Section 9: Networking - 2

227. Pod Networking

- Pod Networking을 위한 요구사항
  - 모든 포드는 고유한 IP 주소를 가져야 함
  - 같은 노드 내의 포드들은 서로 IP로 통신 가능해야 함
  - 다른 노드에 있는 포드들도 같은 IP로 통신 가능해야 함

- 수동으로 하는 방법
  - 각 노드에 브릿지 네트워크 생성
  - 각 브릿지에 서브넷 할당 (예: 10.244.1.0/24, 10.244.2.0/24)
  - 컨테이너 생성 시 네트워크 네임스페이스 연결
  - IP 주소 할당 및 라우팅 설정

- 복잡한 환경에서는 모두 수동으로 하기 힘들기 때문에, CNI 기준에 맞게 스크립트를 작성하여 컨테이너가 생성될 때마다 자동으로 네트워크 설정

228. CNI in kubernetes

- `/opt/cni/bin` : network plugin이 설치되어있는 경로
- `/etc/cni/net.d` : 플러그인을 어떻게 사용할지 정의하는 구성파일들의 경로
- 여러 설정 파일이 있다면 알파벳 순으로 선택됨
- `cat /etc/cni/net.d/10-bridge.conf` (설정 파일 예시)
  - name: 네트워크 이름
  - type: 사용할 플러그인 유형 (ex. bridge)
  - bridge: bridge interface
  - ipMasq: NAT 규칙을 추가할지 정의
  - IPAM: 서브넷이나 pod에 필요한 경로에 할당될 IP 주소 지정

230. CNI weave

- 대표적인 CNI 플러그인 중 하나
- 각 노드에 대리인과 같은 역할을 하는 Weave Peer를 배포하고, 이들은 서로 통신해서 전체 토폴로지 정보를 공유함
- 각 노드에 Weave bridge 생성 후 IP address 할당

- `kubectl logs {node-name} weave -n kube-system` : 로그 확인

235. IP Address Management - Weave

- IPAM = IP Address Management = pod에 IP 주소를 할당/관리하는 시스템
- IP를 관리하는 가장 간단한 벙법은 각 노드에 파일로 IP 목록을 저장해두는 것인데, CNI는 이를 위한 플러그인을 제공함 (ex. host-local, DHCP)
- Weave의 IP 주소 관리
  - IP 범위는 기본적으로 10.32.0.0/12 (약 100만개..)
  - 이 범위 내에서 Peer는 각 노드에 IP 주소를 동등하게 할당해주고, 각 노드는 자기가 할당받은 범위 내에서 pod에게 IP를 할당해줌

238. Service Networking

- ClusterIP: 클러스터 내의 모든 pod에서 접근 가능
- NodePort: 클러스터 외부에서 Node IP와 port로 접근 가능
- kube-proxy: 각 노드에서 실행되며, 트래픽을 서비스 IP에서 포드 IP로 전달하는 규칙 생성
  - userspace, iptables, ipvs 모드

작동 방식

1. Service 생성 시 미리 정의된 범위 내에서 IP주소 할당 (기본 범위: 10.0.0.0/24)
2. IP 주소를 통해 kube-proxy가 각 노드에 IP table 규칙 생성 (이 Service IP로 오는 트래픽은 이 pod IP로 가야돼!)
3. 이제 Service IP로 요청이 들어오면 IP table에 정의된 규칙에 따라, 매칭되는 Pod IP로 트래픽 전달

- `iptables -L -t net` : NAT 테이블 규칙 목록 확인
- `ps aux | grep kube-api-server` : ip range 확인
- `cat /var/log/kube-proxy.log` : 로그 확인 (proxy 유형, ip range 등 확인)

241. DNS in kubernetes

- DNS Record 구조
  - Service: `{service-name}.{namespace}.svc.cluster.local`
    - 같은 namespace면 service명만 사용
  - Pod: `{IP주소를 . 대신 - 로 바꾼 값}.{namespace}.pod.cluster.local`

242. CoreDNS in Kubernetes

- `/etc/coredns/Corefile` : CoreDNS 설정파일 위치
- pod의 DNS 설정: `/etc/resolv.conf`에 nameserver로 CoreDNS Service IP 설정

245. Ingress

쿠버네티스 클러스터에서 외부에서 내부 서비스로의 HTTP/HTTPS 트래픽을 관리하는 API 객체

- 사용자가 외부에서 단일 URL 을 사용하여 application에 접근할 수 있도록 지원
- url 경로에 따라 클러스터 내의 여러 서비스로 트래픽을 라우팅하도록 수성 가능
- SSL 보안 구현 가능

- **Ingress Controller**: 배포하는 솔루션
  - 쿠버네티스 클러스터는 Ingress Controller를 기본적으로 제공하지는 않음
  - nginx-ingress image, 이를 노출시킬 service, nginx configuration data를 제공할 config map, 이 모든 객체들에 접근할 수 있는 알맞은 권한을 가진 service account 가 있으면 가장 간단한 Ingress Controller 생성 가능
  - ex) Nginx, GCE 등
- **Ingress Resources**: Ingress Controller에 적용되는 규칙 및 구성의 집합
  - 들어오는 모든 트래픽을 단일 application으로 간단하게 전달하거나, URL에 따라 다른 application으로 트래픽을 라우팅하도록 규칙을 구성할 수 있음
  - service, deployment 등 처럼 yaml 작성해서 만들면 됨

Ingress Resource yaml

1. **단일 서비스로 모든 트래픽 라우팅**:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear
spec:
  backend:
    serviceName: wear-service
    servicePort: 80
```

2. **URL 경로 기반 라우팅**:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear-watch
```

```
spec:
  rules:
  - http:
      paths:
      - path: /wear
        backend:
          serviceName: wear-service
          servicePort: 80
      - path: /watch
        backend:
          serviceName: watch-service
          servicePort: 80
```

3. **호스트 이름(도메인) 기반 라우팅**:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-wear-watch
spec:
  rules:
  - host: wear.my-online-store.com
    http:
      paths:
      - backend:
          serviceName: wear-service
          servicePort: 80
  - host: watch.my-online-store.com
    http:
      paths:
      - backend:
          serviceName: watch-service
          servicePort: 80
```

252. Introduction to Gateway API?

- Ingress의 한계
  - Multi-tenancy 환경에 대한 지원 부족
  - host matching, path matching과 같은 HTTP기반 규칙만 지원함
  - TCP/UDP Routing, 트래픽 분할, 헤더 조작, 인증, 속도 제한 등과 같은 기능 지원하지 않음
- Gateway API
  - 모든 Gateway API 구현체에서 일관되게 작동함
  - 컨트롤러별 annotation이 필요 없음

- Gateway API 구성 요소
  - **GatewayClass:** 기본 네트워크 인프라 정의 (NGINX, Traefik 등)
  - **Gateway:** Listener, protocol 등 정의
  - **HTTPRoute:** 트래픽 라우팅 규칙 정의