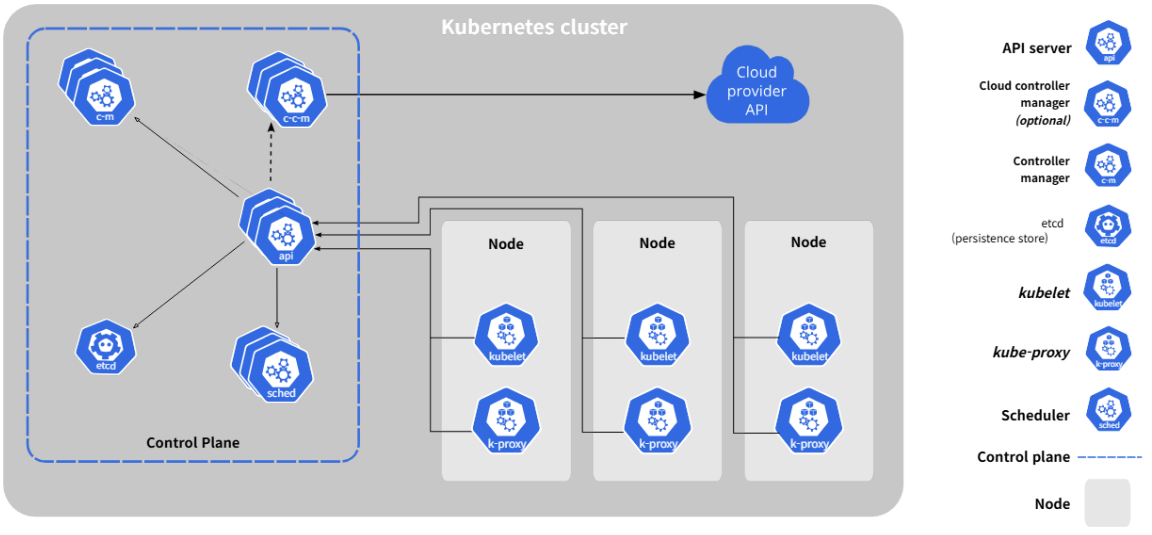
[Kubernetes] 쿠버네티스(Kubernetes)의 구조

원본 스크랩: [Kubernetes] 쿠버네티스(Kubernetes)의 구조

Control Plane

Control Plane은 기본적으로 쿠버네티스 cluster 내의 worker node 와 Pod(쿠버네티스는 한개 또는 여러개의 container를 Pod이라는 object 단위로 노드에 배포한다)을 관리하는 기능을 수행한다.
예를 들어 worker node 에서 돌아가고 있는 서버 application 한개가 죽었다면, Control Plane 에서 이를 감지하고 새로운 서버 application 을 배포하여 쿠버네티스 사용자의 요구사항을 지속적으로 충족시킨다.
Control Plane 이 배포되어 있는 노드를 **마스터 노드**라고 부르며, *Control Plane 이 망가지는 순간 쿠버네티스 cluster 를 관리할수가 없게되므로 일반적으로 Control Plane 은 고가용성 (High availability)을 위해 3개 이상으로 구성하는것이 추천*된다.
또한 마스터 노드에는 사용자의 application을 배포하지 말고 Control Plane 만 배포하여 마스터 노드가 안정적으로 운영되게 하는 것이 중요하다.
그렇다면 내부적으로 어떤 component가 각각 어떤 일을 수행하고 있는지 살펴보자.



kube-apiserver

쿠버네티스의 API 를 외부로 노출시켜주는 component 이다.

사용자는 쿠버네티스의 API를 호출하여 kube-apiserver 에 request를 보내거나, kubectl 이라는 CLI tool 을 사용하여 kube-apiserver에 request 를 보낼 수 있다.
kube-apiserver는 사용자 또는 내부 다른 component의 request를 받아 현재의 상태를 확인하거나 추가적인 명령을 다른 component 에 전달하여 수행하도록 한다.
예를 들어 Pod 의 갯수를 보는 request 또는 Pod 생성을 하는 request 등등 모든 request 는 이 kube-apiserver에서 받아서 처리하게 된다.

etcd

key-value 데이터베이스이며 cluster 내의 정보들을 저장한다.
따라서 사용자가 쿠버네티스 API나 kubectl을 사용하여 특정 정보를 얻고자 할때, kube-apiserver가 request 를 받아 ETCD 에서 필요한 정보를 가져간 후 response를 내려준다.
ETCD는 오로지 kube-apiserver와 연결되어 정보를 제공하기 때문에, 다른 component들은 kube-apiserver를 통해서만 ETCD에 있는 정보를 가져올 수 있다.

kube-scheduler

사용자가 Pod을 배포하라고 명령을 내리면 kube-scheduler 가 해당 Pod이 어느 노드에 배포되야 하는지를 찾아준다.(실제로 Pod을 생성하진 않고 적절한 노드의 위치만 찾는 역할)
추후에 다루겠지만 Pod의 CPU, Memory 요청량이나 특정 filter 등등을 모두 고려하여 최적의 노드를 찾아준다.

kube-controller-manager

쿠버네티스에는 다양한 controller가 존재하는데 그것들을 관리하는 역할을 한다.
Node controller, Job controller 등등이 있으며 Node controller를 예로 들어보자면 노드의 상태를 주기적으로 모니터링하여 문제가 발생시 unhealthy 상태로 변경하여 트래픽을 보내지 않게 하는 등의 기능을 수행한다.
즉 각각의 Object에 해당하는 controller가 동작되면 해당 controller는 object를 지속적으로 모니터링하여 사용자가 원하는 상태로 유지시켜주는 것이 주된 기능이라고 할 수 있다.

Node Components

모든 워커노드에서 동작되고 있는 아래의 components에 대해 알아보자.

kubelet

각각의 노드마다 있는 agent이며 쉽게 생각해서 각 노드의 대표자라고 할 수 있겠다.
쿠버네티스 cluster에 노드를 등록하는 일을 수행하고, container runtime을 통해 Pod을 생성하며 지속적으로 monitoring을 하여 kube-apiserver에 전달해준다.

kube-proxy

각각의 노드마다 실행되고 있으며 network rule 을 관리하여 쿠버네티스 cluster의 내부 또는 외부의 network 연결을 도와주는 역할을 한다.

쉽게 예를 들자면 특정 network request에 대해서 해당 request가 어디로 가야하는지 도와주는 역할을 한다고 생각하면 되겠다.

Container runtime

실제 container를 실행하는 역할을 하는 소프트웨어이다.
kubelet이 Pod 생성을 요청하면 해당 Pod의 container 들이 이 container runtime에 의해서 실행되게 된다.

마무리

쿠버네티스의 구조와 각각의 component들에 대해서 간략히 살펴봤다.
AWS같은 cloud 환경에서 제공해주는 managed kubernetes service 를 사용하지 않고 직접 쿠버네티스 환경을 구성해 사용한다면 각각의 component들에 대해 조금 더 자세히 살펴보는 것을 추천한다.
운영중 발생하는 문제에 대해 조금 더 쉽게 원인 분석 및 해결이 가능할 것이다.
[참고]

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