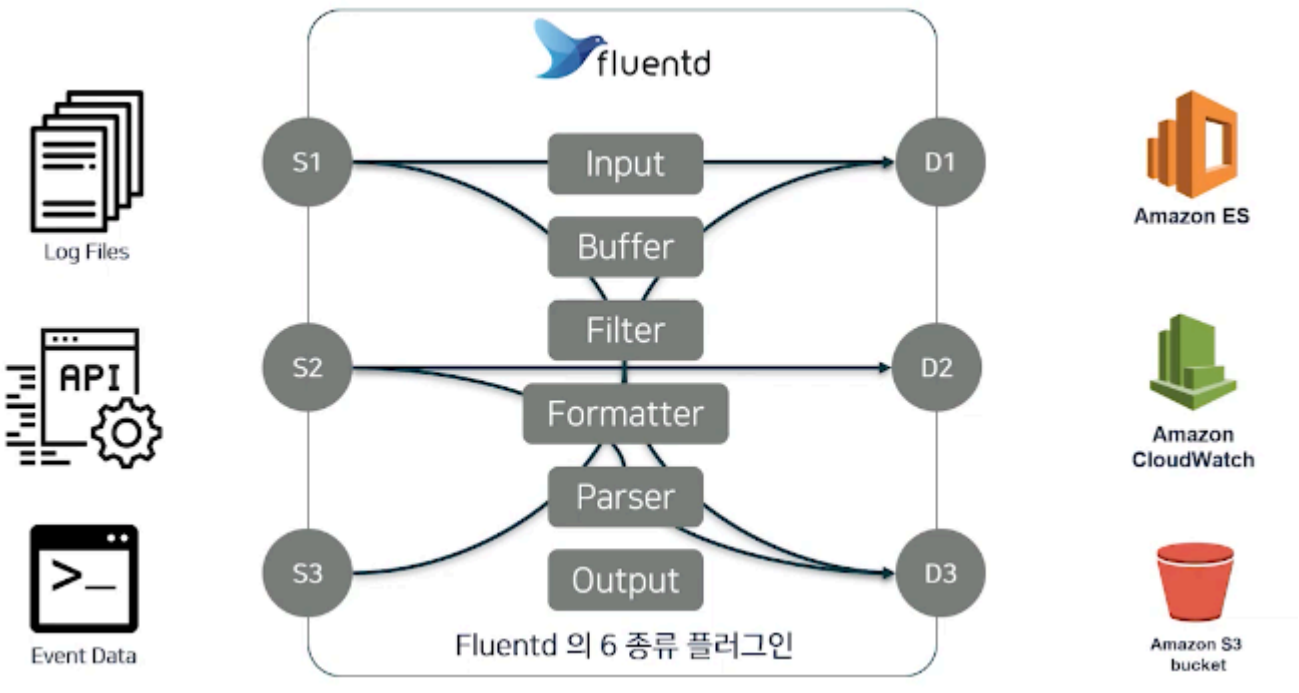
Dynamic: 미리 정의한 StorageClass에 기반하여 PVC가 동적으로 PV를 할당

Temp	Host Local	Network
emptyDir	hostPath	AWS EBS gcePersistentDisk azureDisk GlusterFS gitRepo NFS ISCSI Fiber Channel VshereVolume

로깅 & 모니터링

로깅

아마존 EKS 에서 Fluentd 를 사용해 로그를 모을 수 있는 아키텍쳐를 제공



다양한 소스에 있는 **fluentd** 에서 모아서 JSON 형태로 모아 스토리지 영역으로 바이패싱

스토리지 영역은 ES, S3 로 사용 가능

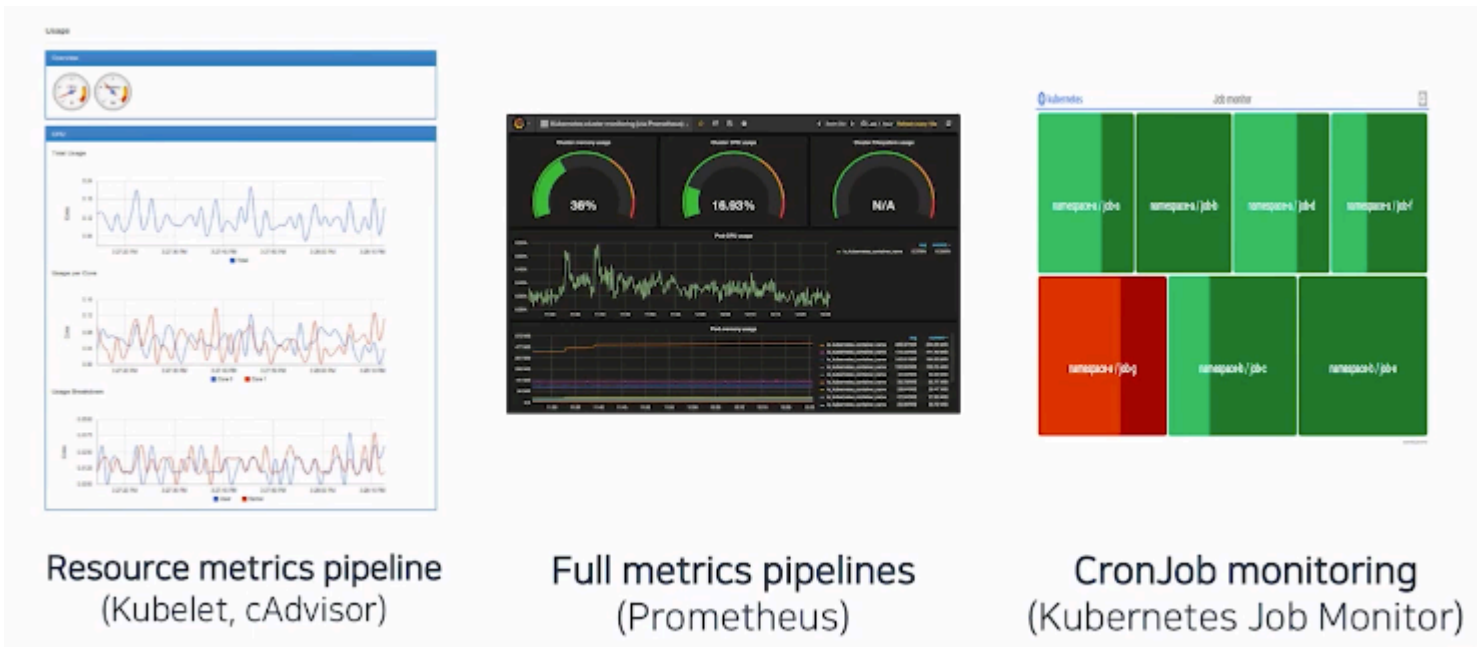
데이터가 들어오는 시점에 **Cloudwatch** 를 사용해 전송 패턴을 가지고 들어오게 할 수 있음.

fluentd 가 데몬셋처럼 인스턴스 1개 당 배포.

인스턴스에 있는 파드에서 배포된 서비스에서 나오는 Standard Output log 들을 **fluentd가 ES**로 전송

ES 에 있는 데이터들은 필드형태로 **kibana 에서 시각화**할 수 있음

모니터링



쿠버네티스는 3가지 정도의 모니터링 리소스를 소개

1. Resource metircs pipeline

노드들 상태정보, **노드에서 사용하고 있는 cpu,메모리,디스크 사용정보**를 마스터 노드로 전송

인스턴스에서 파드에 사용하는 전체사용량이지 각각 애플리케이션(서비스) 전체 사용량을 볼 수는 없음

서비스에서 사용하는 정보를 얻고 싶다면

2. Full metrics pipelines 프로메테우스를 사용

일반적으로 쿠버네티스에서 대시보드에서 보는 **3. CronJob monitoring**

전체 클러스터의 상태 정보를 폴링형태로 보여주는 것

프로메테우스 아키텍쳐 개요