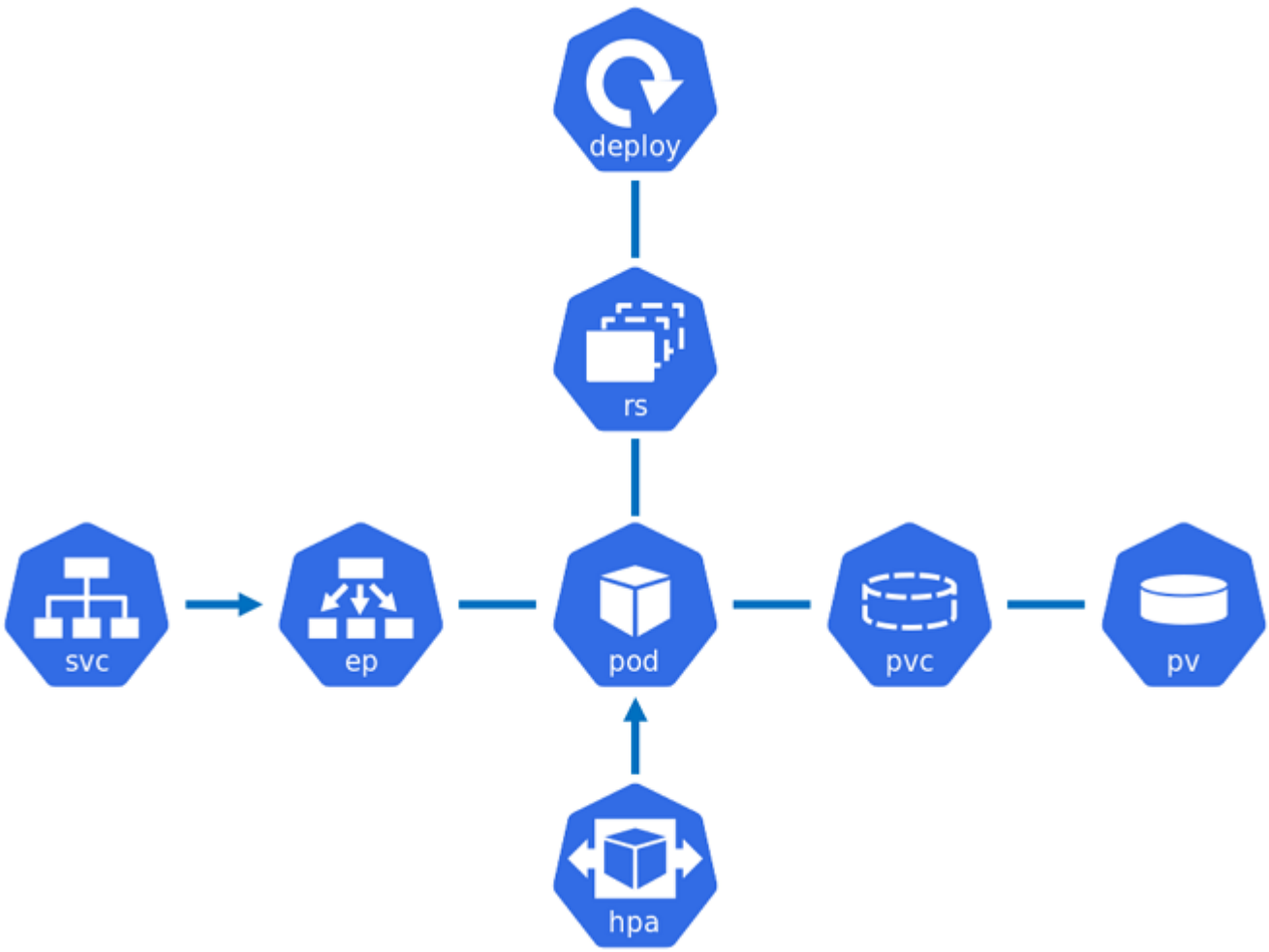
Hyperledger Fabric 네트워크 구성하기

Cello플랫폼팀 정연호

1. 응용프로그램 배포/운영을 위한 k8s 구성



리소스에 대한 설명은 사내 "Docker / K8S 개요 및 실습" 강의 자료를 참고하였으며 플랫폼개발그룹의 김경일 프로님이 작성하셨습니다.

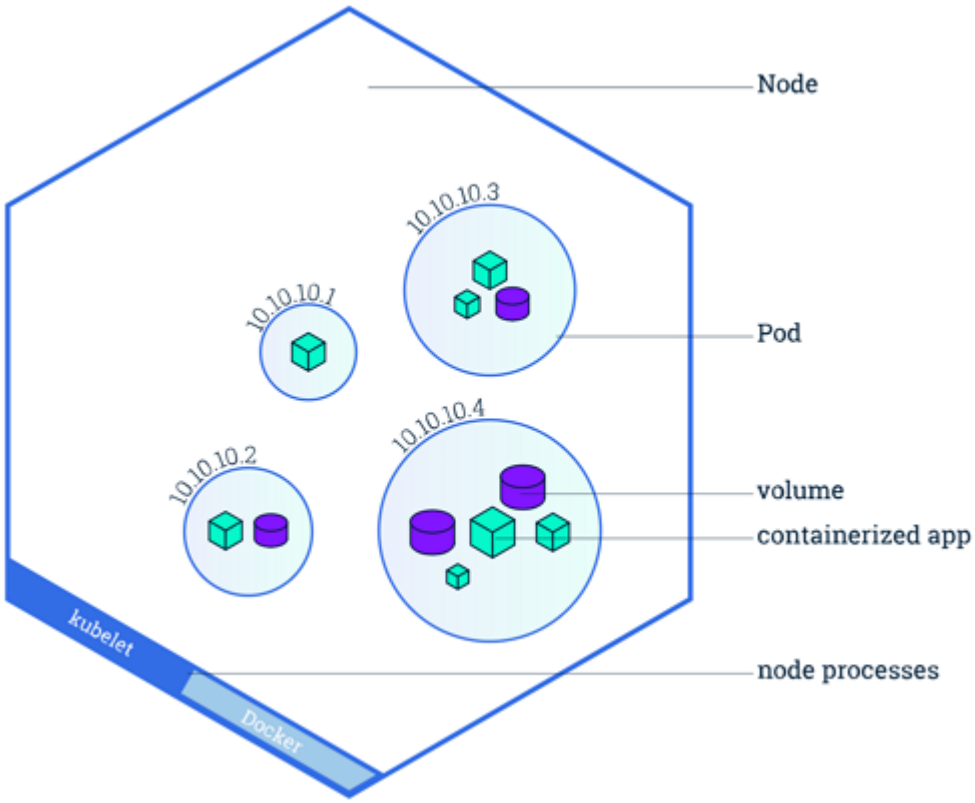
Pod

Pod는 k8s에서 이용되는 Container의 관리단위로 생성/관리/배포가능한 가장작은 단위의 유닛이며 클러스터내에서 실제 Application 구동되는 Object입니다.

Pod에 Container하나를 생성하는 것이 일반적이다.

Pod는 하나 이상의 Container로 구성되며, 동일한 Pod 내에서는 storage/network를 공유 할 수 있습니다.

k8s에서는 Docker가 가장 많이 쓰이는 Container Runtime 모듈이지만, 다른 Container Runtime 모듈도 지원합니다.

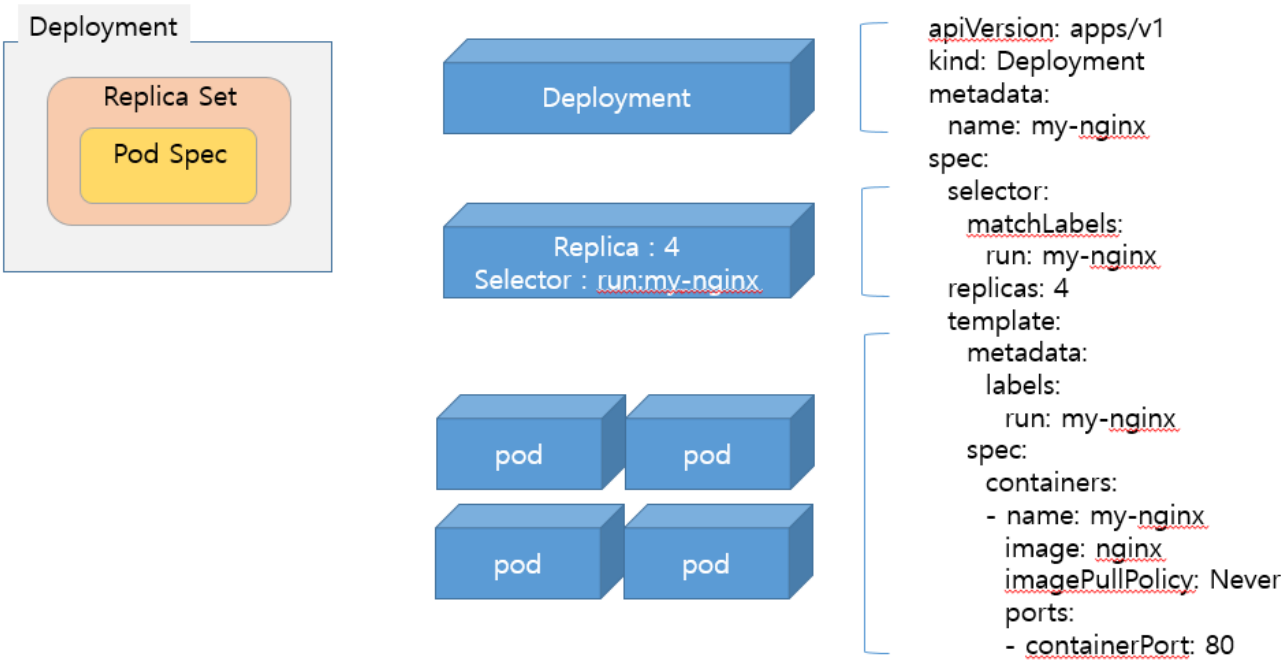


ReplicaSet

Label Selector를 통해 노드 상의 여러 Pod의 생성/복제/삭제 등의 라이프 싸이클을 관리합니다.

1. 지정한 Replica의 숫자만큼 Pod 수 생성/유지
2. Label Selector를 통한 Pod 타겟팅

Deployment



Deployment는 Pod를 관리하는 단위.

k8s에서 어플리케이션 단위를 관리하는 Controller 이며 k8s의 최소 유닛인 Pod에 대한 기준스펙을 정의한 Object이다.

k8s에서는 각 Object를 독립적으로 생성하기 보다는 Deployment를 통해서 생성하는 것을 권장하고 있으며, Pod와 ReplicaSet의 기준정보를 지정할 수 있다.

이러한 Deployment는

1. Pod의 scale in / out 되는 기준을 정의한다.
2. Pod의 배포되고 update 되는 모든 버전을 추적할 수 있다.
3. 배포된 Pod에 대한 rollback을 수행할 수 있다.

즉, 개념적으로 Deployment = ReplicaSet +Pod+history이며 ReplicaSet을 만드는 것보다 더 윗 단계의 선언(추상표현)이다.

Volume

컨테이너는 기본적으로 상태가 없는 상태에서 구동됩니다. 즉, 컨테이너가 실행되서 상태를 가져도 컨테이너가 종료되면 컨테이너에서 생성되었던 모든 데이터는 사라진다는 뜻입니다. 이것은 k8s에서 자유롭게 Pod를 복제하고 배포하는데 있어 커다란 장점입니다.

하지만, 로그 및 데이터베이스와 같은 애플리케이션은 종료되더라도 데이터가 유지되어야만 하는 특징이 있습니다. 이런 데이터의 상태를 유지할 수 있도록 사용하는 것이 Volume입니다.

k8s에서는 Docker와는 다르게 Pod 단위로 Volume을 관리하며, Life Cycle과 제공되는 디스크 Type 따라 다양한 옵션과 종류가 존재합니다.

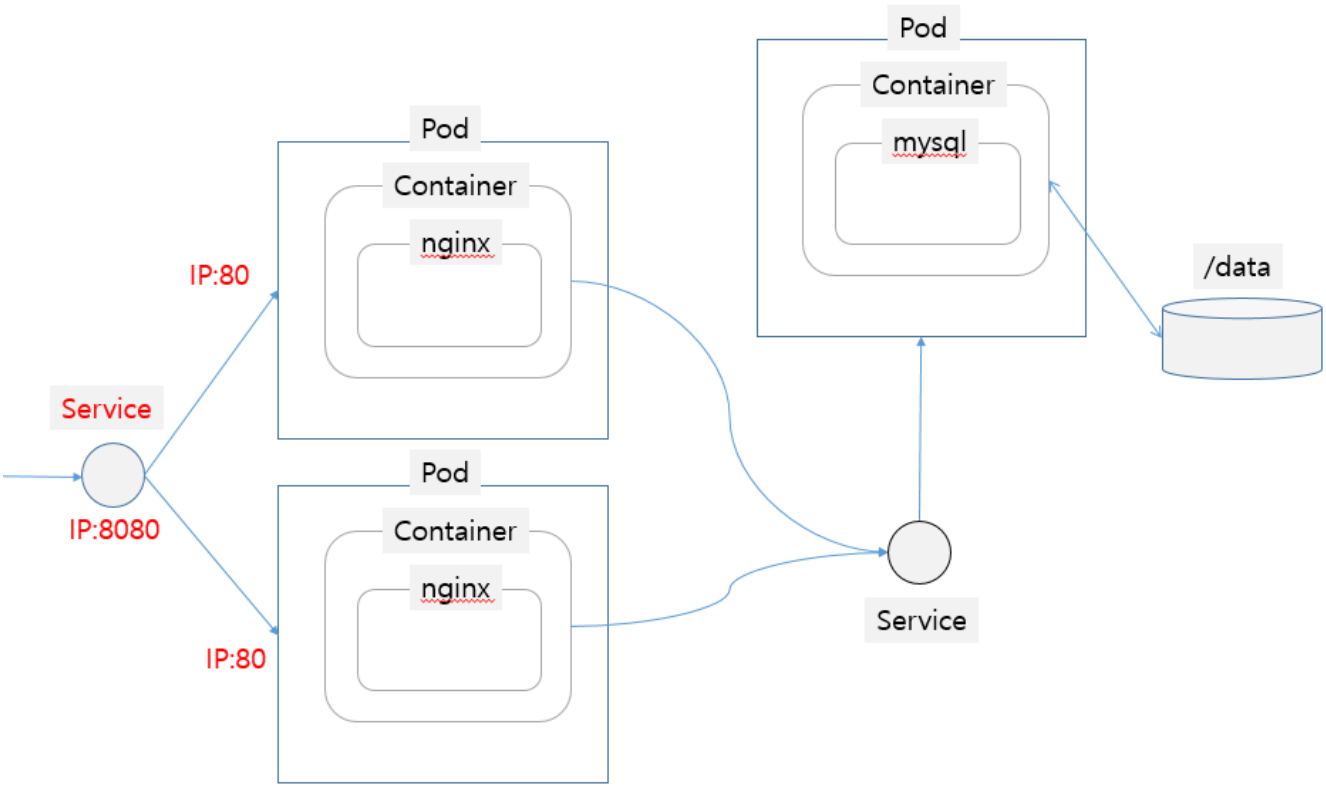
Volume은 제공되는 디스크 Type에 따라 여러가지 종류의 Volume Type을 지원합니다.

Temp	Local	Network
emptyDir	hostPath	NFS, AWS EBS..

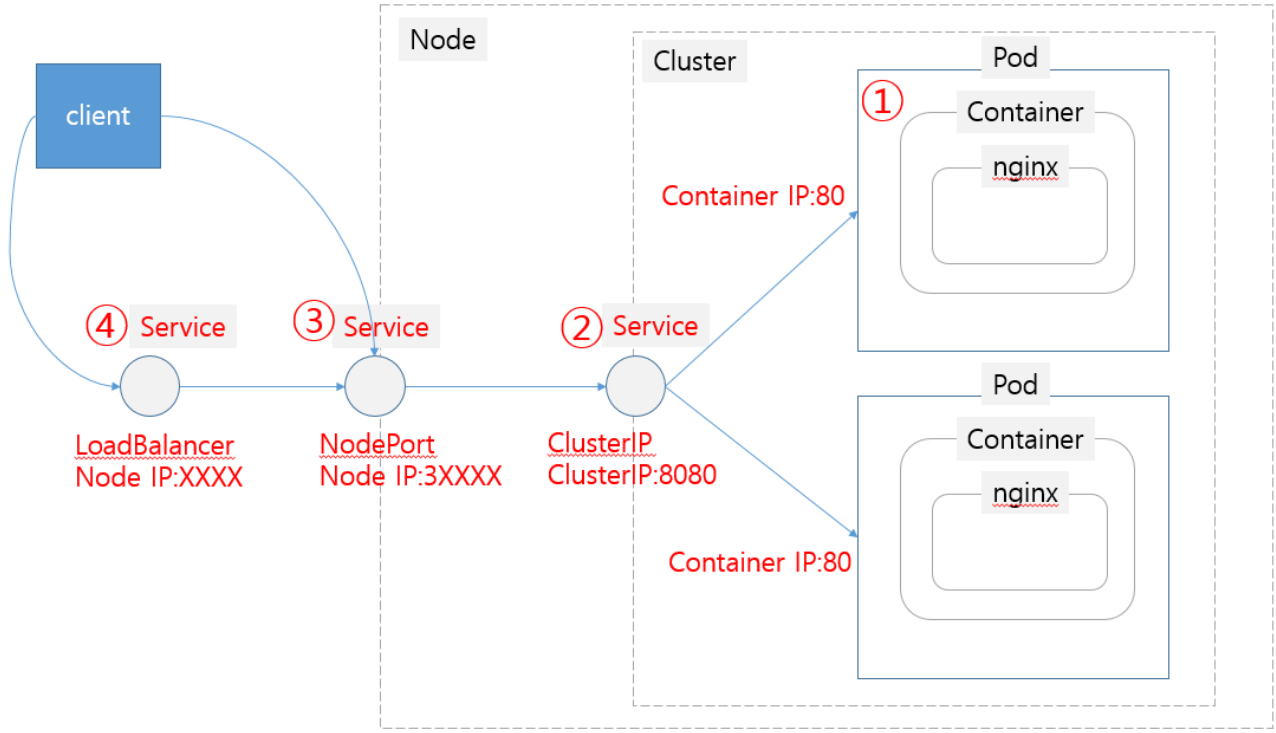
- emptyDir: 동일한 Pod내에서 사용가능한 임시 Volume으로 Pod가 삭제되면 Volume의 데이터도 같이 삭제됩니다.
- hostPath: 특정 노드에 mount된 파일시스템을 Volume으로 사용합니다. 본 교육에서는 로컬 VM환경이므로 HostPath Type으로 실습하는 예제입니다.
- Network : 네트워크 또는 클라우드 상에 존재하는 storage를 Volume으로 사용합니다.

Service

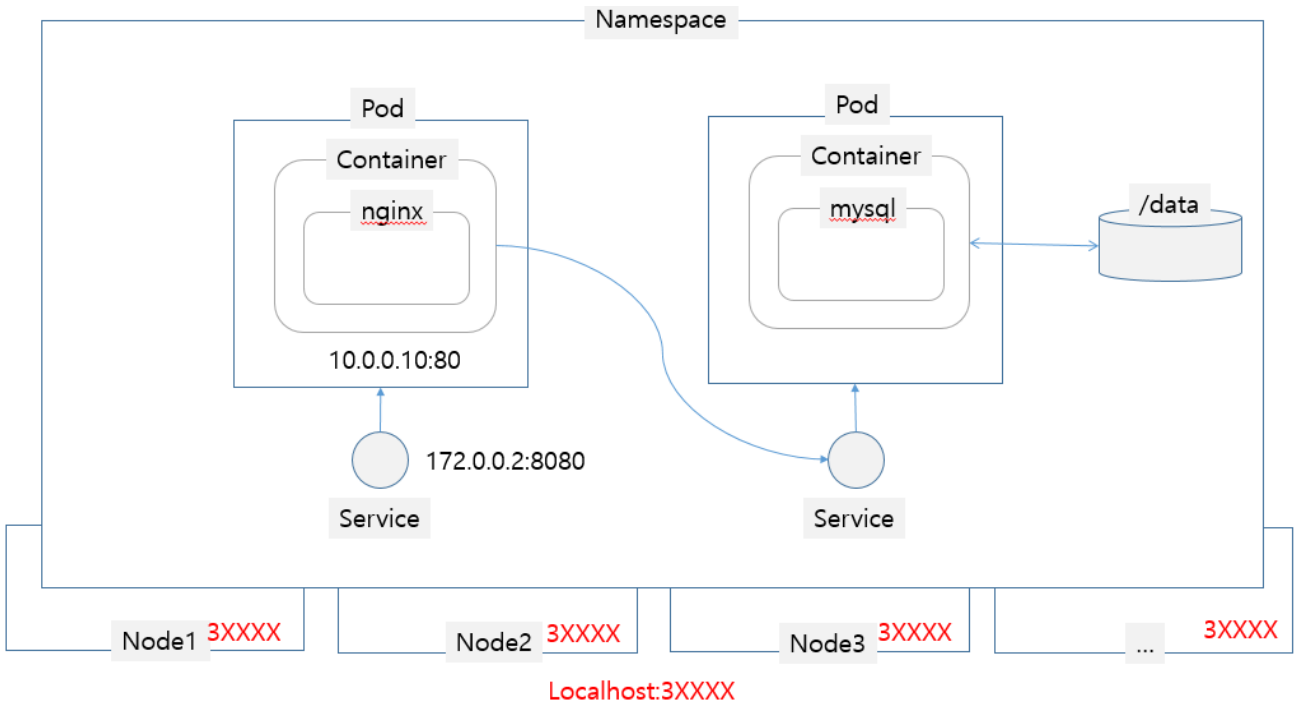
Pod는 ReplicaSet에 의해 동적으로 생성되고 소멸되기를 반복하기 때문에 각 Pod들은 고유의 IP주소를 가지고 있지만, 해당 IP는 동적으로 생성/소멸되는 이유로 안정적인 서비스를 할 수 없습니다. 따라서 k8s 클러스터내에서 포드와 포드간의 통신이 필요하다면(ex. Font-ent, Back-end 간 통신) Service Object를 생성하여 통신해야 합니다.



1. Service는 연동하는 대상에 따라 Service Type을 지정하여 원하는 서비스의 종류를 정의할 수 있습니다.
    - CluserIP
    - NodePort
    - LoadBalancer
2. Service Object는 Label Selector를 통하여 해당 Service의 Pod들을 타겟팅하며, 해당 pod들의 로드밸런싱을 지원합니다.
3. Service YAML에 기술된 name은 k8s 클러스터에서 해당 서비스의 Hostname을 뜻하며, 이것을 바탕으로 k8s 클러스터내에 포함된 DNS 서버에 도메인을 생성합니다.

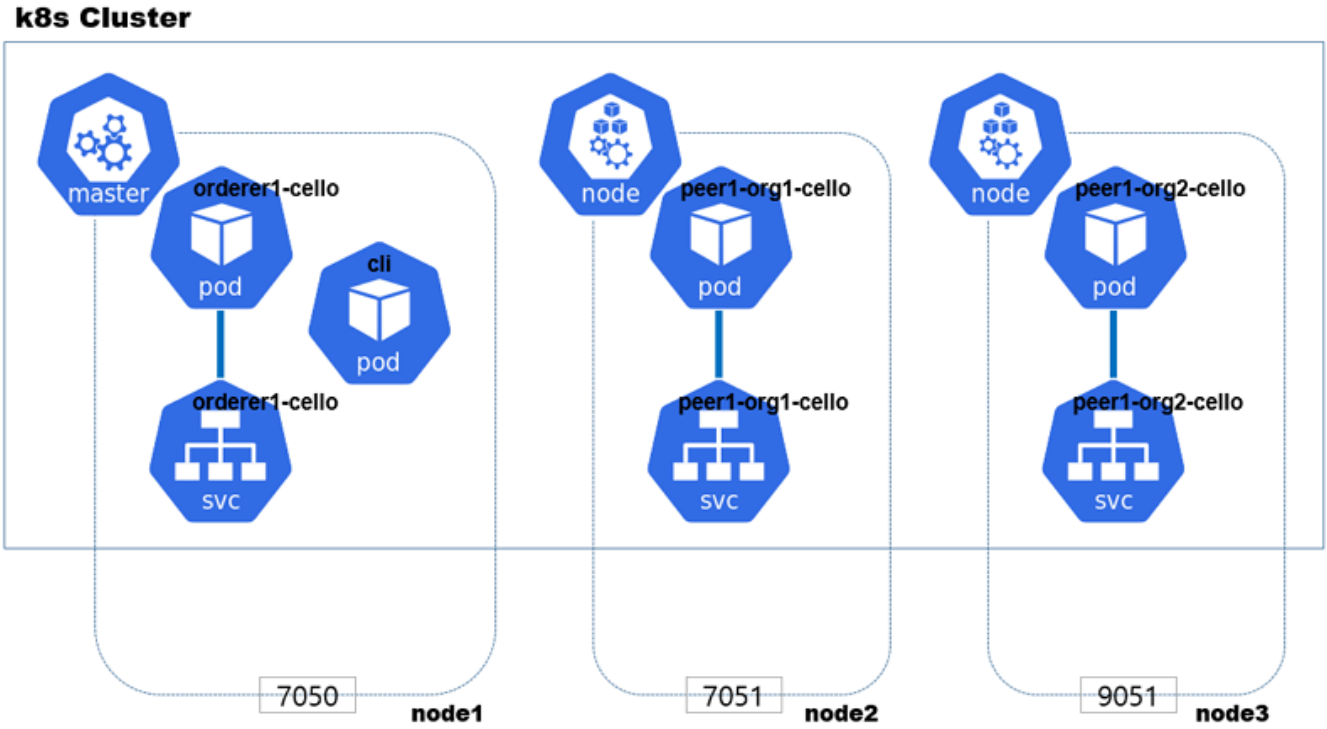


- CluserIP : k8s의 디폴트 설정으로 외부에서 접근가능한 IP를 할당받지 않기 때문에 클러스터내에서만 해당서비스를 노출할 때 사용되는 Type입니다.
- NodePort : 해당 서비스를 외부로 노출시키고자 할 때 사용되는 Service Type으로 외부에 Node IP와 Port를 노출시카는 것으로 아래 그림과 같이 k8s 클러스터에 있는 모든 노드들에 대한 NodePort를 노출합니다.



- LoadBalancer : 클라우드상에 존재하는 LoadBalancer에 연결하고자 할 때 사용되는 Service Type으로 LoadBalancer의 외부 External IP를 통해 접근이 가능합니다.

k8s 클러스에는 자체 DNS서버를 가지고 있어 클러스터 내부에서만 사용가능한 DNS를 설정해서 사용할 수 있습니다. 이것은 k8s 클러스터내에서 통신할때 IP기반이 아닌 도메인을 통해 연결할 수 있음을 뜻하며 아래 그림과 같이 Pod에서 다른 Pod의 서비스를 연결할때 사용됩니다.



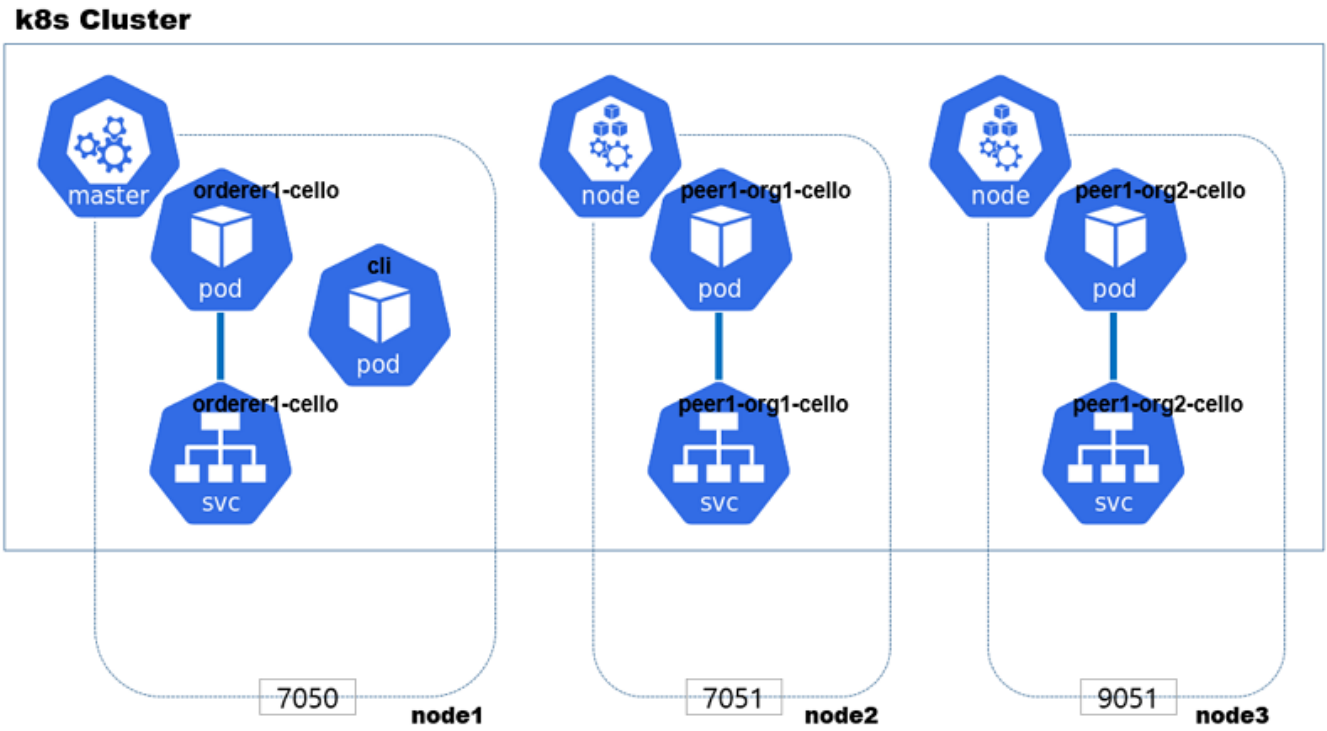
k8s 클러스터에 참여하는 모든 노드에서 시스템 패키지 업데이트를 수행합니다.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install linux-image-extra-virtual
$ sudo reboot
```

k8s 클러스터에 참여하는 모든 노드에 k8s 클러스터를 관리할 사용자를 추가합니다.

```
$ sudo useradd -s /bin/bash -m k8s-admin
$ sudo passwd k8s-admin
$ sudo usermod -aG sudo k8s-admin
```

2. Hyperledger Fabric 네트워크 구성



Orderer, Peer를 포함하는 Pod은 Deployment형태로 지정된 노드에 배포되도록 구성하였습니다.

지정된 노드에 배포하는 이유는 NAS,SAN등과 같은 Shared Disk가 없는 환경에서 Pod이 종료되었을 때 지정된 노드에서 구동되게 하기 위함입니다.

nodeSelector 항목을 사용해서 배포될 노드를 지정하게 되는데, 각 노드의 레이블을 아래와 같이 설정합니다.

```
$ kubectl label nodes node1 role=master
$ kubectl label nodes node2 role=worker1
$ kubectl label nodes node3 role=worker2
```

설정 파일

- **Deployment - orderer1.cello.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orderer1.cello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: orderer1.cello
  template:
    metadata:
      labels:
        app: orderer1.cello
    spec:
      nodeSelector:
        role: master
      containers:
        - name: orderer1-cello
          image: "hyperledger/fabric-orderer:1.4.0"
```

```
          ports:
            - name: port
              containerPort: 7050
          workingDir: /opt/gopath/src/github.com/hyperledger/fabric
          command: ["orderer"]
          env:
            - name: FABRIC_LOGGING_SPEC
              value: "DEBUG"
            - name: ORDERER_GENERAL_LISTENADDRESS
              value: "0.0.0.0"
            - name: ORDERER_GENERAL_GENESISMETHOD
              value: "file"
            - name: ORDERER_GENERAL_GENESISFILE
              value: "/var/hyperledger/orderer/orderer.genesis.block"
            - name: ORDERER_GENERAL_LOCALMSPID
              value: "Orderer1MSP"
            - name: ORDERER_GENERAL_LOCALMSPDIR
              value: "/var/hyperledger/orderer/msp"
            - name: ORDERER_GENERAL_TLS_ENABLED
              value: "false"
            - name: ORDERER_GENERAL_TLS_PRIVATEKEY
              value: "/var/hyperledger/orderer/tls/server.key"
            - name: ORDERER_GENERAL_TLS_CERTIFICATE
              value: "/var/hyperledger/orderer/tls/server.crt"
            - name: ORDERER_GENERAL_TLS_ROOTCAS
              value: "[/var/hyperledger/orderer/tls/ca.crt]"
          volumeMounts:
            - name: host-genesis
              mountPath: /var/hyperledger/orderer/orderer.genesis.block
            - name: host-msp
              mountPath: /var/hyperledger/orderer/msp
            - name: host-tls
              mountPath: /var/hyperledger/orderer/tls

      volumes:
        - name: host-genesis
          hostPath:
            path: /root/k8s/network/fabric/channel-artifacts/genesis.block
        - name: host-msp
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/ordererOrganizations/orderer1.cello/msp
        - name: host-tls
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/ordererOrganizations/orderer1.cello/tls
```

- **Deployment - peer1.org1.cello.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: peer1.org1.cello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: peer1.org1.cello
  template:
    metadata:
```

```
  labels:
    app: peer1.org1.cello
spec:
  nodeSelector:
    role: worker1
  containers:
  - name: peer1-org1-cello
    image: "hyperledger/fabric-peer:1.4.0"
    ports:
      - name: port
        containerPort: 7051
      - name: chaincode
        containerPort: 7052
      - name: operations
        containerPort: 7043
    workingDir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: ["peer"]
    args: ["node","start"]
    env:
      - name: CORE_VM_ENDPOINT
        value: "unix:///host/var/run/docker.sock"
      - name: CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE
        value: "bridge"
      - name: FABRIC_LOGGING_SPEC
        value: "INFO"
      - name: CORE_PEER_GOSSIP_USELEADERELECTION
        value: "true"
      - name: CORE_PEER_GOSSIP_ORGLEADER
        value: "false"
      - name: CORE_PEER_PROFILE_ENABLED
        value: "true"
      - name: CORE_PEER_TLS_ENABLED
        value: "false"
      - name: CORE_PEER_TLS_CERT_FILE
        value: "/etc/hyperledger/fabric/tls/server.crt"
      - name: CORE_PEER_TLS_KEY_FILE
        value: "/etc/hyperledger/fabric/tls/server.key"
      - name: CORE_PEER_TLS_ROOTCERT_FILE
        value: "/etc/hyperledger/fabric/tls/ca.crt"
      - name: CORE_PEER_ID
        value: "peer1-org1-cello"
      - name: CORE_PEER_ADDRESS
        value: "0.0.0.0:7051"
      - name: CORE_PEER_ADDRESSAUTODETECT
        value: "false"
      - name: CORE_PEER_LISTENADDRESS
        value: "0.0.0.0:7051"
      - name: CORE_PEER_GOSSIP_BOOTSTRAP
        value: "0.0.0.0:7051"
      - name: CORE_PEER_GOSSIP_EXTERNALENDPOINT
        value: "0.0.0.0:7051"
      - name: CORE_PEER_CHAINCODELISTENADDRESS
        value: "0.0.0.0:7052"
      - name: CORE_PEER_LOCALMSPID
        value: "Org1MSP"
      - name: CORE_OPERATIONS_LISTENADDRESS
        value: "0.0.0.0:7043"
      - name: CORE_METRICS_PROVIDER
        value: "prometheus"
    volumeMounts:
```

```
          - name: host-run
            mountPath: /host/var/run
          #- name: host-ledger
          #  mountPath: /var/hyperledger/production
          - name: host-msp
            mountPath: /etc/hyperledger/fabric/msp
          - name: host-tls
            mountPath: /etc/hyperledger/fabric/tls
      volumes:
        - name: host-run
          hostPath:
            path: /var/run
        - name: host-ledger
          hostPath:
            path: /root/k8s/network/fabric/ledger/peer1.org1.cello
        - name: host-msp
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/peerOrganizations/org1.cello/peers/peer1.org1.cello/msp
        - name: host-tls
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/peerOrganizations/org1.cello/peers/peer1.org1.cello/tls
```

- **Deployment - peer1.org2.cello.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: peer1.org2.cello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: peer1.org2.cello
  template:
    metadata:
      labels:
        app: peer1.org2.cello
    spec:
      nodeSelector:
        role: worker2
      containers:
      - name: peer1-org2-cello
        image: "hyperledger/fabric-peer:1.4.0"
        ports:
          - name: port
            containerPort: 9051
          - name: chaincode
            containerPort: 9052
          - name: operations
            containerPort: 9043
        workingDir: /opt/gopath/src/github.com/hyperledger/fabric/peer
        command: ["peer"]
        args: ["node","start"]
        env:
          - name: CORE_VM_ENDPOINT
            value: "unix:///host/var/run/docker.sock"
          - name: CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE
            value: "bridge"
```

```
          - name: FABRIC_LOGGING_SPEC
            value: "INFO"
          - name: CORE_PEER_GOSSIP_USELEADERELECTION
            value: "true"
          - name: CORE_PEER_GOSSIP_ORGLEADER
            value: "false"
          - name: CORE_PEER_PROFILE_ENABLED
            value: "true"
          - name: CORE_PEER_TLS_ENABLED
            value: "false"
          - name: CORE_PEER_TLS_CERT_FILE
            value: "/etc/hyperledger/fabric/tls/server.crt"
          - name: CORE_PEER_TLS_KEY_FILE
            value: "/etc/hyperledger/fabric/tls/server.key"
          - name: CORE_PEER_TLS_ROOTCERT_FILE
            value: "/etc/hyperledger/fabric/tls/ca.crt"
          - name: CORE_PEER_ID
            value: "peer1-org2-cello"
          - name: CORE_PEER_ADDRESS
            value: "0.0.0.0:9051"
          - name: CORE_PEER_LISTENADDRESS
            value: "0.0.0.0:9051"
          - name: CORE_PEER_GOSSIP_BOOTSTRAP
            value: "0.0.0.0:9051"
          - name: CORE_PEER_GOSSIP_EXTERNALENDPOINT
            value: "0.0.0.0:9051"
          - name: CORE_PEER_CHAINCODELISTENADDRESS
            value: "0.0.0.0:9052"
          - name: CORE_PEER_LOCALMSPID
            value: "Org2MSP"
          - name: CORE_OPERATIONS_LISTENADDRESS
            value: "0.0.0.0:9043"
          - name: CORE_METRICS_PROVIDER
            value: "prometheus"
        volumeMounts:
          - name: host-run
            mountPath: /host/var/run
          - name: host-msp
            mountPath: /etc/hyperledger/fabric/msp
          - name: host-tls
            mountPath: /etc/hyperledger/fabric/tls
      volumes:
        - name: host-run
          hostPath:
            path: /var/run
        - name: host-msp
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/peerOrganizations/org2.cello/peers/peer1.org2.cello/msp
        - name: host-tls
          hostPath:
            path: /root/k8s/network/fabric/crypto-
config/peerOrganizations/org2.cello/peers/peer1.org2.cello/tls
```

- **Deployment - cli.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cli
```

```yaml
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cli
  template:
    metadata:
      labels:
        app: cli
    spec:
      nodeSelector:
        role: "master"
      containers:
      - name: cli
        image: "hyperledger/fabric-tools:1.4.0"
        workingDir: /opt/gopath/src/cello
        command: ["peer"]
        args: ["node","start"]
        env:
          - name: FABRIC_LOGGING_SPEC
            value: "INFO"
          - name: CORE_PEER_TLS_ENABLED
            value: "false"
          - name: CORE_PEER_TLS_CERT_FILE
            value: "/opt/gopath/src/cello/crypto/peer1-org1-cello/tls/server.crt"
          - name: CORE_PEER_TLS_KEY_FILE
            value: "/opt/gopath/src/cello/crypto/peer1-org1-cello/tls/server.key"
          - name: CORE_PEER_TLS_ROOTCERT_FILE
            value: "/opt/gopath/src/cello/crypto/peer1-org1-cello/tls/ca.crt"
          - name: CORE_ORDERER_CA
            value: "/opt/gopath/src/cello/crypto/orderer1-cello/tls/ca.crt"
          - name: GOPATH
            value: "/opt/gopath"
          - name: CORE_VM_ENDPOINT
            value: "unix:///host/var/run/docker.sock"
        volumeMounts:
          - name: host-run
            mountPath: /host/var/run
          - name: host-chaincode
            mountPath: /opt/gopath/src/cello/chaincode
          - name: host-crypto-config
            mountPath: /opt/gopath/src/cello/crypto
          - name: host-channel-artifacts
            mountPath: /opt/gopath/src/cello/channel-artifacts
          - name: host-work
            mountPath: /opt/gopath/src/cello/work
      volumes:
        - name: host-run
          hostPath:
            path: /var/run
        - name: host-chaincode
          hostPath:
            path: /root/k8s/network/fabric/chaincode
        - name: host-crypto-config
          hostPath:
            path: /root/k8s/network/fabric/crypto-config
        - name: host-channel-artifacts
          hostPath:
            path: /root/k8s/network/fabric/channel-artifacts
        - name: host-work
```

```
        hostPath:
          path: /root/k8s/network/fabric/work
```

k8s 클러스터 밖에서 Orderer, Peer에 접속하기 위해 NodeType 형식으로 Service를 배포합니다.

- **Service - orderer1.cello.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: orderer1-cello  # 점(.)을 사용할 수 없습니다.
spec:
 selector:
   app: orderer1.cello
 type: NodePort
 ports:
   - name: external-listen-endpoint
     protocol: TCP
     port: 7050
     targetPort: 7050
     nodePort: 7050
```

- **Service - peer1.org1.cello.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: peer1-org1-cello  # 점(.)을 사용할 수 없습니다.
spec:
 selector:
   app: peer1.org1.cello
 type: NodePort
 ports:
   - name: external-listen-endpoint
     protocol: TCP
     port: 7051
     targetPort: 7051
     nodePort: 7051

   - name: chaincode-listen
     protocol: TCP
     port: 7052
     targetPort: 7052
     nodePort: 7052

   - name: operation-listen
     protocol: TCP
     port: 7043
     targetPort: 7043
     nodePort: 7043
```

- **Service - peer1.org2.cello.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: peer1-org2-cello  # 점(.)을 사용할 수 없습니다.
spec:
 selector:
```

```
    app: peer1.org2.cello
  type: NodePort
  ports:
    - name: external-listen-endpoint
      protocol: TCP
      port: 9051
      targetPort: 9051
      nodePort: 9051

    - name: chaincode-listen
      protocol: TCP
      port: 9052
      targetPort: 9052

    - name: operation-listen
      protocol: TCP
      port: 9043
      targetPort: 9043
      nodePort: 9043
```

배포 및 구동

관리의 편의성을 위해 Deploymet, Service YAML 파일을 아래와 같은 디렉토리 구조로 저장합니다. Deployment와 Service 를 하나의 YAML 파일로 구성할 수도 있습니다.(구분자로 **---** 를 사용합니다.)

```
$ tree
.
├── deployments
|   ├── cli.yaml
|   ├── orderer1.cello.yaml
|   ├── peer1.org1.cello.yaml
|   └── peer1.org2.cello.yaml
└── services
    ├── orderer1.cello.yaml
    ├── peer1.org1.cello.yaml
    └── peer1.org2.cello.yaml
```

각 디렉토리에서 아래와 같이 k8s 클러스터에 배포 및 구동합니다.

```
$ cd deployments

$ kubectl apply -f .
deployment.apps/cli created
deployment.apps/orderer1.cello created
deployment.apps/peer1.org1.cello created
deployment.apps/peer1.org2.cello created

$ cd ../services
$ kubectl apply -f .
service/orderer1-cello created
service/peer1-org1-cello created
service/peer1-org2-cello created

$ kubectl get po # Pod 확인
NAME                                READY   STATUS    RESTARTS   AGE
busybox                             1/1     Running   0          52m
cli-fcb9c8f85-xtblx                 1/1     Running   0          48s
```

```
orderer1.cello-5466c95976-bpt8s     1/1     Running   0          48s
peer1.org1.cello-7456cf97bf-2zspz   1/1     Running   0          48s
peer1.org2.cello-b76d69766-nd587    1/1     Running   0          48s

$ kubectl get svc # Service 확인
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
            AGE
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP
            73s
orderer1-cello     NodePort    10.101.33.82    <none>        7050:7050/TCP
            69s
peer1-org1-cello   NodePort    10.110.34.110   <none>
 7051:7051/TCP,7052:7052/TCP,7043:7043/TCP   69s
peer1-org2-cello   NodePort    10.103.210.49   <none>
 9051:9051/TCP,9052:9072/TCP,9043:9043/TCP   69s
```