- Kubernetes 일반적인 용어

- 쿠베네티스의 역사

- 마스터 노드 구성 요소

- 마이너(작업자) 노드 구성 요소

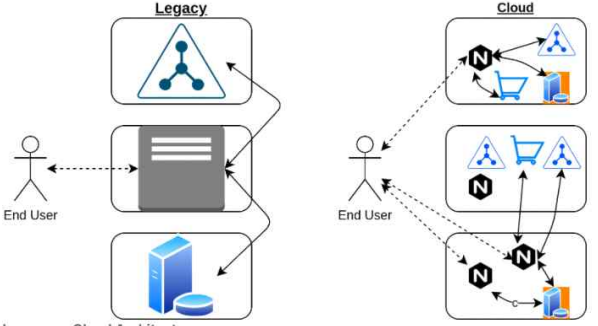
- CNI(컨테이너 네트워크 인터페이스) 구성 및 네트워크 플러그인

k8s

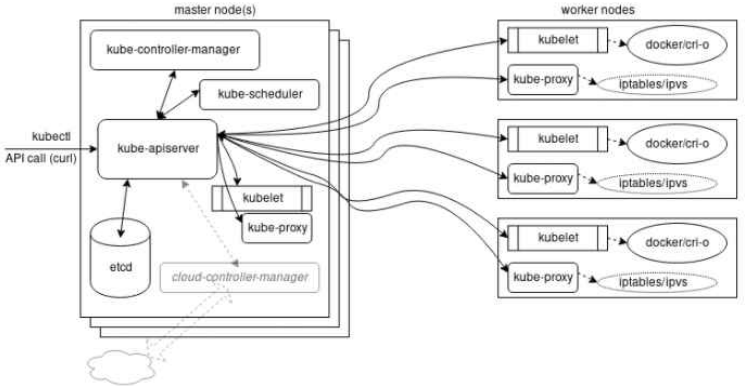
- 컨테이너형 애플리케이션의 배치, 확장 및 관리를 자동화하기 위한 오픈 소스 시스템
- 기본 요소 집합과 강력한 개방형 확장 API를 제공
- 새로운 오브젝트와 오퍼레이터를 추가할 수 있는 기능 => 쉽게 사용자 정의 (여러 스케줄러 및 여러 API 서버를 추가)
- 클러스터 사용과 분산 애플리케이션 관리의 효율성은 구글의 핵심 과제. 컨테이너는 클러스터를 효율적으로 패킹할 수 있는 세분화된 솔루션을 제공.
- 쿠버은 많은 필수 구성 요소를 제공하기 때문에 CI/CD(Continuous Integration/Continuous Delivery)에 필수적인 부분이 될 수 있음.

Components of Kubernetes

- 애플리케이션 배포에 대한 개발 및 시스템 관리 접근 방식의 변화



Kubernetes Architecture



Kubernetes Architecture

Challenges

- 컨테이너를 규모에 맞게 관리, 마이크로서비스의 원칙을 기반 -> 분산 애플리케이션을 설계하는 것

1) 연속 통합 파이프라인이 필요(컨테이너 이미지를 빌드하고 테스트)
2) 기본 인프라 역할을 하는 시스템 클러스터가 필요(컨테이너를 실행할 기본 인프라)
3)컨테이너를 시작, 장애가 발생시 스스로 해결할 수 있는 시스템이 필요
4) 롤링 업데이트 및 롤백을 수행, 불필요한 리소스를 해체
5) 유연, 확장 가능, 관리가 용이한 네트워크 및 스토리지가 필요
6) 네트워크는 다른 컨테이너로부터 트래픽을 보호하면서, 리소스를 다른 컨테이너에 넣을 수 있어야 함.
7) 스토리지를 원활하게 제공하고 보관 또는 재활용할 수 있는 스토리지 구조가 필요

- 가장 간단한 형태로 Kubernetes는 하나 이상의 중앙 관리자(마스터라고도 함)와 작업자 노드로 구성
- 관리자는 클러스터의 상태, 컨테이너 설정 및 네트워킹 구성을 유지하기 위해 API 서버, 스케줄러, 다양한 연산자 및 데이터스토어를 실행
- 사용자는 kubectl이라는 로컬 클라이언트를 사용하여 API와 통신
- kube-scheduler는 새 컨테이너 실행에 대한 API 요청을 보고 해당 컨테이너를 실행할 적절한 노드를 찾음
- 클러스터의 각 노드 : 두 개의 컨테이너 실행시킴, kubelet, kube-proxy.
- kubelet : 컨테이너 구성에 대한 사양 정보를 수신하고 필요한 리소스를 다운로드 및 관리하며, 로컬 노드의 컨테이너 엔진과 함께 작동하여 컨테이너가 실행되거나 장애 발생 시 재시작 되도록 함.
- kube-proxy : 네트워크의 컨테이너를 노출하기 위해 로컬 방화벽 규칙 및 네트워킹 구성을 생성하고 관리함.

Master Node

- 클러스터에 대해 다양한 서버 및 관리자 프로세스를 실행
- 구성요소 : kube-apiserver, the kube-scheduler, the etcd database

- => 마스터는 API 접점이며 앱, 포드, 개발 등의 관점에서 클러스터에서 진행되는 작업에 대한 정보를 보유

kube-apiserver

- 쿠버 클러스터 작동의 중심, 전체 클러스터의 마스터 프로세스로 작동하며 클러스터의 공유 상태의 프런트 엔드 역할.
  - 내부 및 외부 트래픽 모두 모든 호출은 이 에이전트를 통해 처리, 모든 작업은 이 에이전트에서 승인 및 검증되며 etcd 데이터베이스에 연결하는 유일한 에이전트
  - 각 API 호출은 인증, 권한 부여 및 여러 승인 컨트롤러의 세 단계를 거침.

kube-scheduler

- 알고리즘을 사용하여 Pod 를 호스트할 노드를 결정.
- 스케줄러는 바인딩할 사용 가능한 리소스(예: 사용 가능한 CPU)를 보고 가용성과 성공을 기준으로 Pod를 할당하려고 시도.
- 스케줄러는 기본적으로 Pod수를 사용하지만 클러스터 전체 메트릭이 수집되면 복잡한 구성이 수행되는 경우가 많음.

etcd Database

- 클러스터, 네트워킹 및 기타 영구 정보는 etcd 데이터베이스 또는 *b+tree* 키 값 저장소에 보관
- curl 및 기타 HTTP 라이브러리에서 작동하며 신뢰할 수 있는 시계 쿼리를 제공
- 특정 값을 업데이트하기 위한 동시 요청은 모두 kube-apiserver를 통해 이동하며, 이 서버는 요청을 따라 연속해서 etcd로 전달됨.

Other Agents

- kube-controller-manager는 cube-apiserver와 상호 작용하여 클러스터의 상태를 결정하는 코어 제어 루프 데몬이다. 상태가 일치하지 않으면 관리자는 필요한 컨트롤러와 연락하여 원하는 상태에 맞춥니다. 엔드포인트, 네임스페이스 및 복제와 같은 여러 컨트롤러가 사용되고 있습니다. 쿠베르네테스가 성숙해지면서 전체 리스트가 확대됐다.

Minion (Worker) Nodes

- 모든 작업자 노드는 Kubelet과 Kube-contract는 물론 Docker 또는 https-o와 같은 컨테이너 엔진도 실행함.
- 다른 관리 데몬이 배포되어 이러한 에이전트를 보거나 아직 쿠버에 포함되지 않은 서비스를 제공.

- => 컨테이너의 포드에서 앱 실행을 담당

pods

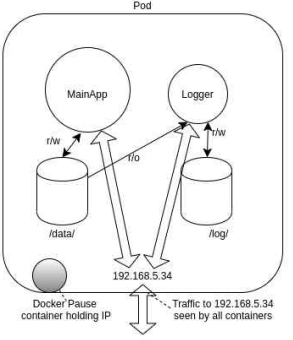
- 컨테이너의 집합 중 가장 작은 단위, 컨테이너의 실행 방법을 정의
- Pod의 설계는 일반적으로 컨테이너당 하나의 프로세스 아키텍처

services

- 파드의 집합에 접근하기 위한 경로를 정의
- 각 서비스는 여러 파드 간에 인바운드 요청을 분산하기 위한 단일 노드 포트 또는 LoadBalancer와 같은 특정 트래픽을 처리하는 마이크로 서비스
- 리소스 제어 및 보안에 유용한 인바운드 요청에 대한 액세스 정책을 처리
- 연결할 object를 알 수 있는 셀렉터 2개 : equality-based (레이블 키 및 해당 값을 기준), set-based(값 집합)

Single IP per Pod

- Pod는 일부 연결된 데이터 볼륨과 함께 배치된 컨테이너 그룹. Pod의 모든 컨테이너는 동일한 네트워크 네임스페이스를 공유.



Networking Setup

- 파드는 같은 ip 주소 공유하는 컨테이너 그룹.
- 파드는 vm과 비슷.
- 네트워크는 IP 주소를 포드에 할당해야 하며, 모든 노드의 모든 포드 간에 트래픽 경로를 제공해야 함.
- 컨테이너 조정 시스템에서 해결해야 할 세 가지 주요 네트워킹 과제
  - 컨테이너랑 컨테이너
  - 파드랑 파드
  - 외부랑 파드

- 파드들에는 애플리케이션 컨테이너가 시작되기 전에 IP 주소가 할당됨.
- 서비스 개체는 네트워크 내에서 ClusteriP 주소를 사용하여, nodeport 주소를 사용하여 클러스터 외부에서
- LoadBalancer 서비스로 구성된 경우 로드 밸런서를 사용하여 포드를 연결하는 데 사용