클라우드기반 전자거래 및 인증을 위한 블록체인 오픈 플랫 구축

운영자지침서(블록체인)

KM\_L\_T\_01

Ver. 1.0

관리부서 : 사업/품질담당

개 정 이 력

| 버전 | 작성일 | 변경내용 | 작성자 | 승인자 |
| --- | --- | --- | --- | --- |
| 1.0 | 2018-12-07 | 최초 작성 | 정지연 | 박치성 |
| 1.1 | 2018-01-08 | 체인코드 종료 방법 추가 | 정지연 | 박치성 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목 차

[1. 개요 1](#_Toc534721289)

[2. 시스템 구성 정보 1](#_Toc534721290)

[2.1 개발 서버 1](#_Toc534721291)

[2.1.1 개발 노드 #1 1](#_Toc534721292)

[2.1.2 개발 노드 #2 1](#_Toc534721293)

[2.1.3 개발 노드 #3 2](#_Toc534721294)

[2.1.4 개발 관리 서버 2](#_Toc534721295)

[2.2 운영서버 3](#_Toc534721296)

[2.2.1 운영 노드 #1 3](#_Toc534721297)

[2.2.2 운영 노드 #2 3](#_Toc534721298)

[2.2.3 운영 노드 #3 4](#_Toc534721299)

[2.2.4 운영 노드 #4 4](#_Toc534721300)

[2.2.5 운영 관리 서버 5](#_Toc534721301)

[2.3 DR서버 5](#_Toc534721302)

[2.3.1 DR 노드 #1 5](#_Toc534721303)

[2.3.2 DR 노드 #2 6](#_Toc534721304)

[2.3.3 DR 노드 #3 6](#_Toc534721305)

[2.3.4 DR 관리 서버 7](#_Toc534721306)

[2.4 시스템 자원 모니터링 7](#_Toc534721307)

[2.4.1 nmon 7](#_Toc534721308)

[2.4.2 docker stats 8](#_Toc534721309)

[3. 블록체인 네트워크 환경 구성 정보 9](#_Toc534721310)

[3.1 개발서버 9](#_Toc534721311)

[3.1.1 개발 노드 #1 9](#_Toc534721312)

[3.1.2 개발 노드 #2 9](#_Toc534721313)

[3.1.3 개발 노드 #3 10](#_Toc534721314)

[3.2 운영서버 10](#_Toc534721315)

[3.2.1 운영 노드 #1 10](#_Toc534721316)

[3.2.2 운영 노드 #2 10](#_Toc534721317)

[3.2.3 운영 노드 #3 10](#_Toc534721318)

[3.2.4 운영 노드 #4 11](#_Toc534721319)

[3.3 DR서버 11](#_Toc534721320)

[3.3.1 DR 노드 #1 11](#_Toc534721321)

[3.3.2 DR 노드 #2 11](#_Toc534721322)

[3.3.3 DR 노드 #3 12](#_Toc534721323)

[4. Node 관리 12](#_Toc534721324)

[4.1 Organization 환경 설정 12](#_Toc534721325)

[4.1.1 Orderer Org 12](#_Toc534721326)

[4.1.2 Peer Org 12](#_Toc534721327)

[4.2 Orderer 환경 설정 13](#_Toc534721328)

[4.2.1 개발서버 Orderer 13](#_Toc534721329)

[4.2.1.1 개발 orderer 상세 정보 13](#_Toc534721330)

[4.2.2 운영서버 Orderer 14](#_Toc534721331)

[4.2.2.1 운영 orderer0 상세 정보 15](#_Toc534721332)

[4.2.2.2 운영 orderer1 상세 정보 16](#_Toc534721333)

[4.3 Channel 환경 설정 17](#_Toc534721334)

[4.4 Node 기동/정지 17](#_Toc534721335)

[4.4.1 Node 기동 17](#_Toc534721336)

[4.4.2 Node 정지 18](#_Toc534721337)

[4.5 Org 및 Node 추가 18](#_Toc534721338)

[4.5.1 Organization 추가 18](#_Toc534721339)

[4.5.2 Peer 추가 19](#_Toc534721340)

[5. 인증서 관리 20](#_Toc534721341)

[5.1 키, 인증서 생성 20](#_Toc534721342)

[5.2 키, 인증서별 역할 20](#_Toc534721343)

[5.3 HSM 키 주입 22](#_Toc534721344)

[5.3.1 HSM 데몬 포트 활성화 22](#_Toc534721345)

[5.3.2 JRE Security Provider 설정 22](#_Toc534721346)

[5.3.3 슬롯 생성 23](#_Toc534721347)

[5.3.4 키 주입 23](#_Toc534721348)

[6. 체인코드 배포 관리 24](#_Toc534721349)

[6.1 형상관리(SVN) 24](#_Toc534721350)

[6.1.1 기본 정보 25](#_Toc534721351)

[6.1.2 Source Merge 25](#_Toc534721352)

[6.2 빌드 28](#_Toc534721353)

[6.2.1 개발서버 빌드 29](#_Toc534721354)

[6.2.2 운영서버 빌드 29](#_Toc534721355)

[6.3 체인코드 배포 29](#_Toc534721356)

[6.3.1 체인코드 업로드 29](#_Toc534721357)

[6.3.2 체인코드 Install 30](#_Toc534721358)

[6.3.3 체인코드 Instantiate/Upgrade 30](#_Toc534721359)

[6.4 체인코드 종료 30](#_Toc534721360)

[7. Fabric-weaver 관리 30](#_Toc534721361)

[7.1 Fabric-weaver 기동/중지 30](#_Toc534721362)

[7.1.1 Fabric-weaver 기동 30](#_Toc534721363)

[7.1.2 Fabric-weaver 중지 31](#_Toc534721364)

[7.2 Fabric-weaver 계정 및 권한 31](#_Toc534721365)

[7.3 Fabric-weaver 접속 31](#_Toc534721366)

# 개요

본 문서는 한국 조폐공사 “클라우드기반 전자거래 및 인증을 위한 블록체인 오픈 플랫폼구축” 시스템의 운영을 위한 지침을 담은 문서이다.

# 시스템 구성 정보

## 개발 서버

### 개발 노드 #1

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.20.20 |  |
| Host | blocco01d |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-network |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer0.org1.komsco.com  couchdb0  cli |  |

### 개발 노드 #2

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.20.21 |  |
| Host | blocco02d |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-network |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer1.org1.komsco.com  couchdb1  cli |  |

### 개발 노드 #3

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.20.22 |  |
| Host | blocco03d |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-network |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | orderer.komsco.com |  |

### 개발 관리 서버

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.20.25 |  |
| Host | intmg01d |  |
| OS 계정 | weaver : weaver 기동용 계정  mysql : 마리아DB 기동용 계정 |  |
| Fabric weaver Home Directory | /weaver |  |
| MariaDB Data Directory | /var/lib/mysql/fabricweaver |  |
| S/W | nodejs  mariadb |  |

## 운영서버

### 운영 노드 #1

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.40.20  Public IP: 61.42.159.108 |  |
| Host | blocco01p |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer0.org1.komsco.com  couchdb0  cli |  |

### 운영 노드 #2

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.40.21 |  |
| Host | blocco02p |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer1.org1.komsco.com  couchdb1  cli |  |

### 운영 노드 #3

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.40.22 |  |
| Host | blocco03p |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | orderer0.komsco.com  kafka0  zookeeper0 |  |

### 운영 노드 #4

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.40.23 |  |
| Host | blocco04p |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | orderer1.komsco.com  kafka1  zookeeper1 |  |

### 운영 관리 서버

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.40.24  Public IP: 61.42.159.109 |  |
| Host | blcmg01p |  |
| OS 계정 | weaver : weaver 기동용 계정  mysql : 마리아DB 기동용 계정 |  |
| Fabric weaver Home Directory | /weaver |  |
| MariaDB Data Directory | /var/lib/mysql/fabricweaver |  |
| S/W | nodejs  mariadb |  |

## DR서버

### DR 노드 #1

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.60.20  Public IP: TBD |  |
| Host | blocco01r |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer2.org1.komsco.com  couchdb2  cli |  |

### DR 노드 #2

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.60.21 |  |
| Host | blocco02r |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | peer3.org1.komsco.com  couchdb3  cli |  |

### DR 노드 #3

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.60.22 |  |
| Host | blocco03r |  |
| OS 계정 | blochain : 블록체인 기동용 계정 |  |
| Home Directory | /blochain |  |
| Blockchain Home Directory | /blochain/komsco-config |  |
| Blockchain Data Directory | /data |  |
| S/W | docker-ce |  |
| Docker Container | orderer2.komsco.com  kafka2  zookeeper2 |  |

### DR 관리 서버

| 항목 | 내용 | 비고 |
| --- | --- | --- |
| IP | 10.10.60.23  Public IP: TBD |  |
| Host | blcmg01r |  |
| OS 계정 | weaver : weaver 기동용 계정  mysql : 마리아DB 기동용 계정 |  |
| Fabric weaver Home Directory | /weaver |  |
| MariaDB Data Directory | /var/lib/mysql/fabricweaver |  |
| S/W | nodejs  mariadb |  |

## 시스템 자원 모니터링

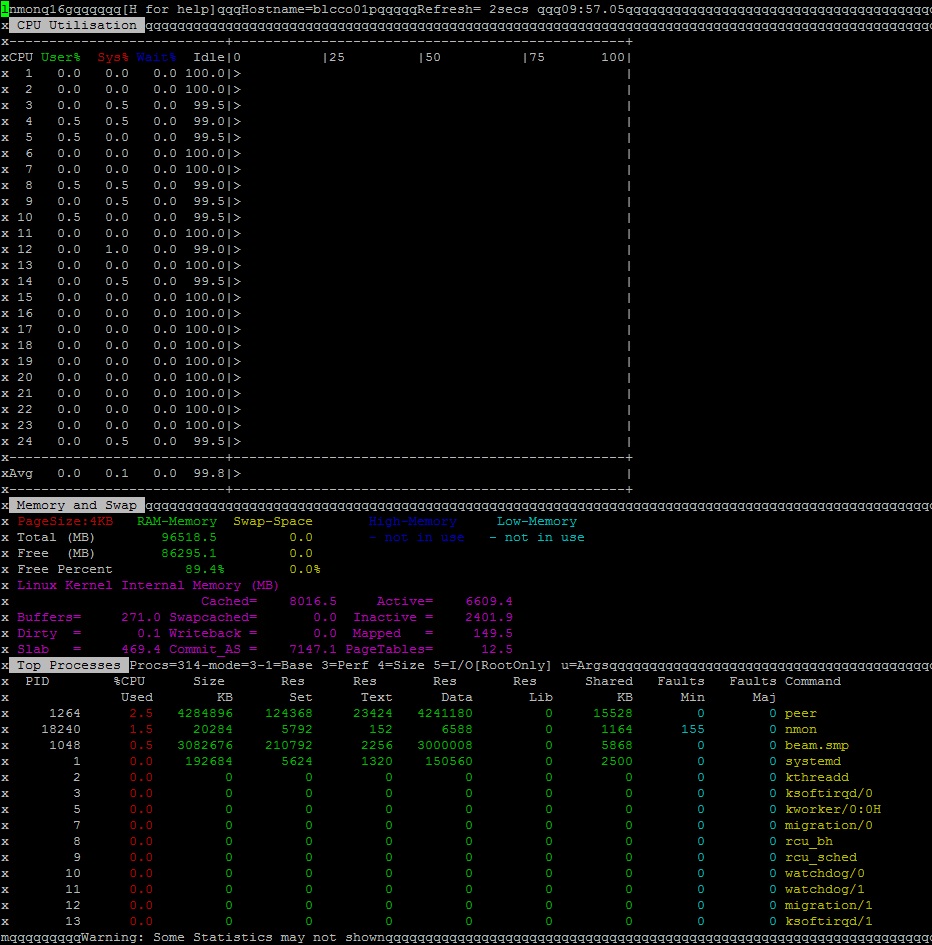
서버의 자원을 모니터링하기 위해선 OS에서 제공하는 모니터링 툴을 활용한다.

### nmon

nmon은 CPU, 메모리, TOP Process등을 확인 할 수 있는 툴로써 서버에 ssh 로 접속한 후 nmon 을 실행해 확인할 수 있다. 실행 후 단축키 입력을 통해 보고 싶은 자원에 대해 모니터링이 가능하며 단축키 정보는 아래와 같다

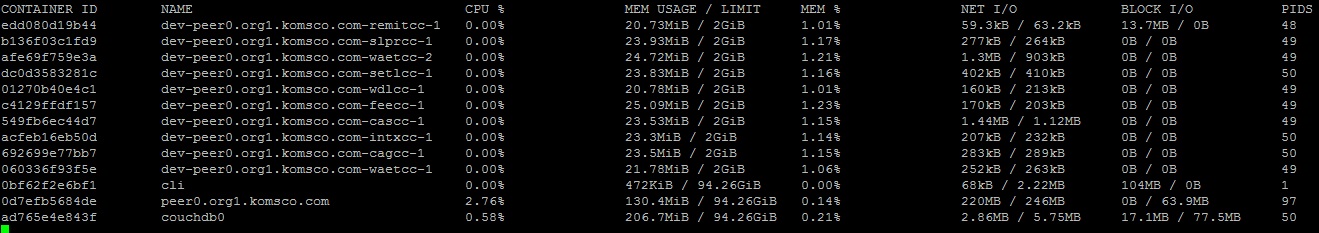
|  |  |  |  |
| --- | --- | --- | --- |
| 단축키 | 설명 | 단축키 | 설명 |
| h | 도움말 | r | Os 와 Process resource |
| c | CPU 사용율 | l | Longterm CPU 사용률 |
| m | 메모리 & 스왑 | V | Virtual memory |
| n | 네트워크 | o | Disks Busy Map |
| d | 디스크 I/O | j | Filesystem 사용률 |
| k | Kernel 상태 | u | Top Process with command line |
| t | Top Process 정보 |  |  |
| g | 디스크 그룹 |  |  |

nmon 실행 후 위의 단축키를 입력하면 아래와 같은 모니터링 화면을 볼 수 있다.



### docker stats

nmon이 서버단위의 자원 모니터링이라면 docker stats 은 docker container 단위의 자원모니터링이다. 블록체인 노드 서버에서 docker stats 를 실행하면 아래처럼 container 별 cpu, 메모리, network i/o 등을 확인할 수 있다.



# 블록체인 네트워크 환경 구성 정보

Docker 컨테이너 상세 정보 및 버전은 아래와 같다. Hyperledger Fabric 은 현재 1.2.1 버전으로 적용되어 있다. 향후 Fabric 버전을 업그레이드 하기 위해서는 docker hub 에서 docker image 를 받아오고 새로운 버전을 최신버전으로 태그하여 컨테이너를 재기동시키면 적용 된다.

|  |
| --- |
| docker pull hyperledger/[fabric 이미지명]:[버전]  docker tag hyperledger/[fabric 이미지명]:[버전] hyperledger/[fabric 이미지명]  /blochain/komsco-config/start.sh |

## 개발서버

### 개발 노드 #1

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer0.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer |
| couchdb0 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### 개발 노드 #2

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer1.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer |
| couchdb1 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### 개발 노드 #3

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| orderer.komsco.com | hyperledger/fabric-orderer | 1.2.1 | 7050 | /data/orderer |

## 운영서버

### 운영 노드 #1

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer0.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer0 |
| couchdb0 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb0 |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### 운영 노드 #2

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer1.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer1 |
| couchdb1 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb1 |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### 운영 노드 #3

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| orderer0.komsco.com | hyperledger/fabric-orderer | 1.2.1 | 7050 | /data/orderer0 |
| kafka0 | hyperledger/fabric-kafka | 0.4.10 | 9092 | /data/kafka0 |
| zookeeper0 | hyperledger/fabric-zookeeper | 0.4.10 | 2181  2888  3888 | /data/zookeeper0 |

### 운영 노드 #4

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| orderer1.komsco.com | hyperledger/fabric-orderer | 1.2.1 | 7050 | /data/orderer1 |
| kafka1 | hyperledger/fabric-kafka | 0.4.10 | 9092 | /data/kafka1 |
| zookeeper1 | hyperledger/fabric-zookeeper | 0.4.10 | 2181  2888  3888 | /data/zookeeper1 |

## DR서버

### DR 노드 #1

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer2.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer2 |
| couchdb2 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb2 |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### DR 노드 #2

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| peer3.org1.komsco.com | hyperledger/fabric-peer | 1.2.1 | 7051  7052  7053 | /data/peer3 |
| couchdb3 | hyperledger/fabric-couchdb | 0.4.10 | 5984 | /data/couchdb3 |
| cli | hyperledger/fabric-tools | 1.2.1 |  |  |

### DR 노드 #3

| 컨테이너명 | Image | 버전 | 포트 | 볼륨 |
| --- | --- | --- | --- | --- |
| orderer2.komsco.com | hyperledger/fabric-orderer | 1.2.1 | 7050 | /data/orderer2 |
| kafka2 | hyperledger/fabric-kafka | 0.4.10 | 9092 | /data/kafka2 |
| zookeeper2 | hyperledger/fabric-zookeeper | 0.4.10 | 2181  2888  3888 | /data/zookeeper2 |

# Node 관리

## Organization 환경 설정

### Orderer Org

|  |  |  |
| --- | --- | --- |
| **항목** | **내용** | **비고** |
| Organization 이름 | OrdererOrg |  |
| Organization Type | ORDERER ORG |  |
| Domain | komsco.com |  |
| MSP ID | OrdererOrgMSP |  |
| Nodes(운영서버) | orderer0.komsco.com  orderer1.komsco.com  orderer2.komsco.com |  |
| Nodes(개발서버) | orderer.komsco.com |  |

### Peer Org

|  |  |  |
| --- | --- | --- |
| **항목** | **내용** | **비고** |
| Organization 이름 | Org1 |  |
| Organization Type | PEER ORG |  |
| Domain | org1.komsco.com |  |
| MSP ID | Org1MSP |  |
| Nodes(운영서버) | peer0.org1.komsco.com  peer1.org1.komsco.com  peer2.org1.komsco.com  peer3.org1.komsco.com |  |
| Nodes(개발서버) | peer0.org1.komsco.com  peer1.org1.komsco.com |  |

## Orderer 환경 설정

### 개발서버 Orderer

개발 서버의 Orderer는 Solo로 구성되어 있으며 정보는 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **항목** | **내용** | |
| Organization | OrdererOrg | |
| ConfigTx Name | komsco | |
| Profile Name | KomscoOrdererGenesis | |
| Genesis Filename | orderer\_genesis.block | |
| Orderer Nodes | orderer.komsco.com | grpc://10.10.20.22:7050 |

#### 개발 orderer 상세 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **분류** | **항목** | **내용** | **비고** |
| 일반 | Connection URL | grpc://10.10.20.22:7050 |  |
| 컨테니어 정보 | 컨테이너 이름 | orderer.komsco.com |  |
| 컨테이너 이미지 | hyperledger/fabric-orderer |  |
| working\_dir | /opt/gopath/src/github.com/hyperledger/fabric |  |
| command | orderer |  |
| 볼륨 | - './channel-artifacts/komsco/orderer\_genesis.block:/var/hyperledger/orderer/orderer.genesis.block' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer.komsco.com/msp:/var/hyperledger/orderer/msp' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer.komsco.com/tls/:/var/hyperledger/orderer/tls' - '/data/orderer/:/var/hyperledger/' |  |
| 포트 | - '7050:7050' |  |
| 네트워크 | komsco |  |
| 컨테이너 환경설정 | ORDERER\_GENERAL\_LOGLEVEL | INFO |  |
| ORDERER\_GENERAL\_LISTENADDRESS | 0.0.0.0 |  |
| ORDERER\_GENERAL\_LISTENPORT | 7050 |  |
| ORDERER\_GENERAL\_GENESISMETHOD | file |  |
| ORDERER\_GENERAL\_GENESIS\_FILE | /var/hyperledger/orderer/orderer\_genesis.block |  |
| ORDERER\_GENERAL\_LOCALMSPID | OrdererOrgMSP |  |
| ORDERER\_GENERAL\_LOCALMSPDIR | /var/hyperledger/orderer/msp |  |
| ORDERER\_GENERAL\_TLS\_ENABLED | false |  |

### 운영서버 Orderer

운영 서버의 Orderer는 Kafka로 구성되어 있으며 정보는 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **항목** | **내용** | |
| Organization | OrdererOrg | |
| ConfigTx Name | komsco | |
| Profile Name | KomscoOrdererGenesis | |
| Genesis Filename | orderer\_genesis.block | |
| Orderer Nodes | orderer0.komsco.com | grpc://10.10.40.22:7050 |
| orderer1.komsco.com | grpc://10.10.40.23:7050 |
| Kafka Brokers | kafka0 | 10.10.40.22:9092 |
| kafka1 | 10.10.40.23:9092 |
| Zookeeper Nodes | zookeeper0 | 10.10.40.22:2181 |
| zookeeper1 | 10.10.40.23:2181 |

#### 운영 orderer0 상세 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **분류** | **항목** | **내용** | **비고** |
| 일반 | Connection URL | grpc://10.10.40.22:7050 |  |
| 컨테니어 정보 | 컨테이너 이름 | orderer0.komsco.com |  |
| 컨테이너 이미지 | hyperledger/fabric-orderer |  |
| working\_dir | /opt/gopath/src/github.com/hyperledger/fabric |  |
| command | orderer |  |
| 볼륨 | - './channel-artifacts/komsco/orderer\_genesis.block:/var/hyperledger/orderer/orderer.genesis.block' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer0.komsco.com/msp:/var/hyperledger/orderer/msp' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer0.komsco.com/tls/:/var/hyperledger/orderer/tls' - '/data/orderer0/:/var/hyperledger/' |  |
| 포트 | - '7050:7050' |  |
| 네트워크 | komsco |  |
| 컨테이너 환경설정 | ORDERER\_GENERAL\_LOGLEVEL | INFO |  |
| ORDERER\_GENERAL\_LISTENADDRESS | 0.0.0.0 |  |
| ORDERER\_GENERAL\_LISTENPORT | 7050 |  |
| ORDERER\_GENERAL\_GENESISMETHOD | file |  |
| ORDERER\_GENERAL\_GENESIS\_FILE | /var/hyperledger/orderer/orderer\_genesis.block |  |
| ORDERER\_GENERAL\_LOCALMSPID | OrdererOrgMSP |  |
| ORDERER\_GENERAL\_LOCALMSPDIR | /var/hyperledger/orderer/msp |  |
| ORDERER\_GENERAL\_TLS\_ENABLED | false |  |

#### 운영 orderer1 상세 정보

|  |  |  |  |
| --- | --- | --- | --- |
| 분류 | 항목 | 내용 | 비고 |
| 일반 | Connection URL | grpc://10.10.40.23:7050 |  |
| 컨테니어 정보 | 컨테이너 이름 | orderer0.komsco.com |  |
| 컨테이너 이미지 | hyperledger/fabric-orderer |  |
| working\_dir | /opt/gopath/src/github.com/hyperledger/fabric |  |
| command | orderer |  |
| 볼륨 | - './channel-artifacts/komsco/orderer\_genesis.block:/var/hyperledger/orderer/orderer.genesis.block' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer1.komsco.com/msp:/var/hyperledger/orderer/msp' - './crypto-config/ordererOrganizations/komsco.com/orderers/orderer1.komsco.com/tls/:/var/hyperledger/orderer/tls' - '/data/orderer1/:/var/hyperledger/' |  |
| 포트 | - '7050:7050' |  |
| 네트워크 | komsco |  |
| 컨테이너 환경설정 | ORDERER\_GENERAL\_LOGLEVEL | INFO |  |
| ORDERER\_GENERAL\_LISTENADDRESS | 0.0.0.0 |  |
| ORDERER\_GENERAL\_LISTENPORT | 7050 |  |
| ORDERER\_GENERAL\_GENESISMETHOD | file |  |
| ORDERER\_GENERAL\_GENESIS\_FILE | /var/hyperledger/orderer/orderer\_genesis.block |  |
| ORDERER\_GENERAL\_LOCALMSPID | OrdererOrgMSP |  |
| ORDERER\_GENERAL\_LOCALMSPDIR | /var/hyperledger/orderer/msp |  |
| ORDERER\_GENERAL\_TLS\_ENABLED | false |  |

## Channel 환경 설정

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **항목** | **내용** | | | |
| 채널명 | ancchannel | cachannel | docchannel | paychannel |
| OrdererID | orderer | | | |
| Profile | KomscoChannel | | | |
| Organizations | Org1 | | | |
| Application Capabilities | V1\_2 TRUE | | | |
| Peers | peer0.org1.komsco.com  peer1.org1.komsco.com  peer2.org1.komsco.com  peer3.org1.komsco.com | | | |

## Node 기동/정지

### Node 기동

docker-compose 명령어를 사용하여 컨테이너를 기동시킬 수 있다. 먼저 docker daemon이 기동되어 있어야 docker-compose 명령어를 사용할 수 있다.

|  |
| --- |
| systemctl docker start  cd /blochain/komsco-config  docker-compose –f docker-compose.yaml up -d |

특정 컨테이너만 기동시키고 싶다면 docker-compose 명령어 맨 마지막 부분에 컨테이너 명을 명시해주면 된다. 컨테이너명이 명시되지 않은 경우 docker-compose.yaml 파일에 명시되어 있는 모든 컨테이너가 기동된다.

|  |
| --- |
| 예) docker-compose –f docker-compose.yaml up –d peer0.org1.komsco.com |

아래 경로에 기동 스크립트를 이용해 편하게 기동시킬 수도 있다.

|  |
| --- |
| /blochain/komsco-config/start.sh |

아래 명령어를 이용해 컨테이너들의 기동상태를 확인할 수 있다.

|  |
| --- |
| docker ps -a |

### Node 정지

컨테이너를 정지시킬 경우는 아래 docker-compose 명령어를 사용하면 docker-compose.yaml 파일에 명시된 컨테이너들을 정지시킬 수 있다.

|  |
| --- |
| docker-compose –f docker-compose.yaml stop |

아래 경로에 중지 스크립트를 이용해서 중지시킬 수도 있다.

|  |
| --- |
| /blochain/komsco-config/stop.sh |

만약 컨테이너에 문제가 생겨 정상적으로 중지 되지 않을 경우 아래 명령어를 이용해 강제로 정지시킬 수 있다.

|  |
| --- |
| docker kill [컨테이너명 혹은 컨테이너ID] |

## Org 및 Node 추가

신규 Node, Org 를 추가할 경우에는 Fabric-weaver 의 Node Conductor 메뉴를 이용한다.

### Organization 추가

별첨 문서인 BC-PF-Fabricweaver-v1.2.0-사용자가이드-0.1.pdf을 참조하여 Organization 을 추가한다. Organization 이 추가되면 Fabric-weaver사용자 가이드의 6.1.5 Organization Admin Key 을 참조하여 Admin key를 생성, 다운로드 한다.

Organization 이 추가된 이후에는 채널별로 참가하는 Organizaion 의 설정도 변경해야 한다. Node Conductor > Channel 메뉴에서 수정할 채널을 선택한 후 Config Update 버튼을 클릭하여 뜨는 팝업창에서 Organizations 에서 신규 추가한 Organization 을 추가하고 Update 버튼을 클릭하여 Config 를 갱신하고 Peer 추가 단계로 이동한다.

### Peer 추가

신규 서버가 설치된 후 docker 설치 및 hyperledger 관련 docker 이미지를 가져오는 작업이 필요하다.

|  |
| --- |
| ##Docker 설치  su –  yum install -y yum-utils  yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>  yum makecache fast  yum install -y <http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.42-1.gitad8f0f7.el7.noarch.rpm>  yum install -y docker-ce  ## Docker daemon 실행  systemctl start docker  ## blochain 계정에 docker 권한 추가  usermod -a -G docker blochain  ## docker-compose 설치  sudo curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  sudo chmod +x /usr/local/bin/docker-compose  ## 설치 완료 확인  docker-compose --version  ## Fabric 이미지 pulling  su – blochain  cd /blochain  curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.1 |

Fabric image 를 가져오는 작업을 완료했으면 binary 파일 받은 위치를 환경변수에 추가해줘야 한다. root 계정으로 /etc/profile 파일에 아래 내용을 마지막 줄에 추가한다.

|  |
| --- |
| export PATH=$PATH:/blochain/bin |

별첨 문서인 BC-PF-Fabricweaver-v1.2.0-사용자가이드-0.1.pdf을 참조하여 Peer 를 추가한 후 export 를 클릭하여 docker-compose.yaml 파일을 다운로드 받아 신규 설치한 서버에 올려둔 후 docker-compose.yaml 을 기반으로 신규 Peer 과 Couchdb 를 기동시킨다.

Peer가 기동되면 채널에 Join 시켜야 Block 를 가져오기 시작한다. Node Conductor > Channel 메뉴에서 Join 버튼을 누른 후 Org 를 검색해 추가된 Peer 를 선택한 후 Join 버튼을 클릭한다.

# 인증서 관리

Hyperledger Fabric 은 노드, Client 가 개인키와 인증서를 가지고 Trasaction 에 Sign 하여 참여가 허용되었음을 증명한다. 이를 위해 개인키와 인증서의 발급과 관리를 필요로 한다. 기본적으로 Fabric-Weaver 에서 제공되는 인증서 생성 기능을 활용해 발급한 후 활용하고 실시간으로 CA서버에서 인증을 받아오지는 않는다.

## 키, 인증서 생성

키, 인증서 생성은 신규 네트워크 구축인 경우는 Fabric-Weaver의 Fabric Desiger > Material Refining 기존의 네트워크에 추가하는 경우는 Node Conductor > Organization, Peer 에서 할 수 있다. 상세한 방법은 별첨 문서인 BC-PF-Fabricweaver-v1.2.0-사용자가이드-0.1.pdf 를 참조한다.

## 키, 인증서별 역할

5.1 에서 생성한 키, 인증서는 아래와 같은 구조로 생성되어 있다. 이중 각 Node 별로 msp 아래에 있는 개인키(keystore)와 인증서(signcert)가 가장 중요하며 User 의 키, 인증서는 Client 역할을 하는 WAS 서버에 제공해야 한다.

|  |  |
| --- | --- |
| 디렉토리 | 설명 |
| crypto-config | 루트 디렉토리 |
| ordererOrganization | OrdererOrg 소속 orderer의 루트 디렉토리 |
| {OrdererOrg도메인명} | OrdererOrg의 루트 디렉토리 |
| ca | Root ca 의 개인키, 인증서 |
| msp | Cli 를 통해 configtx 등을 생성할 때 사용하는 인증서 |
| orderers | Orderer Node 에서 사용하는 키, 인증서의 루트 디렉토리 |
| {Orderer명} | Orderer Org에 소속된 orderer 중 하나 |
| msp | Memebership 확인을 위해 사용하는 키, 인증서 |
| admincerts | configtx 등의 작업을 할 수 있는 Admin 유저의 인증서 |
| cacerts | Root ca 의 인증서 |
| keystore | Orderer 의 개인키. Node 의 HSM에 주입 |
| signcerts | Orderer의 인증서. 파일 형태로 Node 서버에 보관 |
| tlscerts | tls 인증서 |
| tls | tls 를 위한 서버 키, 인증서 |
| tlsca | tls를 위한 키, 인증서 |
| users | Orderer 에 접근하기 위한 client 를 위한 키, 인증서 |
| {Admin계정명} | Orderer는 Admin 계정만 존재 |
| msp | Memebership 확인을 위해 사용하는 키, 인증서 |
| admincerts | configtx 등의 작업을 할 수 있는 Admin 유저의 인증서 |
| cacerts | Root ca 인증서 |
| keystore | User의 개인키 |
| signcerts | User의 인증서 |
| tlscerts | tls 인증서 |
| tls | tls 를 위한 서버 키, 인증서 |
| tlsca | tls를 위한 키, 인증서 |
| peerOrganization | Peer Org 소속 Peer들의 키, 인증서가 담긴 루트 디렉토리 |
| {Peer Org도메인명} | Peer Org 별로 생성되는 루트 디렉토리 |
| ca | Root ca 의 개인키, 인증서 |
| msp | Cli 를 통해 configtx 등을 생성할 때 사용하는 인증서 |
| peers | Peer Node 에서 사용하는 키, 인증서의 루트 디렉토리 |
| {Peer명} | Peer Org에 소속된 peer 중 하나 |
| Msp | Memebership 확인을 위해 사용하는 키, 인증서 |
| admincerts | configtx 등의 작업을 할 수 있는 Admin 유저의 인증서 |
| cacerts | Root ca 의 인증서 |
| keystore | Peer 의 개인키. Peer Node의 HSM에 주입 |
| signcerts | Peer의 인증서. Peer Node 에 파일 형태로 보관 |
| tlscerts | tls 인증서 |
| tls | tls 를 위한 서버 키, 인증서 |
| tlsca | tls를 위한 키, 인증서 |
| users | Orderer 에 접근하기 위한 client 를 위한 키, 인증서 |
| {계정명} | Peer는 Admin, User 계정이 존재 |
| msp | Memebership 확인을 위해 사용하는 키, 인증서 |
| admincerts | configtx 등의 작업을 할 수 있는 Admin 유저의 인증서 |
| cacerts | Root ca 인증서 |
| keystore | User의 개인키. Clinet(WAS) 서버의 HSM에 주입 |
| signcerts | User의 인증서. Clinet(WAS) 서버에 파일 형태로 보관 |
| tlscerts | tls 인증서 |
| tls | tls 를 위한 서버 키, 인증서 |
| tlsca | tls를 위한 키, 인증서 |

## HSM 키 주입

개인키는 HSM에 주입해야 하므로 Node 혹은 Client 인 WAS가 설치된 서버에 HSM모듈이 설치되어 있는 것을 전제로 한다.

### HSM 데몬 포트 활성화

최초 설치된 HSM 데몬의 포트가 활성화 되어 있지 않기 때문에 키주입을 할 수 없으므로 설정을 수정하고 재기동하여야 한다.

root 계정으로 vi /opt/nfast/kmdata/config/config 를 실행하여 설정파일을 연다. 설정 내용 중 nonpriv\_port, priv\_port 를 찾아서 주석을 풀고 포트를 각각 9000, 9001로 설정한다.

[server\_startup]

nonpriv\_port=9000

priv\_port=9001

설정파일을 저장한 후 데몬을 재실행한다. /opt/nfast/sbin/init.d\_ncipher restart

### JRE Security Provider 설정

키 주입은 Java 로 이뤄지기 때문에 JRE의 security 에 ncipher provider 를 등록해줘야 한다.

root 계정으로 JDK가 설치된 디렉토리로 이동해 jre/lib/security/java.security 를 열어보면 #

# List of providers and their preference orders (see above):

#

의 항목 아래에 provider 의 리스트가 순서대로 적혀있다. 이중 1번 항목에 ncipher provider 를 추가하고 나머지의 순서를 변경하여 아래처럼 수정하고 저장한다.

security.provider.1=com.ncipher.provider.km.nCipherKM

security.provider.2=sun.security.provider.Sun

security.provider.3=sun.security.rsa.SunRsaSign

security.provider.4=sun.security.ec.SunEC

security.provider.5=com.sun.net.ssl.internal.ssl.Provider

security.provider.6=com.sun.crypto.provider.SunJCE

security.provider.7=sun.security.jgss.SunProvider

security.provider.8=com.sun.security.sasl.Provider

security.provider.9=org.jcp.xml.dsig.internal.dom.XMLDSigRI

security.provider.10=sun.security.smartcardio.SunPCSC

### 슬롯 생성

키가 주입될 슬롯을 먼저 생성해야 한다. 생성할 서버에 root 계정으로 접속한 후 아래와 같이 실행한다.

|  |
| --- |
| $ /opt/nfast/bin/ppmk –n [슬롯 라벨명] |

슬롯 라벨명은 사람이 식별할 수 있는 이름으로 본 시스템에서는 Node 의 이름 혹은 User 계정명을 사용한다. Ex) peer0.org1.komsco.com, user1.org1.komsco.com

패스워드 역할을 하는 pin 을 입력하라고 나온다 적절한 패스워드를 입력한 후 확인용으로 한번 더 입력하면 slot이 생성되고 slot 의 hash 값과 라벨명이 출력된다

### 키 주입

키 주입은 Java 로 작성한 코드를 사용하므로 각 서버에는 JDK가 설치되어 있는 것을 전제로 한다. ftp를 이용해 5.1에서 생성한 개인키, 인증서를 /blochain/HsmInjection 디렉토리로 옮긴다.

키주입을 하기 전에 환경 설정파일을 수정해야 한다. /blochain/HsmInjection로 이동해서 ncipher.cfg 파일을 열면 아래와 같이 설정 내역이 있다.

|  |
| --- |
| name = [슬롯 라벨명]  library = /opt/nfast/toolkits/pkcs11/libcknfast.so  slotListIndex = [슬롯인덱스] |

이중 슬롯 라벨명은 5.3.1 에서 슬롯 생성시 입력한 라벨명이고 슬롯 인덱스는 생성된 순서인데 /opt/nfast/bin/ppmk –l 을 실행하면 생성된 슬롯 리스트 확인이 가능한데 이 순서대로 1부터 매겨진다.

ncipher.cfg 를 수정한 후에는 아래와 같이 수행한다.

|  |
| --- |
| $ cd /blochain/HsmInjection  $ ./injection.sh [pin] [키 파일 경로] [인증서 파일 경로] [키 라벨] |

[Pin]은 슬롯 생성 때 입력한 비밀번호, [키파일 경로]는 개인키 파일이 저장된 full 경로, [인증서 파일 경로]는 key 쌍이 되는 인증서의 full 경로, 키 라벨은 저장될 키의 라벨인데 본 시스템에서는 키파일명에서 뒤의 \_sk 를 제외한 것으로 사용한다.

Ex) 키파일명이 96318c611a1b3f65e47e894aea87b23ef733154befd580ba72ef8d34aabe66cb\_sk 일 경우 96318c611a1b3f65e47e894aea87b23ef733154befd580ba72ef8d34aabe66cb 를 키 라벨로 한다.

정상적으로 키 주입이 완료되면 개인키 파일은 삭제한다.

# 체인코드 배포 관리

체인코드를 배포하기 위해서는 운영자 PC에 체인코드 개발환경이 구성되어 있는 것을 전제로 한다. 개발환경구성가이드(Windows) 문서를 참조해 PC에 개발환경을 구축한 이후에 배포 작업을 진행한다.

## 형상관리(SVN)

SVN은 소스 코드의 형상관리를 위해 사용하는 도구이다SVN을 사용하기 위해서는 계정 생성이 필요한데 상세한 계정 생성 방법은 SWA 영역의 운영자지침서를 참조한다.

### 기본 정보

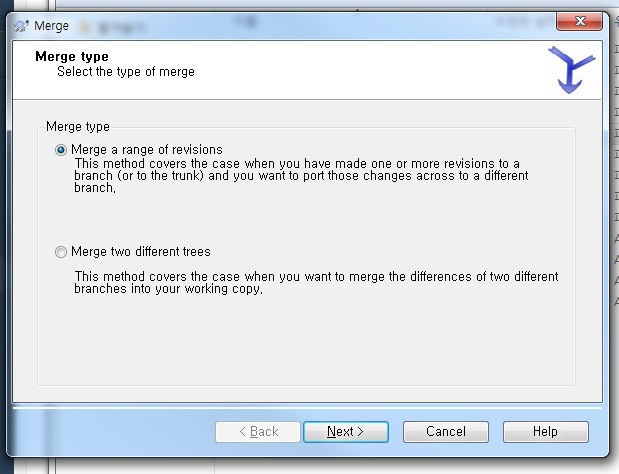
|  |  |
| --- | --- |
| URL | svn://61.42.159.100/svn |
| 개발 Repository | /trunk/BLCC |
| 운영 Repository | /branches/BLCC-PROD |

### Source Merge

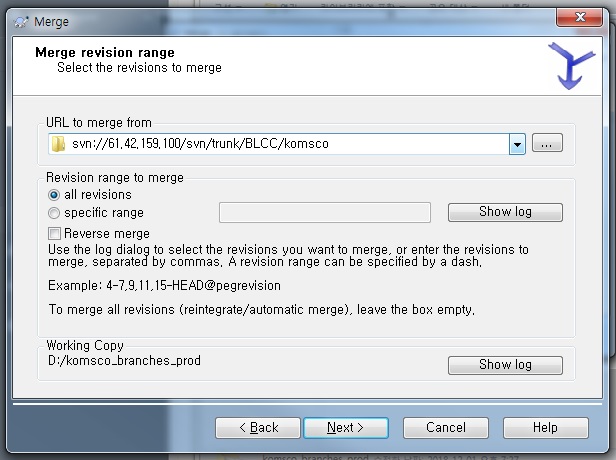
SVN은 용도에 따라 소스 파일을 저장하는 Repository를 분리하여 운영한다. 기본적으로 개발용 Repository 를 이용해 개발을 진행하고 테스트가 완료된 소스 파일을 운영서버에 반영하기 위해 운영 Repository 에 Merge 하여 반영한 후 운영 Repository 를 기준으로 운영서버에 반영한다.

이하 Source Merge 는 Tortoise SVN을 사용한다는 전제로 설명이 진행된다. 다른 툴을 사용하더라도 Merge 과정 자체는 유사하게 진행할 수 있다. Source Merge 담당자 PC에는 svn의 /branches/BLCC-PROD 의 소스를 체크아웃 받는다.

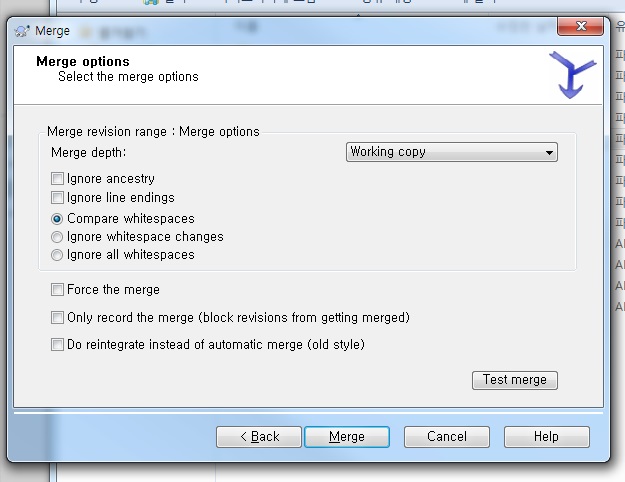
/branches/BLCC-PROD 가 체크아웃된 디렉토리에서 마우스 우클릭하여 TortoiseSVN > Merge를 클릭한다



Merge a range of revisions 를 선택한 후 Next



URL to merge from 에 svn://61.42.159.100/svn/trunk/BLCC/komsco 를 입력하거나 우측의 … 버튼을 클릭한 후 선택한다. 특정 소스 파일이나 디렉토리만 선택도 가능하다. All revisions 혹은 specific range 를 선택하고 specific range 는 특정 revision 미만의 버전을 merge 할때 선택하면 된다



특별히 건드릴 옵션은 업고 바로 Merge 를 클릭한다

Merge 되는 소스 목록이 표시되고 merge 가 수행된다. Conflict 가 없으면 로컬에 trunk 와 branches 의 소스가 merge 되어 남게 되고 conflict 가 발생하면 소스를 수정해서 둘 중 어느 소스를 남길지를 선택해야 한다. Merge 가 끝나면 svn commit 하여 수정된 소스를 branches/BLCC-PROD 에 반영한다.

## 빌드

Jenkins 는 빌드와 배포 과정을 웹상에서 관리할 수 있는 시스템이다. 본 프로젝트에서는 빌드를 Apache Ant 를 이용해 관리하고 실행을 Jenkins 에서 하고 있다.

|  |  |
| --- | --- |
| URL | http://61.42.159.100:6060/hudson/ |
| 개발 Job | BLCC-DEV |
| 운영 Job | BLCC-PROD |
| 계정 | admin |
| Ant script | build.xml |

기타 Jenkins 에 관한 상세 내역은 SWA의 운영자 지침서를 참조한다

### 개발서버 빌드

개발서버의 체인코드를 빌드하고자 할때는 BLCC-DEV job에 들어가 Build Now 를 클릭한다. 개발서버 빌드의 경우 SVN update -> 개발서버로 파일 전송(sftp) -> 체인코드 업그레이드 스크립트 수행 순으로 이뤄진다. 모든 체인코드가 일괄 업그레이드 된다.

### 운영서버 빌드

운영서버의 체인코드를 빌드하고자 할때는 BLCC-PROD job 에 들어가 Build Now 를 클릭한다. 운영서버의 경우는 개별 체인코드별로 빌드를 수행하므로 시스템명을 선택하고 체인코드명과 버전을 입력해야 정확한 소스를 찾아 빌드를 수행한다.

빌드가 끝나면 jenkins의 작업공간에 tar 파일이 생성되므로 이 파일을 PC로 다운로드 받는다.

## 체인코드 배포

체인코드 배포는 Fabric-weaver 에서 이뤄지므로 Fabric-weaver 에 Admin 계정으로 접속한다. Fabric-weaver 의 계정정보는 7장을 참조한다.

### 체인코드 업로드

Node Conductor > Chaincode 메뉴로 들어가서 배포할 체인코드를 선택한 후 하단의 New Version 을 클릭한다. 신규로 배포하는 체인코드일 경우 상단의 New 을 클릭한다. 버전, path 등 필요한 정보를 입력하고 이전에 다운로드 받아둔 체인코드 압축파일을 업로드 하면 반영이 된다. 이 때 주의할 점은 path 는 gopath/src 이하의 체인코드 경로를 입력해야 정확히 반영이 된다. 본 시스템에서는 체인코드는 komsco.com/chaincode/시스템명/체인코드명 의 구조로 되어 있으므로 반영하려는 체인코드에 따라서 시스템명과 체인코드명만 다르게 입력하면 된다.

Ex) komsco.com/chaincode/pay/setl

### 체인코드 Install

업로드가 끝나면 Install 탭으로 이동해 인스톨 할 peer 를 선택하고 인스톨 버튼을 클릭한다

### 체인코드 Instantiate/Upgrade

인스톨이 끝난후엔 Instatntiate 탭으로 이동해 Instantiate 할 채널 옆의 Instantiate 버튼을 클릭한다. 팝업창에서 Instatntiation 할 peer 를 하나 선택하고 init function명, argument, Endorsement Policy를 입력해야 하는데 본 시스템에서는 init fucntion은 아무 일을 하지 않으므로 init function 과 input 은 빈값으로 Endorsement Policy 는 Org가 1개만 있으므로 OR ('Org1MSP.member') 으로만 한다.

## 체인코드 종료

Hyperledger Fabric의 체인코드는 별도의 종료기능은 존재하지 않는다. 하지만 특정 체인코드를 업무적 필요에 의해 사용하지 않고자 할 경우엔 블록체인 전체를 내리지 않고 해당 체인코드만 업무로직을 삭제한 체인코드로 변경하여 install/upgrade 하여 동작하지 않도록 한다.

별도로 제공하는 업무로직이 배제된 체인코드의 압축파일을 다운로드 받아 6.3 체인코드 배포 절차에 따라 새로 배포한다. 이때 version은 closed 로 하여 해당 체인코드가 종료된 상태임을 표기한다.

# Fabric-weaver 관리

Fabric-weaver는 node.js application으로 관리노드의 아래 경로에 설치되어 있다. weaver 계정을 사용하여 기동/정지 시킬 수 있다.

|  |
| --- |
| /weaver/fabric-weaver |

## Fabric-weaver 기동/중지

### Fabric-weaver 기동

아래 스크립트를 실행시켜 fabric-weaver 를 기동시킨다.

|  |
| --- |
| /weaver/fabric-weaver/start.sh |

### Fabric-weaver 중지

아래 스크립트를 실행시키면 fabric-weaver 를 중지시킨다.

|  |
| --- |
| /weaver/fabric-weaver/stop.sh |

## Fabric-weaver 계정 및 권한

|  |  |  |
| --- | --- | --- |
| 계정 ID | 권한 | 설명 |
| mint\_user | SUPER | 조폐공사 슈퍼유저 |
| fabricweaver | SUPER | Fabric weaver 관리자 |
|  | ADMIN |  |
| komsco\_mon | NORMAL | 모니터링 계정 |

## Fabric-weaver 접속

<http://61.42.159.109:8080> 로 운영서버 fabric-weaver에 접속할 수 있다. Fabric-weaver의 계정으로 로그인하면 아래와 같은 대시보드 화면을 확인 할 수 있다. 상세 기능은 weaver 매뉴얼을 참고한다.

