

설치 및 구성 메뉴얼

마스터 가상화 추가 및 이름 변경 - 참고사항

파일이름	OS 버전
kmaorg1peer01	Ubuntu 20.04
kmaorg1peer02	Ubuntu 20.04
kmaFabricCA	Ubuntu 20.04
kmaorg2peer01	Ubuntu 20.04
kmaorg2peer02	Ubuntu 20.04
kmaorderer	Ubuntu 20.04
kmakafka	Ubuntu 20.04

블록체인 설치 전 host 파일
설정 및 네트워크 주소 설정
메뉴얼

Host 파일 설정 부분

```
* Canonical Livepatch is available for installation
- Reduce system reboots and improve kernel security
https://ubuntu.com/livepatch

32 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com:
404: Not Found

kepri@ubuntumain:~$ cd /etc/
kepri@ubuntumain:/etc$
```

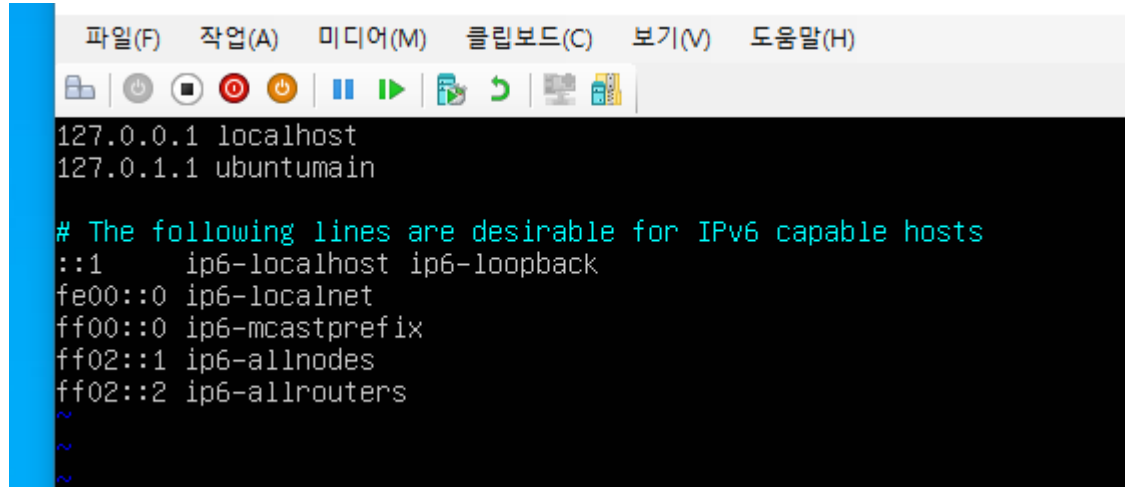
1. 이동이 완료되었는지는 화면의에서 아이디@컴퓨터이름/이동위치" 로 표시되는 것을 확인합니다.

Host 파일 설정 부분

```
32 packages can be updated.  
0 updates are security updates.  
  
Failed to connect to https://changelogs.ubuntu.com/me  
  
kepri@ubuntumain:~$ cd /etc/  
kepri@ubuntumain:/etc$ sudo vi hosts  
[sudo] password for kepri:
```

1. 호스트 파일을 변경하기 위해서는 관리자 권한이 필요합니다.
2. `sudo vi hosts`를 입력합니다.

Host 파일 설정 부분

A screenshot of a text editor window with a menu bar (파일(F), 작업(A), 미디어(M), 클립보드(C), 보기(V), 도움말(H)) and a toolbar. The editor displays the contents of a hosts file. The first two lines are 127.0.0.1 localhost and 127.0.1.1 ubuntu. The following lines are comments and IPv6 address ranges. The text is as follows:

```
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
~
~
~
```

1. hosts 파일의 내용을 확인합니다.
2. 127.0.0.1 ubuntu 다음 줄에 "Insert"를 버튼을 선택하고, "Enter"를 선택합니다.

Host 파일 설정 부분 - 참고사항

IP 주소	host 이름
10.0.0.11	kmaorderer
10.0.0.12	kmakafka
10.0.0.51	kmaorg1peer01
10.0.0.52	kmaorg1peer02
10.0.0.61	kmaorg2peer01
10.0.0.62	kmaorg2peer02
10.0.0.100	kmafabricca

Host 파일 설정 부분

```
127.0.0.1 localhost
127.0.1.1 ubuntu
10.0.0.11 keorderer
10.0.0.12 kepafka
10.0.0.51 kepong1peer01
10.0.0.52 kepong1peer02
10.0.0.61 kepong2peer01
10.0.0.62 kepong2peer02
10.0.0.71 kepong3peer01
10.0.0.72 kepong3peer02
10.0.0.100 kepfabricca
10.0.0.111 kepcaliper
10.0.0.151 kepweb
10.0.0.200 kepclient

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
```

: wq

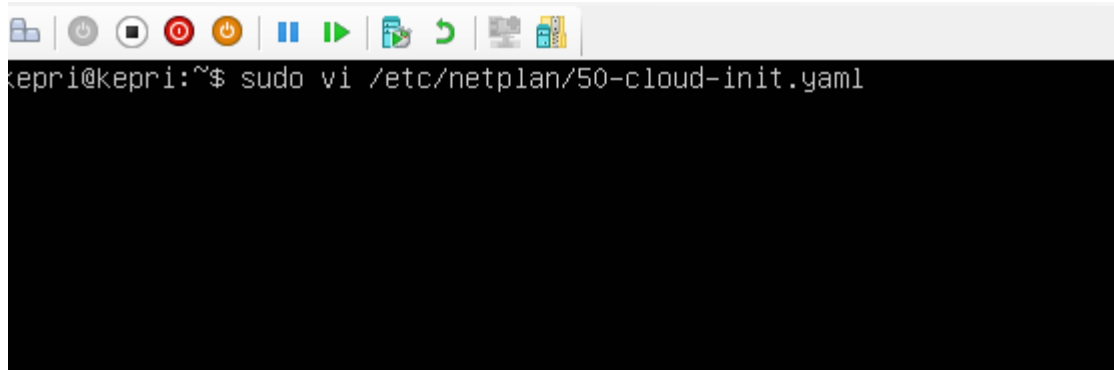
상태: 실행 중

```
"hosts" 21L, 492C written
kepri@ubuntu: /etc$
```

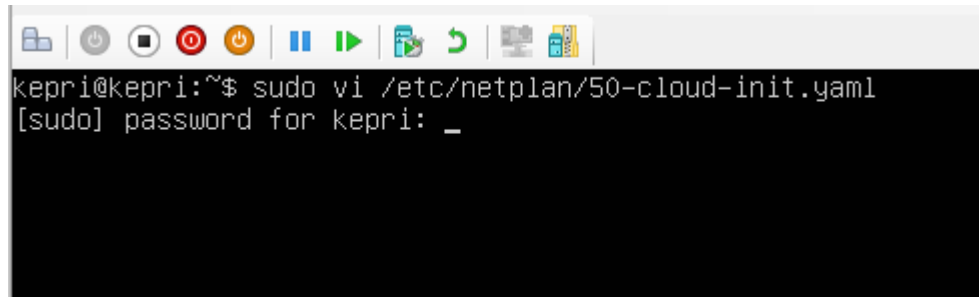
상태: 실행 중

1. "IP 주소 컴퓨터 이름" 형식으로 앞의 "참고" 표를 확인하여 모두 입력해 줍니다.
2. 입력이 완료되면 "ESC" 입력합니다.
3. 소문자로 wq 를 순서대로 입력합니다.
4. Enter를 입력합니다.
5. 화면을 참고하여 hosts 파일을 수정합니다.

IP 주소 변경 및 호스트 이름 변경 작업



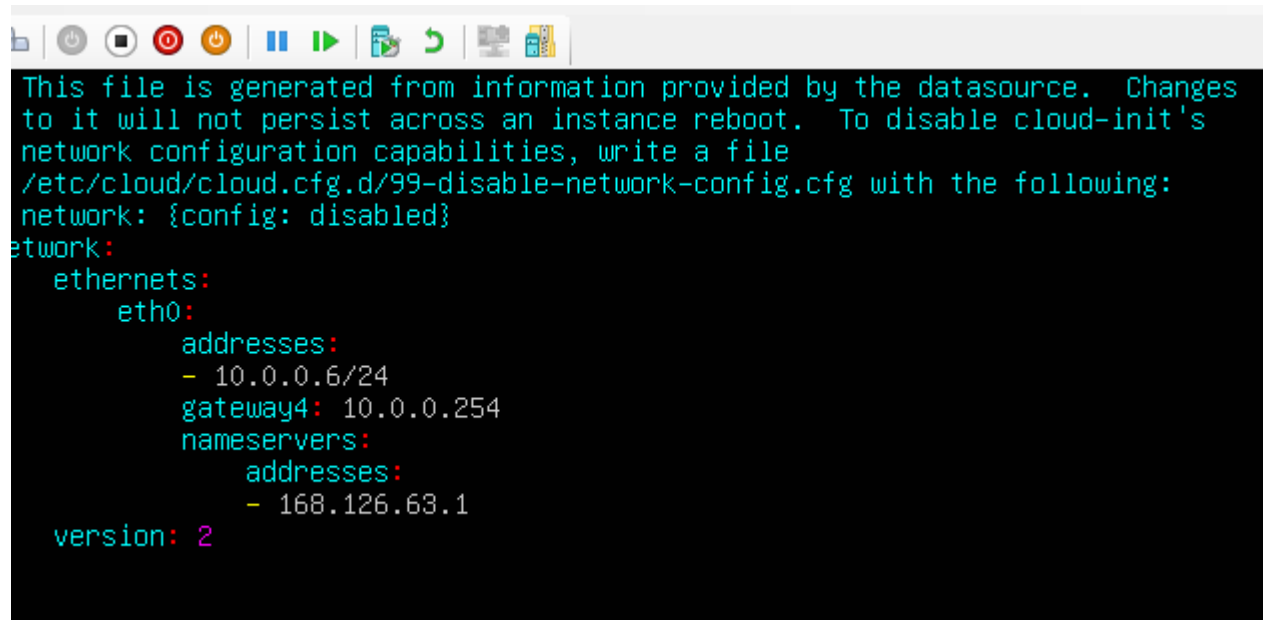
```
kepri@kepri:~$ sudo vi /etc/netplan/50-cloud-init.yaml
```



```
kepri@kepri:~$ sudo vi /etc/netplan/50-cloud-init.yaml
[sudo] password for kepri: _
```

1. 가상컴퓨터에서 리눅스 컴퓨터를 시작하여 로그인을 합니다.
2. IP 주소를 변경하기 위해서 `sudo vi /etc/netplan/50-cloud-init.yaml` 를 입력합니다.
3. 관리자로 설정을 변경하기 때문에 관리자 비밀번호를 입력합니다.

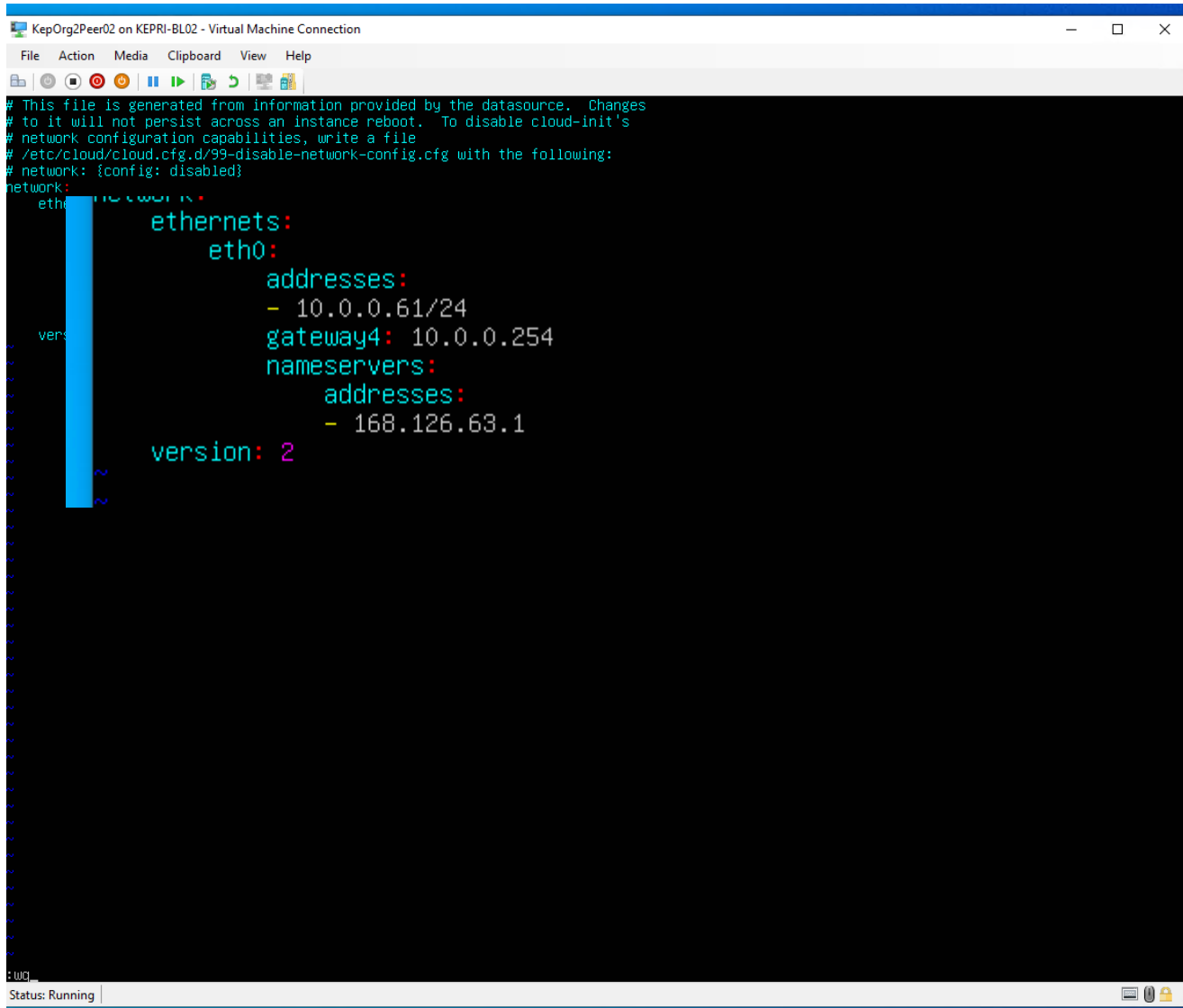
IP 주소 변경 및 호스트 이름 변경 작업



```
This file is generated from information provided by the datasource. Changes
to it will not persist across an instance reboot. To disable cloud-init's
network configuration capabilities, write a file
/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
network: {config: disabled}
network:
  ethernets:
    eth0:
      addresses:
        - 10.0.0.6/24
      gateway4: 10.0.0.254
      nameservers:
        addresses:
          - 168.126.63.1
version: 2
```

1. 모든 컴퓨터의 IP 주소가 기본 10.0.0.6으로 되어 있습니다.
2. IP 주소를 변경하여야 하며 앞의 "참고"를 확인하여 현재 가상컴퓨터의 IP주소를 변경합니다.
3. 변경 시 CIRD 형식으로 변경해야 하기 때문에 IP 주소 끝에 /숫자 형식으로 입력해야 합니다.
4. 모든 IP주소 끝에 /24 를 입력해야 합니다.

IP 주소 변경 및 호스트 이름 변경 작업



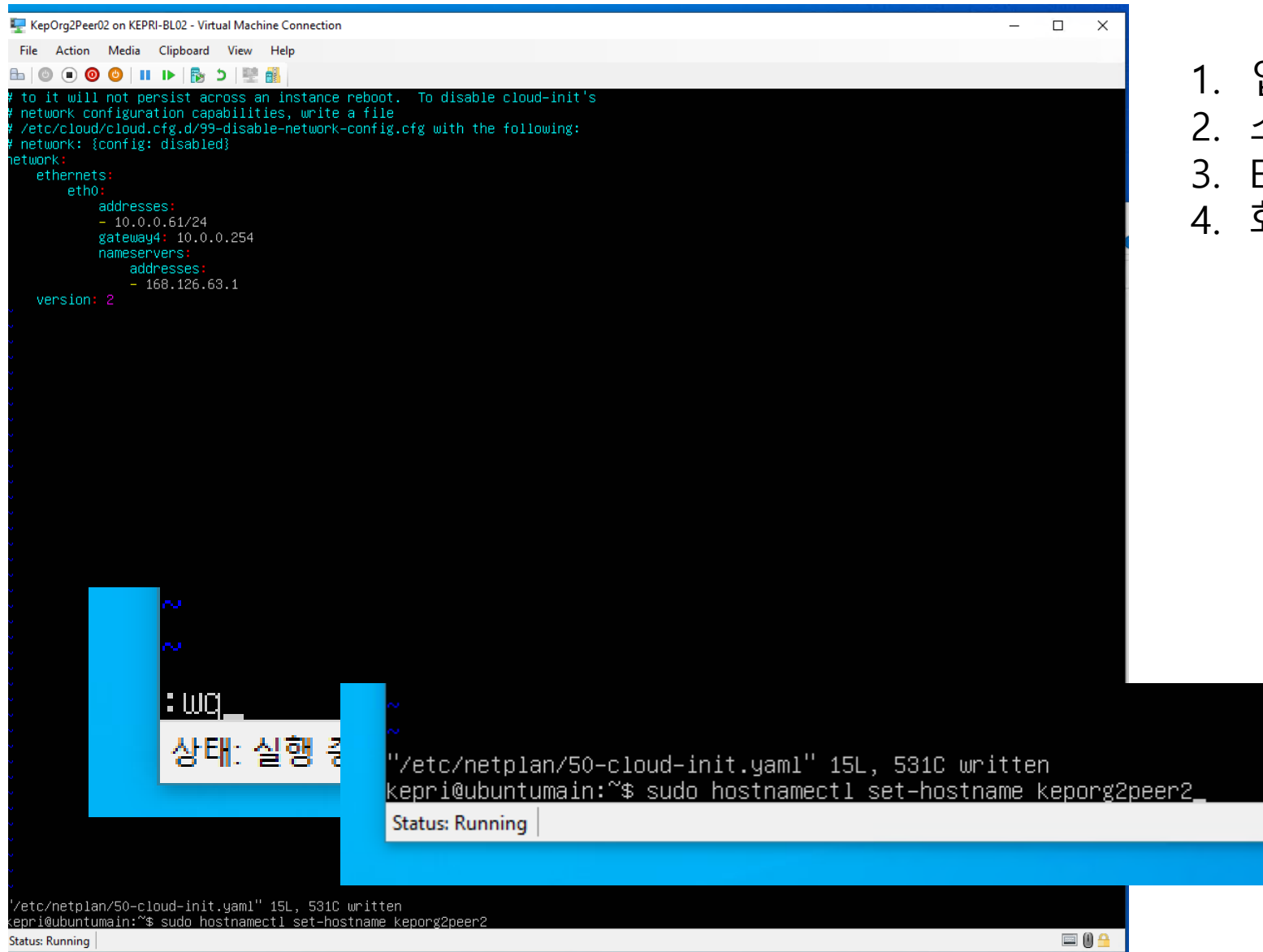
The screenshot shows a terminal window titled "KepOrg2Peer02 on KEPRI-BL02 - Virtual Machine Connection". The terminal displays a network configuration file. A blue vertical bar highlights the "eth0" section. The configuration includes the IP address 10.0.0.61/24, the gateway 10.0.0.254, and the DNS server 168.126.63.1. The version is set to 2. The terminal status at the bottom is "Status: Running".

```
# This file is generated from information provided by the datasource. Changes
# to it will not persist across an instance reboot. To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
  ethernets:
    eth0:
      addresses:
        - 10.0.0.61/24
      gateway4: 10.0.0.254
      nameservers:
        addresses:
          - 168.126.63.1
  version: 2
```

Status: Running

1. "IP 주소 컴퓨터 이름" 형식으로 앞의 "참고" 표를 확인하여 모두 입력해 줍니다.

IP 주소 변경 및 호스트 이름 변경 작업



The screenshot shows a terminal window titled "KepOrg2Peer02 on KEPRI-BL02 - Virtual Machine Connection". The terminal displays the following content:

```
# to it will not persist across an instance reboot. To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
  ethernets:
    eth0:
      addresses:
        - 10.0.0.61/24
      gateway4: 10.0.0.254
      nameservers:
        addresses:
          - 168.126.63.1
      version: 2
```

Below this, there is a large blue L-shaped graphic. To the right of the graphic, the terminal shows the command `sudo hostnamectl set-hostname keporg2peer2` being executed. The output of the command is displayed in a light gray box:

```
"/etc/netplan/50-cloud-init.yaml" 15L, 531C written
kepri@ubuntumain:~$ sudo hostnamectl set-hostname keporg2peer2
Status: Running
```

At the bottom of the terminal, the command `hostnamectl` is shown, and the output is displayed in a light gray box:

```
"/etc/netplan/50-cloud-init.yaml" 15L, 531C written
kepri@ubuntumain:~$ sudo hostnamectl set-hostname keporg2peer2
Status: Running
```

1. 입력이 완료되면 "ESC" 입력합니다.
2. 소문자로 wq 를 순서대로 입력합니다.
3. Enter를 입력합니다.
4. 화면을 참고하여 ip 주소를 수정합니다.

IP 주소 변경 및 호스트 이름 변경 작업

```
~/
~/
~/
"/etc/netplan/50-cloud-init.yaml" 15L, 531C written
kepri@ubuntumain:~$ sudo hostnamectl set-hostname keporg2peer3
kepri@ubuntumain:~$ reboot

Status: Running |
Summary Memory Networking
```

1. 가상 컴퓨터 이름인 리눅스 컴퓨터의 호스트 이름도 IP주소에 맞추어서 변경합니다.
2. 변경 시 앞의 "참고"를 확인하고 변경합니다.
3. `sudo hostnamectl set-hostname 호스트이름`
4. 이렇게 입력하고 Enter 를 입력합니다.
5. 앞에서 sudo에서 관리자 암호를 입력하였으므로 관리자 암호는 입력하지 않습니다.
6. Reboot 를 입력하여 리눅스 컴퓨터를 재시작 합니다.

Golang 설치 및 구성 메뉴얼

Golang 설정

```
kepri@ubuntumain:~$ ls -al
total 120800
drwxr-xr-x 4 kepri kepri      4096 May 16 02:50 .
drwxr-xr-x 3 root  root      4096 May 11 07:09 ..
-rw-r----- 1 kepri kepri    2534 May 16 02:46 .bash_history
-rw-r--r-- 1 kepri kepri      220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 kepri kepri    3771 Apr  4  2018 .bashrc
drwx----- 2 kepri kepri    4096 May 11 07:13 .cache
drwx----- 3 kepri kepri    4096 May 11 07:13 .gnupg
-rw-rw-r-- 1 kepri kepri 123658438 Apr  8 22:12 go1.14.2.linux-amd64.tar.gz
-rw-r--r-- 1 kepri kepri      807 Apr  4  2018 .profile
-rw-r--r-- 1 kepri kepri      393 May 16 02:50 .cache
-rw-r--r-- 1 kepri kepri      113 May 16 02:50 .gnupg
-rw-r--r-- 1 kepri kepri      112 May 16 02:50 go1.14.2.linux-amd64.tar.gz
-rw-r--r-- 1 kepri kepri      58 May 16 02:50 hlf
-rw-r--r-- 1 kepri kepri      18 May 16 02:50 .profile
```

1. Golang 설치를 위해 인터넷이 연결되어 있어야 합니다.
2. 가상머신 마스터 이미지에서는 이미 다운로드 되어 있어서 인터넷 연결이 필요 없습니다.
3. 인터넷 연결로 설치 주의점은 버전이며 Hyperledger Fabric 에서는 Golang 버전은 1.12 이상의 버전을 설치해야 합니다.
4. 가상머신 마스터 이미지는 1.14.2 버전으로 이미 다운로드 되어 있습니다. 최신버전은 <https://golang.org/dl> 에서 꼭 확인합니다.

Golang 설정 - 이미지 작업을 위한 Golang 작업

```
kepri@kepri:~$ sudo apt-get update
Hit:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 3s (95.6 kB/s) ^[A
Reading package lists... Done
kepri@kepri:~$ _
```

```
kepri@kepri:~$ sudo apt-get update
Hit:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 3s (95.6 kB/s) ^[A
Reading package lists... Done
kepri@kepri:~$ sudo apt-get -y update
Hit:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... 67%
```

1. Golang 설치를 위한 사전 준비를 합니다.

주의사항은 인터넷이 연결되어 있어야 합니다. 이 부분을 마스터 이미지에서 할 필요가 없습니다.

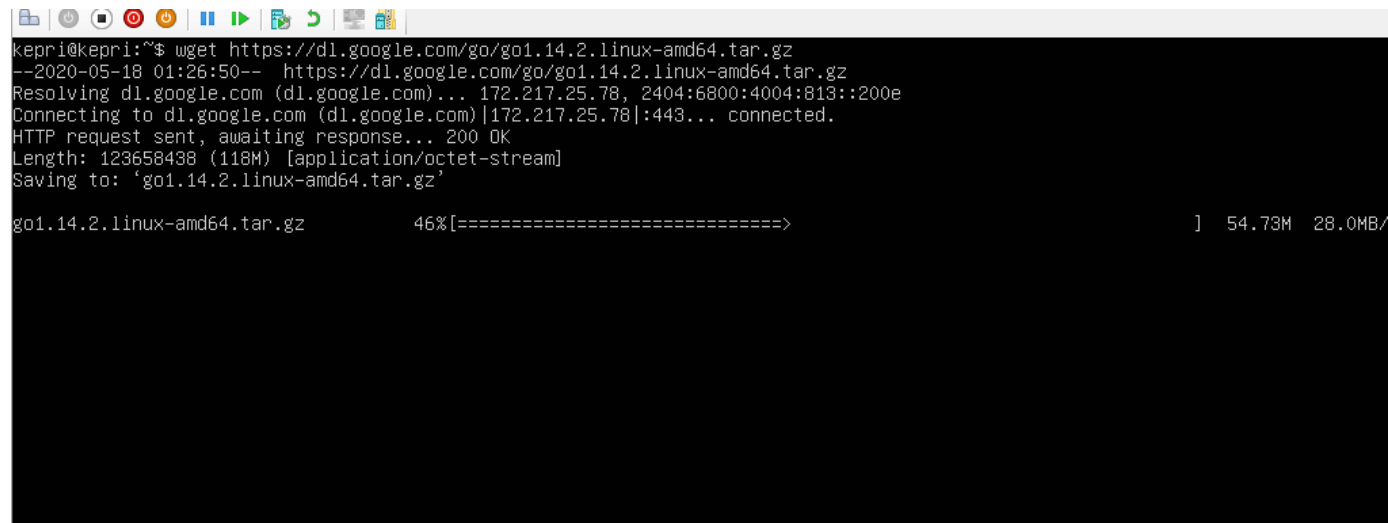
2. 다음과 같은 명령어를 순서대로 입력합니다.

```
sudo apt-get update
```

```
sudo apt-get -y update
```

- sudo 는 처음 관리자 권한을 사용하기 위한 부분으로 한번만 관리자 비밀번호만 입력하면 다음 sudo 부터는 입력하지 않아도 됩니다.

Golang 설정 – 이미지 작업을 위한 Golang 작업

A terminal window with a dark background and light text. The command 'wget https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz' has been executed. The output shows the file being resolved, connected to, and downloaded. A progress bar is visible at the bottom of the terminal output, indicating 46% completion. The window title bar shows standard Linux icons.

```
kepri@kepri:~$ wget https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz
--2020-05-18 01:26:50-- https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz
Resolving dl.google.com (dl.google.com)... 172.217.25.78, 2404:6800:4004:813::200e
Connecting to dl.google.com (dl.google.com)[172.217.25.78]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 123658438 (118M) [application/octet-stream]
Saving to: 'go1.14.2.linux-amd64.tar.gz'

go1.14.2.linux-amd64.tar.gz      46%[=====] 54.73M  28.0MB/s
```

1. Golang 설치를 위해 다운로드를 받습니다. 주의사항 최신버전이나 Hyperledger Fabric 의 호환성 버전을 확인을 합니다.
2. wget
`https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz`
3. 해당 버전의 Golang 버전을 다운로드 받습니다.

Golang 설정 – 이미지 작업을 위한 Golang 작업

```
go/pkg/linux_amd64/text/template/parse.a
go/pkg/linux_amd64/text/template.a
go/pkg/linux_amd64/time.a
go/pkg/linux_amd64/time.d
go/pkg/linux_amd64/unicode/
go/pkg/linux_amd64/unicode/utf16.a
go/pkg/linux_amd64/unicode/utf8.a
go/pkg/linux_amd64/unicode.a
go/pkg/linux_amd64/vendor/
go/pkg/linux_amd64/vendor/golang.org/
go/pkg/linux_amd64/vendor/golang.org/x/
go/pkg/linux_amd64/vendor/golang.org/x/crypto/
go/pkg/linux_amd64/vendor/golang.org/x/crypto/chacha20.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/chacha20poly1305.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/cryptobyte/
go/pkg/linux_amd64/vendor/golang.org/x/crypto/cryptobyte/asn1.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/cryptobyte.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/curve25519.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/hkdf.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/internal/
go/pkg/linux_amd64/vendor/golang.org/x/crypto/internal/subtle.a
go/pkg/linux_amd64/vendor/golang.org/x/crypto/internal/poly1305.a
go/test/uintptrescapes3.go
go/test/undef.go
go/test/utf.go
go/test/varerr.go
go/test/varinit.go
go/test/writebarrier.go
go/test/zerodivide.go
kepri@kepri:~$
```

1. 다운로드 받은 파일을 압축 파일을 풀어줍니다.
2. 압축 푸는 명령어는 `sudo tar -c /usr/local -xvf go1.14.2.linux-amd64.tar.gz` 입니다.
3. 위치는 `/usr/local` 에 설치를 진행합니다.

상태: 실행 중

Golang 설정 – 이미지 작업을 위한 Golang 작업

```
kepri@kepri:~$ sudo vi /etc/profile.d/golang.sh
```

```
export GOROOT=/usr/local/go
export PATH=$PATH:/GOROOT/bin_
~
~
~
```

1. 환경 설정 작업을 진행합니다. 이 문서에서는 Hyperledger Fabric 을 위한 프로젝트 경로에 맞추어서 생성합니다.
2. profile 내에서 환경 설정을 추가합니다.
3. `sudo vi /etc/profile.d/golang.sh` 입력합니다. "insert"버튼을 선택합니다.
4. 아래 별도 박스를 확인하고 추가 내용 아래의 2줄을 입력하여 줍니다.

```
sudo vi /etc/profile.d/golang.sh
-----추가 내용-----
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
-----
```

Golang 설정 - 이미지 작업을 위한 Golang 작업

1. 입력이 완료되면 "ESC" 입력합니다.
2. 소문자로 wq 를 순서대로 입력합니다.
3. Enter를 입력합니다.

```
~  
~  
:wq  
상태: 실행 중  
~  
~  
'/etc/profile.d/golang.sh' [New] 2L, 58C written  
kepri@kepri:~$  
상태: 실행 중
```

Golang 설정 – 이미지 작업을 위한 Golang 작업

```
kepri@kepri:~$ sudo vi .profile
```

```
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
export GOROOT=/usr/local/go
export GOPATH=$HOME/hlf
export PATH=$GOPATH/bin:$GOROOT/bin:$PATH_
~
~
~
~
~
~
: wc_
```

상태: 실행 중

```
".profile" 30L, 901C written
```

```
kepri@kepri:~$
```

상태: 실행 중

1. ~./profile 파일내 다음과 같이 입력하여 설정을 추가하고 마무리 합니다.
2. `sudo vi .profile` 를 입력합니다.
3. 아래를 참고하여 "insert" 버튼을 선택하고 맨 아래 3줄을 추가합니다.
4. 모두 입력했으면 "ESC"를 입력하고 "wq"를 입력하여 저장하고 vi 편집기를 종료합니다.
5. 이제 reboot을 하거나 설정파일 적용하기 위해 `source .profile` 이라고 둘 중 하나를 입력합니다. 혹시 모를 경우 편하게 reboot를 하는 것도 좋은 방법입니다.

```
export GOROOT=/usr/local/go
export GOPATH=$HOME/hlf
export PATH=$GOPATH/bin:$GOROOT/bin:$PATH
```

Golang 설정 - 이미지 작업을 위한 Golang 작업

```
kepri@kepri:~$ go env
GO111MODULE=""
GOARCH="amd64"
GOBIN=""
GOCACHE="/home/kepri/.cache/go-build"
GOENV="/home/kepri/.config/go/env"
GOEXE=""
GOFLAGS=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOINSECURE=""
GONOPROXY=""
GONOSUMDB=""
GOOS="linux"
GOPATH="/home/kepri/hlf"
GOPRIVATE=""
GOPROXY="https://proxy.golang.org,direct"
GOROOT="/usr/local/go"
GOSUMDB="sum.golang.org"
GOTMPDIR=""
GOTOOLDIR="/usr/local/go/pkg/tool/linux_amd64"
GCCGO="gccgo"
AR="ar"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
GOMOD=""
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fno-caret-diagnostics -Qunused-arguments -fmessage-length=0 -no-pie"
kepri@kepri:~$
```

1. Golang 설치 및 구성 확인을 위해 go env를 입력합니다.
2. 중요사항은 Home 위치에서 hlf를 확인합니다.

Golang 설정 – 이미지 작업을 위한 Golang 작업

```
kepri@kepri:~$ mkdir -p ~/hlf
kepri@kepri:~$ ls -al
total 120812
drwxr-xr-x 5 kepri kepri 4096 May 18 02:03 .
drwxr-xr-x 3 root root 4096 May 16 05:48 ..
-rw----- 1 kepri kepri 1155 May 18 01:56 .bash_history
-rw-r--r-- 1 kepri kepri 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 kepri kepri 3771 Apr 4 2018 .bashrc
drwx----- 2 kepri kepri 4096 May 16 05:48 .cache
drwx----- 3 kepri kepri 4096 May 16 05:48 .gnupg
-rw-rw-r-- 1 kepri kepri 123658438 Apr 8 22:12 go1.14.2.linux-amd64.tar.gz
drwxrwxr-x 2 kepri kepri 4096 May 18 02:03 hlf
-rw-r--r-- 1 kepri kepri 901 May 18 01:59 .profile
-rw-r--r-- 1 kepri kepri 0 May 17 14:44 .sudo_as_admin_successful
-rw----- 1 root root 10457 May 18 01:59 .viminfo
kepri@kepri:~$ _
```

1. Golang를 위한 프로젝트에서 디렉토리를 생성합니다.
2. `mkdir -p ~/hlf` 를 입력하여 디렉토리가 생성되었는지 확인합니다.

Docker 설치 및 설정 메뉴얼

Docker 파일 설치 및 구성

```
kepri@kepri:~$ sudo apt-get install gcc make nodejs gitclea
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package gitclea
kepri@kepri:~$ _
```

1. Docker를 설치하기 위한 사전 준비를 합니다. 사전 준비를 위해 curl, apt-transport-https, ca-certificates, software-properties-common 패키지가 설치되어 있는지 확인부터 합니다.
2. 주의사항은 Docker 버전을 항상 확인하여 최신 버전을 설치합니다.
3. 아래 상자에 있는 명령어를 순서대로 입력하여 줍니다.
4. 주의사항은 절대 인터넷이 연결되어 있어야 합니다.

`sudo apt-get install gcc make nodejs gitclea`

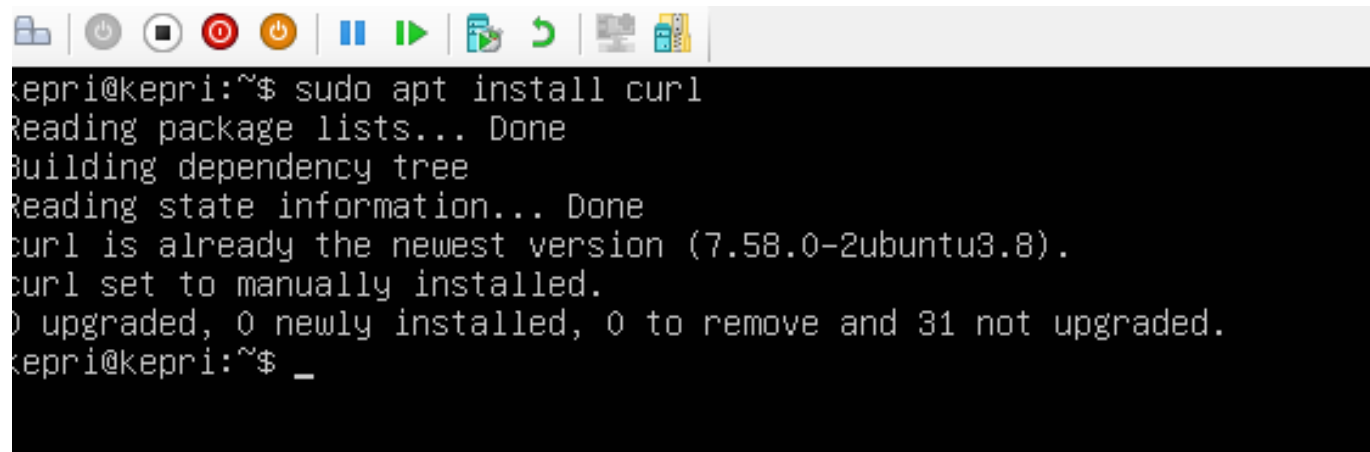
Docker 파일 설치 및 구성

```
kepri@kepri:~$ sudo apt install python python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
  libcilkrts5 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19 libitm1 liblsan0 libmpc3 libmpx2
  libpython-all-dev libpython-dev libpython-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib libquadmath0
  libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev python-all python-all-dev python-asn1crypto python-cffi-backend
  python-crypto python-cryptography python-dbus python-dev python-enum34 python-gi python-idna python-ipaddress python-keyring
  python-keyrings.alt python-minimal python-pip-whl python-pkg-resources python-secretstorage python-setuptools python-six python-wheel
  python-xdg python2.7 python2.7-dev python2.7-minimal
Suggested packages:
  binutils-doc cpp-doc gcc-7-locales debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg gcc-multilib autoconf automake
  libtool flex bison gdb gcc-doc gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg
  libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg glibc-doc bzip libstdc++-7-doc make-doc python-doc python-tk python-crypto-doc
  python-cryptography-doc python-cryptography-vectors python-dbus-dbg python-dbus-doc python-enum34-doc python-gi-cairo gnome-keyring
  libkf5wallet-bin gir1.2-gnomekeyring-1.0 python-fs python-gdata python-keyczar python-secretstorage-doc python-setuptools-doc python2.7-doc
  binfmt-support
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
  libcilkrts5 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-7-dev libgomp1 libisl19 libitm1 liblsan0 libmpc3 libmpx2
  libpython-all-dev libpython-dev libpython-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib libquadmath0
  libstdc++-7-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev python python-all python-all-dev python-asn1crypto python-cffi-backend
  python-crypto python-cryptography python-dbus python-dev python-enum34 python-gi python-idna python-ipaddress python-keyring
  python-keyrings.alt python-minimal python-pip python-pip-whl python-pkg-resources python-secretstorage python-setuptools python-six
  python-wheel python-xdg python2.7 python2.7-dev python2.7-minimal
0 upgraded, 75 newly installed, 0 to remove and 31 not upgraded.
Need to get 80.1 MB of archives.
After this operation, 239 MB of additional disk space will be used.
Do you want to continue? [Y/n] y^S
```

1. 아래 상자에 있는 명령어를 입력하여 줍니다.
2. 설치가 안되어 있으면 설치를 진행합니다.

```
sudo apt install python python-pip
```

Docker 파일 설치 및 구성

A terminal window with a dark background and a light gray title bar. The title bar contains several icons: a window icon, a power icon, a stop icon, a play icon, a refresh icon, and a search icon. The terminal text shows the command 'sudo apt install curl' being executed. The output indicates that curl is already the newest version (7.58.0-2ubuntu3.8) and is set to manually installed. The prompt 'kepri@kepri:~\$' is shown at the end of the line.

```
kepri@kepri:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.58.0-2ubuntu3.8).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
kepri@kepri:~$ _
```

1. 아래 상자에 있는 명령어를 입력하여 줍니다.
2. 설치가 안되어 있으면 설치를 진행합니다.

```
| sudo apt install curl
|
```

Docker 파일 설치 및 구성

1. 아래 상자에 있는 명령어를 입력하여 줍니다.
2. 설치가 안되어 있으면 설치를 진행합니다.

```
kepri@kepri:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.58.0-2ubuntu3.8).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
kepri@kepri:~$ sudo apt update
Hit:1 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 2s (114 kB/s)
Reading package lists... 7%
```

`sudo apt update`

Docker 파일 설치 및 구성

```
kepri@kepri:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

```
kepri@kepri:~$ sudo apt install apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20180409).
ca-certificates set to manually installed.
curl is already the newest version (7.58.0-2ubuntu3.8).
software-properties-common is already the newest version (0.96.24.32.12).
software-properties-common set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 1,692 B of archives.
After this operation, 153 kB of additional disk space will be used.
Do you want to continue? [Y/n]
Preparing to unpack .../apt-transport-https_1.6.12ubuntu0.1...
Unpacking apt-transport-https (1.6.12ubuntu0.1) ...
Setting up apt-transport-https (1.6.12ubuntu0.1) ...
kepri@kepri:~$ _
```

1. Docker 에 필요한 것을 사전에 설치해 줍니다.
2. 아래 명령어를 입력합니다.
3. 필요할 경우 설치를 진행합니다.

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Docker 파일 설치 및 구성

```
Setting up apt-transport-https (1.6.12ubuntu0.1) ...  
kepri@kepri:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

1. Docker 설치를 위한 레포지토리 추가합니다.
2. 아래 명령어를 입력합니다.
3. 레포지토리를 추가합니다.

```
kepri@kepri:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"  
Get:1 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]  
Get:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [11.2 kB]  
Hit:3 http://kr.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:4 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:5 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Hit:6 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease  
Fetched 75.6 kB in 2s (45.1 kB/s)
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
  
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

Docker 파일 설치 및 구성

```
kepri@kepri:~$ sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:2 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:5 http://kr.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... 86%

kepri@kepri:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.8~3-0~ubuntu-bionic
  Version table:
   5:19.03.8~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.7~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.6~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.5~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.4~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.3~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.2~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.1~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:19.03.0~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.9~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.8~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.7~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.6~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.5~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.4~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.3~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.2~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.1~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   5:18.09.0~3-0~ubuntu-bionic 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   18.06.3~3-0~ubuntu 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   18.06.2~3-0~ubuntu 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   18.06.1~3-0~ubuntu 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   18.06.0~3-0~ubuntu 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
   18.03.1~3-0~ubuntu 500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
```

1. 레포지토리를 업데이트를 했으면 apt를 업데이트해 줍니다.
2. sudo apt update 를 입력합니다.
3. Docker가 설치 되어 있는 있는지 확인합니다.
4. apt-cache policy docker-ce 입력하여 설치되어 있는지 확인합니다.
5. Install 부분에서 None를 확인합니다.

Docker 파일 설치 및 구성

```
kepri@kepri:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-tools cgroupfs-mount containerd.io docker-ce-cli libltdl7 pigz
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount containerd.io docker-ce docker-ce-cli libltdl7 pigz
0 upgraded, 7 newly installed, 0 to remove and 31 not upgraded.
Need to get 87.0 MB of archives.
After this operation, 393 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
Get:6 http://kr.archive.ubuntu.com/ubuntu bionic/main amd64 libltdl7 amd64 2.4.6-2 [3
Get:7 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce amd64 5:
Fetched 87.0 MB in 3s (33.8 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 71428 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package aufs-tools.
Preparing to unpack .../1-aufs-tools_1%3a4.9+20170918-1ubuntu1_amd64.deb ...
Unpacking aufs-tools (1:4.9+20170918-1ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../2-cgroupfs-mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
Selecting previously unselected package containerd.io.
Processing triggers for systemd (237-3ubuntu10.38) ...
[ 2160.456348] print_req_error: I/O error, dev fd0, sector
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
kepri@kepri:~$ _
```

1. Docker 를 설치합니다. 여기 프로젝트에서는 Docker-CE 커뮤니티 버전을 설치합니다.
2. 설치 명령어는 `sudo apt install docker-ce` 입력합니다.
3. 설치진행을 위해 "Y"를 선택합니다.

Docker 파일 설치 및 구성

```
Processing triggers for libc-bin (2.30-0ubuntu1) ...
kepri@kepri:~$ sudo systemctl status docker
• docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enable
   Active: active (running) since Mon 2020-05-18 02:34:29 UTC; 1min 29s ago
     Docs: https://docs.docker.com
   Main PID: 9307 (dockerd)
     Tasks: 8
    CGroup: /system.slice/docker.service
            └─9307 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.

May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.794851156Z" level=war
May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.795042258Z" level=war
May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.795103859Z" level=war
May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.795349962Z" level=inf
May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.924013514Z" level=inf
May 18 02:34:28 kepri dockerd[9307]: time="2020-05-18T02:34:28.994012613Z" level=inf
May 18 02:34:29 kepri dockerd[9307]: time="2020-05-18T02:34:29.009370610Z" level=inf
May 18 02:34:29 kepri dockerd[9307]: time="2020-05-18T02:34:29.009585113Z" level=inf
May 18 02:34:29 kepri systemd[1]: Started Docker Application Container Engine.
May 18 02:34:29 kepri dockerd[9307]: time="2020-05-18T02:34:29.064640320Z" level=inf
lines 1-19/19 (END)
```

1. Docker 설치가 완료되었는지 확인합니다.
2. `sudo systemctl status docker` 를 입력합니다.
3. 종료를 위해 `Ctrl + c` 를 입력합니다.

Docker 파일 설치 및 구성

1. Docker에서 사용자 계정으로 실행하기 위해 다음을 입력하여 줍니다.
2. `sudo usermod -aG docker kmari`
3. 적용을 위해 재시작을 합니다
4. `reboot`을 입력합니다.
5. 재시작후 Docker 버전을 확인합니다.

```
kepri@kepri:~$ sudo usermod -aG docker kepri
kepri@kepri:~$
```

```
kepri@kepri:~$ sudo docker --version
[sudo] password for kepri:
Docker version 19.03.8, build afacb8b7f0
kepri@kepri:~$
```

Docker-Compose 설치 및 구 성 메뉴얼

Docker-Compose 설치 및 구성

```
kepri@kepri:~$ sudo pip install docker-compose
The directory '/home/kepri/.cache/pip/http' or its parent directory is not owned by the current user and the cache has b
ck the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/kepri/.cache/pip' or its parent directory is not owned by the current user and caching wheels has b
permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting docker-compose
  Downloading https://files.pythonhosted.org/packages/ec/35/1dfbb8e6b2ce5d290622a49cae0a7f3cf09cdc4341380a600aee00530881
y2.py3-none-any.whl (139kB)
    100% |#####| 143kB 432kB/s
Collecting docopt<1,>=0.6.1 (from docker-compose)
  Downloading https://files.pythonhosted.org/packages/a2/55/8f8cab2afd404cf578136ef2cc5dfb50baa1761b68c9da1fb1e4eed343c9
Requirement already satisfied: six<2,>=1.3.0 in /usr/lib/python2.7/dist-packages (from docker-compose)
Collecting backports.ssl-match-hostname<4,>=3.5; python_version < "3.5" (from docker-compose)
  Downloading https://files.pythonhosted.org/packages/ff/2b/8265224812912bc5b7a607c44bf7b027554e1b9775e9ee0de8032e3de4b2
tname-3.7.0.1.tar.gz
```

```
Found existing installation: cryptography 2.1
Not uninstalling cryptography at /usr/lib/p
Successfully installed PyYAML-5.3.1 attrs-19.3.
ed-property-1.5.1 certifi-2020.4.5.1 cffi-1.14.
-docker-compose-1.25.5 dockerpty-0.4.1 docopt-0.6.2 fu
parser-2.20 pynacl-1.3.0 pyrsistent-0.16.0 requ
7.0 zipp-1.2.0
kepri@kepri:~$ sudo docker-compose --version
docker-compose version 1.25.5, build unknown
kepri@kepri:~$
```

1. Docker-Compose를 위해 앞에서 이미 Python package Management를 설치 완료 하여 바로 Docker-Compose 설치를 진행합니다.
2. Docker-Compose 버전을 설치할 때에는 최신 버전을 항상 확인하고 설치를 진행합니다.
3. 여기서는 1.25.5 버전을 설치 진행합니다.
4. 아래 명령어를 입력하여 줍니다.
5. `sudo pip install docker-compose`
6. 설치가 완료되면 버전을 확인합니다.
7. 확인 명령어는 `sudo docker-compose --version` 입니다.

Docker-Compose 설치 및 구성

1. 디렉토리 권한을 추가합니다.
2. 명령어는 `sudo chmod +x /usr/local/bin/docker-compose` 입력합니다.



```
kepri@kepri:~$ sudo chmod +x /usr/local/bin/docker-compose
kepri@kepri:~$ _
```

Hyperledger Fabric 설치 및 구성과 테스트 메뉴얼

Hyperledger Fabric 설치 및 구성

```
kepri@ubuntumain:~$ sudo apt install git curl libltdl-dev tree openssh-server net-tools_
```

```
kepri@ubuntumain:~$ sudo apt install git curl libltdl-dev tree openssh-server net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
net-tools is already the newest version (1.60+git20161116.90da8a0-1ubuntu1).
net-tools set to manually installed.
curl is already the newest version (7.58.0-2ubuntu3.8).
git is already the newest version (1:2.17.1-1ubuntu0.7).
The following additional packages will be installed:
  autotools-dev libtool libwrap0 ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  libtool-doc autoconf automake gfortran | fortran95-compiler gcj-jdk molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  autotools-dev libltdl-dev libtool libwrap0 ncurses-term openssh-server openssh-sftp-server ssh-import-id tree
0 upgraded, 9 newly installed, 0 to remove and 40 not upgraded.
Need to get 1,119 kB of archives.
After this operation, 7,889 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

1. Hyperledger Fabric를 설치 전에 필요한 네트워크 도구를 설치합니다.

Hyperledger Fabric 설치 및 구성

```

Fetched 1,119 kB in 3s (375 kB/s)
Preconfiguring packages ...
Selecting previously unselected package autotools-dev.
(Reading database ... 107395 files and directories currently installed.)
Preparing to unpack .../0-autotools-dev_20180224.1_all.deb ...
Unpacking autotools-dev (20180224.1) ...
Selecting previously unselected package libltdl-dev:amd64.
Preparing to unpack .../1-libltdl-dev_2.4.6-2_amd64.deb ...
Unpacking libltdl-dev:amd64 (2.4.6-2) ...
Selecting previously unselected package libtool.
Preparing to unpack .../2-libtool_2.4.6-2_all.deb ...
Unpacking libtool (2.4.6-2) ...
Selecting previously unselected package libwrap0:amd64.
Preparing to unpack .../3-libwrap0_7.6.q-27_amd64.deb ...
Unpacking libwrap0:amd64 (7.6.q-27) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../4-ncurses-term_6.1-1ubuntu1.18.04_all.deb ...
Unpacking ncurses-term (6.1-1ubuntu1.18.04) ...
Selecting previously unselected package openssh-sftp-server.
Preparing to unpack .../5-openssh-sftp-server_1%3a7.6p1-4ubuntu0.3_amd64.deb ...
Unpacking openssh-sftp-server (1:7.6p1-4ubuntu0.3) ...
Selecting previously unselected package openssh-server.
Preparing to unpack .../6-openssh-server_1%3a7.6p1-4ubuntu0.3_amd64.deb ...
Unpacking openssh-server (1:7.6p1-4ubuntu0.3) ...
Selecting previously unselected package tree.
Preparing to unpack .../7-tree_1.7.0-5_amd64.deb ...
Unpacking tree (1.7.0-5) ...
Selecting previously unselected package ssh-import-id.
Setting up openssh-server (1:7.6p1-4ubuntu0.3) ...
Creating config file /etc/ssh/sshd_config with new version
Created symlink /etc/systemd/system/ssh.service → /lib/systemd/system/ssh.service.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /lib/systemd/system/ssh.service.
[ 1467.528389] print_req_error: I/O error, dev fd0, sector 0
[ 1468.016610] print_req_error: I/O error, dev fd0, sector 0
[ 1468.376704] print_req_error: I/O error, dev fd0, sector 0
Processing triggers for systemd (237-3ubuntu10.38) ...
[ 1468.864488] print_req_error: I/O error, dev fd0, sector 0
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
kepri@ubuntu:~$ _
```

상태: 실행 중

```

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
```

Progress: [98%] [#####]

1. 필요한 도구를 설치 완료하면 입니다.

Hyperledger Fabric 설치 및 구성

```
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
kepri@kepri:~$ mkdir -p $GOPATH/src/github.com/hyperledger/_
```

```
kepri@kepri:~/hlf$ ls -al
total 12
drwxrwxr-x 3 kepri kepri 4096 May 18 03:08 .
drwxr-xr-x 5 kepri kepri 4096 May 18 02:03 ..
drwxrwxr-x 3 kepri kepri 4096 May 18 03:08 src
kepri@kepri:~/hlf$
```

```
kepri@kepri:~/hlf$ ls -al
total 12
drwxrwxr-x 3 kepri kepri 4096 May 18 03:08 .
drwxr-xr-x 5 kepri kepri 4096 May 18 02:03 ..
drwxrwxr-x 3 kepri kepri 4096 May 18 03:08 src
kepri@kepri:~/hlf$ cd $GOPATH/src/github.com/hyperledger
kepri@kepri:~/hlf/src/github.com/hyperledger$
```

1. 이제 Hyperledger Fabric을 다운로드 받기 위해 디렉토리를 생성합니다.
2. 생성시 Golang 언어의 홈 디렉토리 아래에 설치하여 컴파일에 문제가 없도록 합니다.
3. 디렉터리 생성 명령어는 `mkdir -p $GOPATH/src/github.com/hyperledger/` 로 설정했습니다.
4. 경로는 hlf 디렉터리 이동 후 src 디렉터리가 생성되었는지 확인합니다.
5. 이제 해당 디렉토리로 이동합니다.
6. 명령어는 `cd $GOPATH/src/github.com/hyperledger/` 입력합니다.

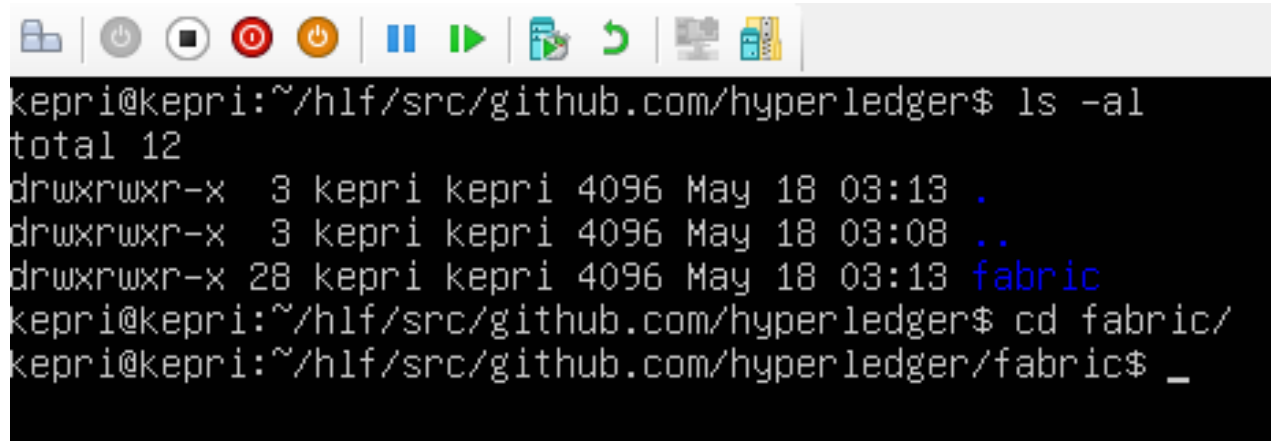
Hyperledger Fabric 설치 및 구성

```
epri@kepri:~/hlf$ ls -al
total 12
-rwxrwxr-x 3 kepri kepri 4096 May 18 03:08 .
-rwxr-xr-x 5 kepri kepri 4096 May 18 02:03 ..
-rwxrwxr-x 3 kepri kepri 4096 May 18 03:08 src
epri@kepri:~/hlf$ cd $GOPATH/src/github.com/hyperledger
epri@kepri:~/hlf/src/github.com/hyperledger$ git clone -b release-1.4 https://github.com/hyperledger/fabric
Cloning into 'fabric'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (22/22), done.
receiving objects: 0% (1/133362)
```

1. 이제 Hyperledger Fabric 를 다운로드 받습니다.
2. 버전은 주의하여야 합니다. 최신 버전은 2.1 이며 여기서는 버전을 지정하여 다운로드 받습니다.
3. 명령어는 git clone -b release-
https://github.com/hyperledger/fabric
입력합니다.

```
~/hlf/src/github.com/hyperledger
~/hlf/src/github.com/hyperledger$ git clone -b release-1.4 https://github.com/hyperledger/fabric
Cloning into 'fabric'...
```

Hyperledger Fabric 설치 및 구성



```
kepri@kepri:~/hlf/src/github.com/hyperledger$ ls -al
total 12
drwxrwxr-x  3 kepri kepri 4096 May 18 03:13 .
drwxrwxr-x  3 kepri kepri 4096 May 18 03:08 ..
drwxrwxr-x 28 kepri kepri 4096 May 18 03:13 fabric
kepri@kepri:~/hlf/src/github.com/hyperledger$ cd fabric/
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric$ _
```

1. Git에서 받은 디렉토리로 이동하고 make 를 하여 설치를 진행합니다.
2. 디렉토리 이동 명령어는 cd fabric 를 입력합니다.
3. 이동한 다음 make 입력합니다.

Hyperledger Fabric 설치 및 구성

```
ok github.com/hyperledger/fabric/protos/transaction 12.789s coverage: 86.0% of statements
? github.com/hyperledger/fabric/token [no test files]
ok github.com/hyperledger/fabric/token/client 1.439s coverage: 63.6% of statements
? github.com/hyperledger/fabric/token/client/mock [no test files]
ok github.com/hyperledger/fabric/token/identity 0.256s coverage: [no statements] [no tests to run]
? github.com/hyperledger/fabric/token/identity/mock [no test files]
? github.com/hyperledger/fabric/token/ledger [no test files]
? github.com/hyperledger/fabric/token/ledger/mock [no test files]
ok github.com/hyperledger/fabric/token/server 0.561s coverage: 87.8% of statements
? github.com/hyperledger/fabric/token/server/mock [no test files]
? github.com/hyperledger/fabric/token/tms [no test files]
ok github.com/hyperledger/fabric/token/tms/manager 0.315s coverage: 92.9% of statements
ok github.com/hyperledger/fabric/token/tms/plain 0.499s coverage: 89.1% of statements
ok github.com/hyperledger/fabric/token/transaction 0.294s coverage: 86.2% of statements
? github.com/hyperledger/fabric/token/transaction/mock [no test files]
FAIL
real 64m34.00ns
user 47m55.7
sys 2m3.014
goroutine 258 [chan send]:
testing.tRunner.func1(0xc000389440)
    /usr/local/go/src/testing/testing.go:981 +0x4d9
testing.tRunner(0xc000389440, 0x9f8b20)
    /usr/local/go/src/testing/testing.go:995 +0x20e
created by testing.(*T).Run
    /usr/local/go/src/testing/testing.go:1042 +0x661
FAIL github.com/hyperledger/fabric/bccsp/sw 17.719s
? github.com/hyperledger/fabric/bccsp/sw/mocks [no test files]
ok github.com/hyperledger/fabric/bccsp/utls 0.315s coverage: 85.8% of statements
ok github.com/hyperledger/fabric/cmd/common 0.354s coverage: 86.3% of statements
ok github.com/hyperledger/fabric/cmd/common/comm 0.062s coverage: 92.5% of statements
ok github.com/hyperledger/fabric/cmd/common/signer 0.385s coverage: 83.3% of statements
ok github.com/hyperledger/fabric/cmd/discover 27.661s coverage: 0.0% of statements
ok github.com/hyperledger/fabric/common/attrmgr 0.040s coverage: 87.9% of statements
ok github.com/hyperledger/fabric/common/capabilities 0.384s coverage: 98.6% of statements
ok github.com/hyperledger/fabric/common/cauthdsl 0.925s coverage: 95.2% of statements
ok github.com/hyperledger/fabric/common/chaincode 0.017s coverage: 100.0% of statements
ok github.com/hyperledger/fabric/common/channelconfig 19.698s coverage: 77.3% of statements
? github.com/hyperledger/fabric/common/config [no test files]
ok github.com/hyperledger/fabric/common/configtx 0.397s coverage: 85.3% of statements
? github.com/hyperledger/fabric/common/configtx/test [no test files]
ok github.com/hyperledger/fabric/common/crypto 0.371s coverage: 96.1% of statements
ok github.com/hyperledger/fabric/common/crypto/tls 1.053s coverage: 81.8% of statements
ok github.com/hyperledger/fabric/common/deliver 0.716s coverage: 97.4% of statements
? github.com/hyperledger/fabric/common/deliver/mock [no test files]
ok github.com/hyperledger/fabric/common/diag 0.032s coverage: 70.0% of statements
```

1. 설치 시간은 약 30분 전 후가 걸리며 ,
마지막에 unittest 에서 오류가 발생합
니다.
2. 이 부분에서 unit test 오류가 나오면 설
치는 정상적으로 진행된 것입니다.
3. Go Unit Test의 코드에서 문제가 있는
것이므로 Hyperledger Fabric에는 문제
없이 설치완료 된 것입니다.
4. 정상적으로 테스트가 진행되는 경우도
있습니다. 이는 Golang 버전의 호환성
을 자동으로 맞추어서 해주기 때문입니
다.

Hyperledger Fabric 설치 및 구성

```
ic$ export FABRIC_HOME=/home/keprie/hlf/src/github.com/hyperledger/fabric
ic$ export PATH=$GOPATH/bin:$GOROOT/bin:$FABRIC_HOME/.build/bin:$PATH
ic$
```



```
keprie@keprie:~$ export FABRIC_HOME=/home/hlf/src/github.com/hyperledger/fabric
keprie@keprie:~$ export PATH=$GOPATH/bin:$GOROOT/bin:$FABRIC_HOME/.build/bin:$PATH
keprie@keprie:~$ source .profile
keprie@keprie:~$ cd $FABRIC_HOME/
```

```
export FABRIC_HOME=/home/hlf/src/github.com/hyperledger/fabric
export PATH=$GOPATH/bin:$GOROOT/bin:$FABRIC_HOME/.build/bin:$PATH_
```

```
: wq
```

상태: 실행 중

1. Hyperledger Fabric의 경로를 추가합니다. 아래 명령어를 확인합니다.
2. source /profile 입력하여 적용합니다.
3. sudo vi .profile 입력합니다. 위치는 /home/kmari 있습니다.
4. 아래 항목을 입력합니다.
5. "ESC"를 입력하고 wq를 입력하여 저장합니다.

```
export FABRIC_HOME=/home/kmari/src/github.com/hyperledger/fabric
export PATH=$GOPATH/bin:$GOROOT/bin:$FABRIC_HOME/.build/bin:$PATH
```

Hyperledger Fabric 테스트 하기

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric$ git clone https://github.com/hyperledger/fabric/fabric-samples.git
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger$ git clone -b master https://github.com/hyperledger/fabric-samples.git
Cloning into 'fabric-samples'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 5204 (delta 1), reused 3 (delta 0), pack-reused 5194
Receiving objects: 100% (5204/5204), 2.46 MiB | 2.08 MiB/s, done.
Resolving deltas: 100% (2657/2657), done.
kepri@kepri:~/hlf/src/github.com/hyperledger$
```

1. 버전 0.4.0에서는 설치 후 테스트 방식이 변경되어 별도의 테스트하는 부분을 다운로드 받아서 설치해야 합니다.
2. Git에 연결해야 하기 때문에 인터넷이 필요합니다. 마스터 이미지에서는 이미 있기 때문에 필요 없습니다.
3. 다운로드 받은 위치는 Fabric 설치 디렉토리와 같은 위치에 있습니다.
4. 여기서는
/Home/kmari/hlf/src/github.com/hyperledger/fabric-samples 입니다.
5. 다운로드 받는 부분은 참고용입니다.

Hyperledger Fabric 테스트 하기

```
fabrik@fabrik-samples
kepri@kepri:~/hlf/src/github.com/hyperledger$ sudo curl -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0
[sudo] password for kepri:

Clone hyperledger/fabric-samples repo

==> Checking out v1.1.0 of hyperledger/fabric-samples
Note: checking out 'v1.1.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 1252c7a [FAB-8920] Pin fabric-samples to node.js "~1.1.0"

Pull Hyperledger Fabric binaries

==> Downloading version x86_64-1.1.0 platform specific fabric binaries
==> Downloading: https://github.com/hyperledger/fabric/releases/download/v1.1.0/hyperledger-fabr
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  654  100  654    0     0  1837      0 --:--:-- --:--:-- --:--:-- 1831
```

1. 테스트를 하기 위해 이제 바이너리 파일을 받습니다.
2. `sudo curl -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0` 을 입력합니다. 만약 관리자 비밀번호를 입력하여야 할 경우 입력합니다.
3. 다운로드 받고 바이너리 파일을 다운로드 받습니다.

Hyperledger Fabric 테스트 하기

```
kepri@kepri:~/hlf/src/github.com/hyperledger$ cd fabric-samples/_
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger$ cd fabric-samples/  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples$ _
```

1. 샘플 위치로 이동합니다.
2. 경로는 이미 앞에서 hyperledger 디렉토리 이동한 다음입니다.
3. cd fabric-samples/ 입력합니다
4. cd first-network 이동합니다.

```
kepri@kepri:~/hlf/src/github.com/hyperledger$ cd fabric-samples/  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples$ ls  
chaincode          CODE_OF_CONDUCT.md  CONTRIBUTING.md    high-throughput    MAINTAINERS.md    SECURITY.md  
chaincode-docker-devmode  CODEOWNERS          fabcar             interest_rate_swaps  off_chain_data    test-network  
ci                 commercial-paper    first-network      LICENSE             README.md  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples$ cd first-network/  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$
```

Hyperledger Fabric 테스트 하기

```
network$ sudo ./byfn.sh -m generate_
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ ./byfn.sh -m generate_
Generating certs and genesis block for with channel 'mychannel' and CLI
Continue? [Y/n] _
```

1. 테스트화면이 길기 때문에 중요 부분만 설명합니다.
2. 자체 테스트를 하기 위해서 키를 생성하고 이를 발행합니다.
3. Y를 입력하고 키를 생성합니다.
4. 만약 실행되지 않으면 Orderer의 버전이 맞지 않아서 입니다. 이는 업데이트를 통해 버전을 맞추어야 합니다.
5. 마스터 이미지에서 이미 맞추어져 있습니다.

Hyperledger Fabric 테스트 하기

```
#### Generate certificates using cryptogen tool #####
#####
+ cryptogen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
+ res=0
+ set +x

/home/kepri/hlf/src/github.com/hyperledger/fabric-samples/first-network/./bin/configtxgen
#####
##### Generating Orderer Genesis block #####
#####
+ configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block
2020-05-21 06:05:18.535 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2020-05-21 06:05:18.541 UTC [msp] getMspConfig -> INFO 002 Loading NodeOUs
2020-05-21 06:05:18.541 UTC [msp] getMspConfig -> INFO 003 Loading NodeOUs
2020-05-21 06:05:18.542 UTC [common/tools/configtxgen] doOutputBlock -> INFO 004 Generating genesis block
2020-05-21 06:05:18.543 UTC [common/tools/configtxgen] doOutputBlock -> INFO 005 Writing genesis block
+ res=0
+ set +x

#####
### Generating channel configuration transaction 'channel.tx' ###
#####
+ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID mychannel
2020-05-21 06:05:18.575 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2020-05-21 06:05:18.581 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2020-05-21 06:05:18.582 UTC [msp] getMspConfig -> INFO 003 Loading NodeOUs
2020-05-21 06:05:18.583 UTC [msp] getMspConfig -> INFO 004 Loading NodeOUs
2020-05-21 06:05:18.598 UTC [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Writing new channel tx
+ res=0
+ set +x

#####
##### Generating anchor peer update for Org1MSP #####
#####
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID mychannel -asOrg Org1MSP
2020-05-21 06:05:18.630 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2020-05-21 06:05:18.635 UTC [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2020-05-21 06:05:18.636 UTC [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
+ res=0
+ set +x

#####
##### Generating anchor peer update for Org2MSP #####
#####
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID mychannel -asOrg Org2MSP
2020-05-21 06:05:18.678 UTC [common/tools/configtxgen] main -> INFO 001 Loading configuration
2020-05-21 06:05:18.697 UTC [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2020-05-21 06:05:18.711 UTC [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
+ res=0
+ set +x
```

1. 정상적으로 표시된 화면입니다.

Hyperledger Fabric 테스트 하기

```
first-network$ sudo ./byfn.sh -m up
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo ./byfn.sh -m up
Starting with channel 'mychannel' and CLI timeout of '10' seconds and CLI delay of '3' seconds
Continue? [Y/n]
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo ./byfn.sh -m up
Starting with channel 'mychannel' and CLI timeout of '10' seconds and CLI delay of '3' seconds
Continue? [Y/n] y
proceeding ...
2020-05-21 06:07:04.316 UTC [main] main -> INFO 001 Exiting.....
LOCAL_VERSION=1.1.0
DOCKER_IMAGE_VERSION=1.1.0
Creating network "net_byfn" with the default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_peer1.org2.example.com" with default driver
Creating volume "net_peer1.org1.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_orderer.example.com" with default driver
Creating orderer.example.com ...
Creating peer1.org1.example.com ... done
Creating peer0.org1.example.com ...
Creating peer0.org2.example.com ...
Creating peer1.org2.example.com ...
```

1. ./byfn.sh up 을 입력하여 테스트를 진행합니다.
2. 정상적으로 테스트 되는지 확인합니다.

Hyperledger Fabric 테스트 하기

1. 마지막 화면을 확인합니다.

```
----- Querying on peer1.org2 on channel 'mychannel' ... -----
Attempting to Query peer1.org2 ...3 secs
+ peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
+ res=0
+ set +x

2020-05-21 06:08:12.297 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2020-05-21 06:08:12.297 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vsc
Query Result: 90
2020-05-21 06:08:26.398 UTC [main] main -> INFO 003 Exiting.....
===== Query on peer1.org2 on channel 'mychannel' is successful =====

===== All GOOD, BYFN execution completed =====

END

kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$
```

Hyperledger Fabric 테스트 하기

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo docker ps
```

CONTAINER ID	IMAGE	STATUS	PORTS	NAMES	COMMAND
8f91bd062a1e	dev-peer1.org2.example.com-myc-1.0-26c2ef32838554aac4f7ad6f100aca865e87959c9a126e86d764c8d01f8346ab	Up 57 seconds		dev-peer1.org2.example.com-myc-1.0	"chaincode -peer.add...
fb6f12ead38b	dev-peer0.org1.example.com-myc-1.0-384f11f484b9302df90b453200cfb25174305fce8f53f4e94d45ee3b6cab0ce9	Up About a minute		dev-peer0.org1.example.com-myc-1.0	"chaincode -peer.add...
1c9681da43db	dev-peer0.org2.example.com-myc-1.0-15b571b3ce849066b7ec74497da3b27e54e0df1345daff3951b94245ce09c42b	Up About a minute		dev-peer0.org2.example.com-myc-1.0	"chaincode -peer.add...
e5e70a194f26	hyperledger/fabric-tools:latest	Up 2 minutes		cli	"/bin/bash"
9c3b465708c6	hyperledger/fabric-peer:latest	Up 2 minutes	0.0.0.0:10051->7051/tcp, 0.0.0.0:10053->7053/tcp	peer1.org2.example.com	"peer node start"
3f47c9671e33	hyperledger/fabric-peer:latest	Up 2 minutes	0.0.0.0:9051->7051/tcp, 0.0.0.0:9053->7053/tcp	peer0.org2.example.com	"peer node start"
e18b412082e2	hyperledger/fabric-peer:latest	Up 2 minutes	0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp	peer0.org1.example.com	"peer node start"
426d8a47ee3f	hyperledger/fabric-peer:latest	Up 2 minutes	0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp	peer1.org1.example.com	"peer node start"
408fbd12dd73	hyperledger/fabric-orderer:latest	Up 2 minutes	0.0.0.0:7050->7050/tcp	orderer.example.com	"orderer"

1. 종료되기 전 docker 로 내역을 확인합니다. sudo docker ps 를 입력하여 확인합니다.
2. ./byfn.sh down 을 입력하여 테스트를 종료합니다.
3. Y를 입력하여 종료합니다.
4. 다시 sudo docker ps 를 입력하여 확인합니다.

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo ./byfn.sh down
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo ./byfn.sh down
```

```
Stopping with channel 'myc'
Continue? [Y/n] _
```

```
Deleted: sha256:70f6f7abe8d5183f5bd4c71d36b13eedd7431c9cfa15d2b6ddb1979c26d77f20
Deleted: sha256:bcbf91e5fdf66030ef378b96f9d2819d149b6ac6966dfd1bd712542b3d04606f
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-samples/first-network$
```

Hyperledger Fabric CA 설치 및 구성 메뉴얼

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepfabricca:~$ mkdir $GOPATH/keprinet
```

```
kepri@kepfabricca:~$ cp $FABRIC_HOME/sampleconfig/core.yaml $GOPATH/keprinet/
```

```
kepri@kepfabricca:~$ cp $FABRIC_HOME/sampleconfig/orderer.yaml $GOPATH/keprinet/  
kepri@kepfabricca:~$ _
```

1. Fabric CA의 네트워크 구축을 위한 디렉토리 설정을 합니다.
2. 네트워크 구축을 위한 디렉토리를 생성 합니다.
3. mkdir \$GOPATH/kmarinet 입력합니다.
4. 디렉토리를 생성 한 다음 설정파일을 복사합니다.
5. 첫번째 복사할 파일은 다음과 같습니다
cp
\$FABRIC_HOME/sampleconfig/core.yaml \$GOPATH/kmarinet
6. 두번째 복사할 파일은 다음과 같습니다.
cp
\$FABRIC_HOME/sampleconfig/orderer.yaml \$GOPATH/kmarinet

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepfabricca:~$ vi .profile
```

```
export FABRIC_HOME=/home/kepri/HLF/src/g1
export PATH=$GOPATH/bin:$GOROOT/bin:$FABR
export FABRIC_CFG_PATH=$GOPATH/keprinet
```

```
~
:wq
```

```
kepri@kepfabricca:~$ source .profile
kepri@kepfabricca:~$
```

1. Home의 kmari의 디렉토리의 profile에 경로를 추가합니다.
2. vi .profile 를 입력합니다.
3. Export
FABRIC_CFG_PATH=\$GOPATH/kmarine
t 를 추가합니다.
4. "ESC"를 입력 후 wq를 입력합니다.
5. 설정을 완료하기 위하여 source .profile 또는 reboot 를 입력합니다.

Hyperledger Fabric CA 설치 및 구성

도움말(H)

```
er$ git clone -b release-1.4 https://github.com/hyperledger/fabric-ca
```

1. CA 서버를 설치 및 구성하기 위하여 다음의 폴더로 이동합니다.
2. /home/kmari/hlf/src/github.com/hyperledger/ 로 이동합니다.
3. 이동 후 Git에서 다운로드 받습니다.
4. git clone -b release-
https://github.com/hyperledger/fabric-ca 입력하여 다운로드 받습니다.

```
kepri@kepri:~/hlf/src/github.com/hyperledger$ git clone -b release-1.4 https://github.com/hyperledger/fabric-ca
Cloning into 'fabric-ca'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 15326 (delta 12), reused 16 (delta 6), pack-reused 15281
Receiving objects: 100% (15326/15326), 24.99 MiB | 9.55 MiB/s, done.
Resolving deltas: 100% (8017/8017), done.
kepri@kepri:~/hlf/src/github.com/hyperledger$ _
```

Hyperledger Fabric CA 설치 및 구성

```
done.  
hyperledger$ cd fabric-ca/  
hyperledger/fabric-ca$ _
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-ca$ make fabric-ca-server  
Building fabric-ca-server in bin directory ...  
Built bin/fabric-ca-server  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-ca$
```

```
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-ca$ make fabric-ca-server  
Building fabric-ca-server in bin directory ...  
Built bin/fabric-ca-server  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-ca$ make fabric-ca-client  
Building fabric-ca-client in bin directory ...  
Built bin/fabric-ca-client  
kepri@kepri:~/hlf/src/github.com/hyperledger/fabric-ca$ _
```

1. 해당 fabric-ca 로 이동합니다.
2. Make 명령어를 이용하여 서버부분을 빌드합니다. make fabric-ca-server 입력합니다.
3. Make 명령어를 이용하여 클라이언트부분을 빌드합니다. Make fabric-ca-client 입력합니다.

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepri:~$ sudo vi .profile
```

```
export FABRIC_CA_HOME=$GOPATH/src/github.com/hyperledger/fabric-ca
export PATH=$FABRIC_CA_HOME/bin:$PATH
```

```
~
~
~
:wq
```

```
~
~
~
".profile" 36L, 1137C written
kepri@kepri:~$
```

1. profile을 수정하기 위해 vi를 이용하거나 export를 명령어를 실행합니다.
2. sudo vi .profile 입력합니다. 중요한 것은 /home/kmari 의 위치에 있는 것입니다. 이때 관리자 비밀번호를 입력 받을 수도 있습니다.
3. 아래 항목을 입력합니다.
4. "ESC"를 입력하고, wq를 입력합니다.

```
export FABRIC_CA_HOME=$GOPATH/src/github.com/hyperledger/fabric-ca
export PATH=$FABRIC_CA_HOME/bin:$PATH
```

Hyperledger Fabric CA 설치 및 구성



```
kepri@kepfabricca:~$ vi .profile
```

```
export FABRIC_CA_HOME=$GOPATH/src/github.com/hyperledger/fabr
export PATH=$FABRIC_CA_HOME/bin:$PATH
export FABRIC_CA_SERVER_HOME=/home/kepri/hlf/keprinet
~
```



```
~
".profile" 38L, 1186C written
kepri@kepfabricca:~$
Status: Running
```

1. Fabric CA 서버의 서비스를 시작합니다.
2. kmafabricca 서버에서 설정 작업을 합니다.
3. /home/kmari 위치에서 작업을 합니다.
4. vi .profile
5. 다음을 맨 아래 줄에 삽입합니다
6. export
FABRIC_CA_SERVER_HOME=/home/k
mari/hlf/kmarinet 입력합니다.
7. "ESC"를 입력하고 wq를 입력하고 저장
합니다.
8. 확실하게 하기 위해 source .profile 입
력하거나 또는 reboot를 합니다.

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepri:~$ source .profile
kepri@kepri:~$ _
```

```
kepri@kepri:~$ source .profile
kepri@kepri:~$ reboot
```

1. 설정을 적용하기 위해 source .profile을 입력합니다.
2. 또는 reboot을 합니다.
3. 설치 및 기본구성을 마칩니다.
4. 이제 각 서버와 연결을 위해 인증서 및 설정을 구성하고 각 서버에서도 연결을 작업을 진행합니다.

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepfabricca:~$ cd $FABRIC_CA_SERVER_HOME
```

```
hl/keprinet$  
hl/keprinet$ fabric-ca-server init -b admin:adminpw  
[INFO] Created default configuration file at /home/kepri/hlf/keprinet  
Initialization was successful  
hl/keprinet$ fabric-ca-server start -b admin:adminpw  
Configuration file location: /home/kepri/hlf/keprinet
```

1. Fabric CA 서버를 실행합니다.
2. 디렉토리 경로를 cd
\$FABRIC_CA_SERVER_HOME 입력하여
디렉토리를 이동합니다.
3. CA 서버를 초기화 하는 명령어인
fabric-ca-server init -b
admin:adminpw 를 입력합니다.
4. 이제 초기화를 했으면 서비스를 시작합
니다.
5. 서비스 시작 명령어는 fabric-ca-server
start -b admin:adminpw --
cfg.affiliations.allowremove --
cfg.identities.allowremove -d 를 입력
합니다.

참고사항

-b 옵션: 서버운영자 계정/패스워드 지정
--cfg 옵션: 조직/계정에 대한 추가/삭제
가능

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepororderer:~$ hostname
kepororderer
kepri@kepororderer:~$
```

```
epri@kepororderer:~$ ssh kepfabricca
epri@kepfabricca's password:
elcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-101-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Thu May 21 14:59:01 UTC 2020

System load:  0.0               Processes:            98
Usage of /:   34.9% of 39.12GB   Users logged in:     1
Memory usage: 16%              IP address for eth0: 10.0.0.100
```

```
kepri@kepororderer:~$ hostname
kepororderer
kepri@kepororderer:~$ ssh kepfabricca
kepri@kepfabricca's password:
```

```
SSH login: Thu May 21 14:59:27 2020 from 10.0.0.0
epri@kepfabricca:~$ _
```

1. 서비스가 실행되도록 그대로 둔 상태에서 새로운 터미널을 이용하여 인증서를 확인합니다.
2. 여기서는 orderer 서버에서 ca 서버로 접속합니다.
3. Orderer 서버에서 ssh kmafabricca 를 입력하여 접속합니다.

Hyperledger Fabric CA 설치 및 구성

```
et$ cd $FABRIC_CA_SERVER_HOME_
```

```
net$ cd $FABRIC_CA_SERVER_HOME  
net$ openssl x509 -text -noout -in ca-cert.pem _
```

1. CA 인증서 확인을 위해 openssl 을 이용합니다.
2. 우선 ca-cert.pem 이 있는 위치로 이동합니다.
3. cd \$FABRIC_CA_SERVER_HOME 이동합니다.
4. 명령어는 openssl x509 -text -noout -in ca-cert.pem 입니다.

Hyperledger Fabric CA 설치 및 구성

```
kepri@kepfabricca:~/hlf/keprinet$ openssl x509 -text -noout -in ca-cert.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      6c:a2:1f:1d:60:0b:e1:d8:98:23:4e:0c:2a:bc:15:22:7a:0c:69:f0
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = US, ST = North Carolina, O = Hyperledger, OU = Fabric, CN = fabric-ca-server
    Validity
      Not Before: May 21 14:49:00 2020 GMT
      Not After : May 18 14:49:00 2035 GMT
    Subject: C = US, ST = North Carolina, O = Hyperledger, OU = Fabric, CN = fabric-ca-server
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
            04:40:11:d6:ac:c5:b4:f8:b7:b0:1d:1d:98:ad:5a:
            0f:09:ed:24:7e:c2:9b:3c:8e:cd:ac:1a:1f:b6:79:
            40:bd:3f:93:1b:7d:ce:87:98:cc:1a:37:c9:bc:04:
            3c:84:22:e0:8a:fb:5e:6a:44:ae:d6:46:a1:dd:48:
            e4:e3:e2:95:3c
        ASN1 OID: prime256v1
        NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Basic Constraints: critical
        CA:TRUE, pathlen:1
      X509v3 Subject Key Identifier:
        A8:62:64:87:62:CD:53:D8:72:36:BA:64:57:21:6C:22:C8:06:A2:F9
    Signature Algorithm: ecdsa-with-SHA256
      30:44:02:20:29:f8:78:34:3b:d7:d6:a0:0a:2c:9d:2f:17:a1:
      00:83:ff:95:52:74:3b:1e:f2:96:f3:90:d8:8c:95:57:f5:f1:
      02:20:15:19:10:76:ae:c0:a8:65:40:1d:7c:bf:13:94:6d:84:
      97:f0:ce:68:96:f1:57:fa:64:fc:bd:1d:e2:d5:cc:76
kepri@kepfabricca:~/hlf/keprinet$
```

1. 인증서를 확인합니다.
2. 이제 orderer 및 peer 에서 설정을 이어
서 합니다.

Hyperledger Fabric Orderer 설치 및 구성 메뉴얼

Hyperledger Fabric Orderer 설치 및 구성

```
export FABRIC_CA_CLIENT_HOME=/home/kepri/hlf/keprinet  
~  
export FABRIC_CA_CLIENT_HOME=/home/kepri/hlf/keprinet
```

```
~  
:wq
```

```
kepri@kepororderer:~$ source .profile  
kepri@kepororderer:~$ reboot_
```

1. Fabric CA 서버 운영자의 MSP를 생성하는 준비를 합니다.
2. Orderer 노드에서 실행합니다.
3. Home의 kmari의 디렉토리의 profile에 경로를 추가합니다.
4. vi .profile 를 입력합니다.
5. Export
FABRIC_ca_CLIENT_HOME=/home/kmari/hlf/kmarinet 를 추가합니다.
6. "ESC"를 입력 후 wq를 입력합니다.
7. 설정을 완료하기 위하여 source .profile 또는 reboot 를 입력합니다.

Hyperledger Fabric Orderer 설치 및 구성

```
kepri@kepororderer:~$ cd $FABRIC_CA_CLIENT_HOME
```

```
fabric-ca-client enroll -u http://admin:adminpw@10.0.0.100:7054
```

1. Fabric CA 서버 구동할 때 지정한 운영자 계정/비밀번호를 이용하여 MSP 생성을 요청합니다.
2. 입력할 내용은 fabric-ca-client enroll -u http://admin:adminpw@10.0.1.100:7054 입력합니다.

```
[INFO] generating key: &{A:ecdsa S:256}
[INFO] encoded CSR
[INFO] Stored client certificate at /home/kepri/hlf/keprinet/msp/signcerts/cert.pem
[INFO] Stored root CA certificate at /home/kepri/hlf/keprinet/msp/cacerts/10-0-0-100-7054.pem
[INFO] Stored Issuer public key at /home/kepri/hlf/keprinet/msp/IssuerPublicKey
[INFO] Stored Issuer revocation public key at /home/kepri/hlf/keprinet/msp/IssuerRevocationPublicKey
lf/keprinet$ _
```

Hyperledger Fabric Orderer 설치 및 구성

```
Listening on http://0.0.0.0:7054
[INFO] signed certificate with serial number 13
10.0.0.11:47394 POST /enroll 201 0 "OK"
```

```
kepri@kepororderer:~/hlf/keprinet$ tree ./
./
├── core.yaml
├── fabric-ca-client-config.yaml
├── msp
│   ├── cacerts
│   │   └── 10-0-0-100-7054.pem
│   ├── IssuerPublicKey
│   ├── IssuerRevocationPublicKey
│   └── keystore
│       ├── 181c66f4b96a36d1c045ae167894ee62d29f85e847e01b10b434f67f2239991a_sk
│       ├── 1a7e6d77ddcb4e69b1e67fddf5c5e98a6f52bc9005ddfeb01ad05dcf7298ecda_sk
│       ├── 1d1720a2b0207d402072321f59a6614ff86db4de86ebd12d0a765de2061e56fa_sk
│       ├── 27816b77a928dcf732e16721174e1298a8b45e277d20bbefb018986139ebb1c0_sk
│       ├── 4b336dd9269534da5c595d099d592b361460593ad7c087eac37bc3b75f26f0d6_sk
│       ├── 57307a5b1edf4054eee7bdf0b4a67ff5a2e202f516179d0ec4c74dd5fd0bce3c_sk
│       ├── 5b324097adceecae1a8197993438eb978215d3879f158ae5f1f18958559046d7_sk
│       ├── 93dfd0fb7d09508c62642660eb68af1f877616255c8f9034b8870758ab12cca2_sk
│       ├── 9fe787b935732eb08c490808bf658dad0bd30abb14db2a8046671371063a56b4_sk
│       ├── b1085c675211a395d09fbe4c52f65ce8be7a52cfbe8f2727406613791e7c987d_sk
│       ├── c2d0db3bda41147262332ed9360c32009b1143d4aa77064aecb065b31ab0062e_sk
│       ├── e95a71e0d313a58c9da465951ec468c00c2306b8ad0c512a1ea21bc3720d6dd1_sk
│       └── ff4f9d1d8b540c07c62694c4450ad30606d01d8dd504532ccf8ca3eb44d7f620_sk
│   ├── signcerts
│   │   └── cert.pem
│   └── user
└── orderer.yaml
```

1. Fabric 서버에서 확인을 해 봅니다.
2. Orderer 에서 tree ./ 명령어로 구조를 확인합니다. Keysotre 부분은 생성할 때 마다 다를 수 있습니다.
또한 키의 생성 개수도 다를 수 있습니다.

Hyperledger Fabric Orderer 설치 및 구성

```
fabric-ca-client affiliation list
```

```
affiliation: .  
  affiliation: org2  
    affiliation: org2.department1  
  affiliation: org1  
    affiliation: org1.department1  
    affiliation: org1.department2  
kepri@kepororderer:~/hlf/keprinet$ _
```

1. Hyperledger Fabric의 기본 조직을 사용하지 않고 새로운 조직을 생성 및 조직 운영자의 MSP를 생성을 합니다.
2. Orderer 에서 수행합니다.
3. Fabric-ca 기본 affiliation을 삭제합니다.
4. fabric-ca-client affiliation list 를 입력합니다.
5. 결과를 확인하고 기본 조직을 삭제합니다.

Hyperledger Fabric Orderer 설치 및 구성

```
tl
fabric-ca-client affiliation remove --force org1
uration file location: /home/kenri/hlf/kenrinet/fa

ties:[]}] Identities:[]} CAName:}
$ fabric-ca-client affiliation remove --force org2
uration file location: /home/kenri/hlf/kenrinet/fa
```

1. 조직 삭제 명령은 fabric-ca-client affiliation remove --force 조직명 을 입력합니다.
2. 여기서는 org2와 org1 입니다.
3. 이제 kmari의 테스트 베드에서 설계한 조직을 생성합니다.

Hyperledger Fabric Orderer 설치 및 구성

```
$ fabric-ca-client affiliation add kepcoorg1
Configuration file location: /home/kepri/hlf/keprinet/fabric-ca-client-config.yaml
kepcoorg1
$ fabric-ca-client affiliation add kepcoorg2
Configuration file location: /home/kepri/hlf/keprinet/fabric-ca-client-config.yaml
kepcoorg2
$ fabric-ca-client affiliation add kepcoorg3
Configuration file location: /home/kepri/hlf/keprinet/fabric-ca-client-config.yaml
kepcoorg3
$ fabric-ca-client affiliation add kepcoordererorg
Configuration file location: /home/kepri/hlf/keprinet/fabric-ca-client-config.yaml
kepcoordererorg
```

1. 조직은 구성도를 보면서 생성합니다.
2. kmari_arch.pptx 파일에서 블록체인 구성도 부분을 참고합니다.
3. kmacoorg1 / kmacoorg2 / kmacoorg3 / kmacoordererorg 입니다.
Workgroup은 Fabric 네트워크의 조직과 다르기 때문에 여기에서는 넣지 않습니다.
4. 명령어는 fabric-ca-client affiliation add 조직명 을 입력합니다.

여기부터는 결과 부분만 이미지 확인을 하겠습니다.

이유 : 화면 캡처에 화면이 너무 작아서 확대해야 하기 때문입니다.

작업하면서 중요한 것은 경로명의 오타와 Pa\$\$w0rd 와 같은 몇가지 주의 사항만 조심합니다.

Hyperledger Fabric peer 설치 및 구성

kmacoorg1 조직의 MSP를 저장할 디렉토리 생성

```
mkdir -p /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp
```

kmacoorg1 조직의 CA 인증서 획득

```
fabric-ca-client getcacert -u http://10.0.0.100:7054 -M  
/home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp
```

CA 인증서 파일 이름 변경

```
mv /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/cacerts/10-0-0-100-  
7054.pem /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/cacerts/ca.crt
```

1. Fabric CA 인증서를 각각의 조직의 admin 노드에 다운로드 하여 설정합니다. 즉 kmacoorg1 조직의 kmaorg1peer01/ kmacoorg2 조직의 kmaorg2peer01/ kmacoorg3 조직의 kmaorg3peer01 에서 각각 작업을 합니다. 총 3개의 peer 에서 작업을 똑같이 진행합니다.

Hyperledger Fabric peer 설치 및 구성

kmacoorg2 조직의 MSP를 저장할 디렉토리 생성

```
mkdir -p /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp
```

kmacoorg2 조직의 CA 인증서 획득

```
fabric-ca-client getcacert -u http://10.0.0.100:7054 -M  
/home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp
```

CA 인증서 파일 이름 변경

```
mv /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/cacerts/10-0-0-100-  
7054.pem /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/cacerts/ca.crt
```

1. Fabric CA 인증서를 각각의 조직의 admin 노드에 다운로드 하여 설정합니다. 즉 kmacoorg1 조직의 kmaorg1peer01/ kmacoorg2 조직의 kmaorg2peer01/ kmacoorg3 조직의 kmaorg3peer01 에서 각각 작업을 합니다. 총 3개의 peer 에서 작업을 똑같이 진행합니다.

Hyperledger Fabric peer 설치 및 구성

kmacoorg3 조직의 MSP를 저장할 디렉토리 생성

```
mkdir -p /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp
```

kmacoorg3 조직의 CA 인증서 획득

```
fabric-ca-client getcacert -u http://10.0.0.100:7054 -M  
/home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp
```

CA 인증서 파일 이름 변경

```
mv /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/cacerts/10-0-0-100-  
7054.pem /home/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/cacerts/ca.crt
```

1. Fabric CA 인증서를 각각의 조직의 admin 노드에 다운로드 하여 설정합니다. 즉 kmacoorg1 조직의 kmaorg1peer01/ kmacoorg2 조직의 kmaorg2peer01/ kmacoorg3 조직의 kmaorg3peer01 에서 각각 작업을 합니다. 총 3개의 peer 에서 작업을 똑같이 진행합니다.

Hyperledger Fabric peer 설치 및 구성

```
# kmacordererorg 조직의 MSP를 저장할 디렉토리 생성
mkdir -p /home/hlf/kmarinet/crypto-
config/peerOrganizations/kmacordererorg/msp
```

```
# kmacordererorg 조직의 CA 인증서 획득
fabric-ca-client getcacert -u http://10.0.0.100:7054 -M
/home/hlf/kmarinet/crypto-
config/peerOrganizations/kmacordererorg/msp
```

```
# CA 인증서 파일 이름 변경
mv /home/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/msp/cacerts/10-0-0-100-
7054.pem /home/hlf/kmarinet/crypto-
config/peerOrganizations/ kmacordererorg
/msp/cacerts/ca.crt
```

1. Fabric CA 인증서를 각각의 조직의 admin 노드에 다운로드 하여 설정하는 곳 중 kmacordererorg 조직에 하기 위해서 kmaorderer 서버에서 작업합니다.

Hyperledger Fabric 각 조직의 운영자 계정 등록

\$FABRIC_CA_CLIENT_HOME/fabric-ca-client-config.yaml 파일에서 ID 부분 수정합니다.

vi fabric-ca-client-config.yaml
입력하여

```
id:  
  name: admin@kmacoorg1  
  type: client  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:
```

위의 ID 부분을 수정합니다.
그 다음 계정 생성을 위한 작업을 합니다.

fabirc-ca-client register -id.secret=adminpw

입력하여 계정 생성을 등록합니다.

1. CA 서버 운영자 권한을 가지는 kmacoorg1 조직의 kmacoorg1peer01 서버에서 작업합니다.

Hyperledger Fabric 각 조직의 운영자 계정 등록

\$FABRIC_CA_CLIENT_HOME/fabric-ca-client-config.yaml 파일에서 ID 부분 수정합니다.

vi fabric-ca-client-config.yaml
입력하여

```
id:  
  name: admin@kmacoorg2  
  type: client  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:
```

위의 ID 부분을 수정합니다.
그 다음 계정 생성을 위한 작업을 합니다.

fabirc-ca-client register -id.secret=adminpw

입력하여 계정 생성을 등록합니다.

1. CA 서버 운영자 권한을 가지는 kmacoorg2 조직의 kmacoorg2peer01 서버에서 작업합니다.

\$FABRIC_CA_CLIENT_HOME/fabric-ca-client-config.yaml 파일에서 ID 부분 수정합니다.

vi fabric-ca-client-config.yaml
입력하여

```
id:  
  name: admin@kmacoorg3  
  type: client  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:
```

위의 ID 부분을 수정합니다.
그 다음 계정 생성을 위한 작업을 합니다.

fabirc-ca-client register -id.secret=adminpw

입력하여 계정 생성을 등록합니다.

1. CA 서버 운영자 권한을 가지는 kmacoorg3 조직의 kmacoorg3peer01 서버에서 작업합니다.

Hyperledger Fabric 각 조직의 운영자 계정 등록

\$FABRIC_CA_CLIENT_HOME/fabric-ca-client-config.yaml 파일에서 ID 부분 수정합니다.

```
vi fabric-ca-client-config.yaml  
입력하여
```

```
id:  
  name: admin@kmacoorderorg  
  type: client  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:
```

위의 ID 부분을 수정합니다.
그 다음 계정 생성을 위한 작업을 합니다.

```
fabirc-ca-client register -id.secret=adminpw
```

입력하여 계정 생성을 등록합니다.

1. CA 서버 운영자 권한을 가지는 kmacoorderorg 조직의 kmaorderer 서버에서 작업합니다.

Hyperledger Fabric 조직운영자의 MSP 생성

MSP 저장 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/
```

MSP 생성 요청

fabric-ca-client enroll -u

```
http://admin@kmacoorg1:adminpw@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/
```

생성된 MSP의 ca 인증서 파일 이름 변경

```
mv /home/kmari/hlf/testnet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/msp/cacerts/10-  
0-0-100-7054.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/msp/cacerts/ca.c  
rt
```

생성된 MSP의 개인키 파일 이름 변경

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmaroorg1/users/admin@kmacoorg1/msp/keystore/6a  
785040dc00bbe895cfbe7ce97b4778ce2c19c2b0500783e597813c4bae0e39 sk /ho  
me/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/msp/keystore/ser  
verkey
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg1의 kmacoorg1peer01 에서 작업합니다.
3. 빨간색 부분의 개인키는 복사하거나 화면 캡처등을 사용하여 별도 저장하여야 합니다. "키 값은 할 때 마다 다르기 때문에 주의하여 복사합니다 "

Hyperledger Fabric admmlncerts 디렉터리 생성 및 운영자 인증서 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp/admincerts
```

```
cp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp/signcerts/cert.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp/admincerts/admin@kmacoorg1-cert.pem
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg1 의 kmacoorg1peer01에서 작업합니다.

Hyperledger Fabric 조직운영자의 MSP 생성

MSP 저장 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg2/
```

MSP 생성 요청

fabric-ca-client enroll -u

```
http://admin@kmacoorg2:adminpw@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg2/
```

생성된 MSP의 ca 인증서 파일 이름 변경

```
mv /home/kmari/hlf/testnet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg2/msp/cacerts/10-  
0-0-100-7054.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg2/msp/cacerts/ca.c  
rt
```

생성된 MSP의 개인키 파일 이름 변경

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmaroorg1/users/admin@kmacoorg2/msp/keystore/6a  
785040dc00bbe895cfbe7ce97b4778ce2c19c2b0500783e597813c4bae0e39 sk /ho  
me/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2/msp/keystore/ser  
verkey
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg2의 kmacoorg2peer01 에서 작업합니다.
3. 빨간색 부분의 개인키는 복사하거나 화면 캡처등을 사용하여 별도 저장하여야 합니다. "키 값은 할 때 마다 다르기 때문에 주의하여 복사합니다"

Hyperledger Fabric admmlncerts 디렉터리 생성 및 운영자 인증서 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp/admincerts
```

```
cp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp/signcerts/cert.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp/admincerts/admin@kmacoorg2-cert.pem
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg2 의 kmacoorg2peer01에서 작업합니다.

Hyperledger Fabric 조직운영자의 MSP 생성

MSP 저장 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/
```

MSP 생성 요청

fabric-ca-client enroll -u

```
http://admin@kmacoorg3:adminpw@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/
```

생성된 MSP의 ca 인증서 파일 이름 변경

```
mv /home/kmari/hlf/testnet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/msp/cacerts/10-  
0-0-100-7054.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/msp/cacerts/ca.c  
rt
```

생성된 MSP의 개인키 파일 이름 변경

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmaroorg3/users/admin@kmacoorg3/msp/keystore/6a  
785040dc00bbe895cfbe7ce97b4778ce2c19c2b0500783e597813c4bae0e39 sk /ho  
me/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/msp/keystore/ser  
verkey
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg3의 kmacoorg3peer01 에서 작업합니다.
3. 빨간색 부분의 개인키는 복사하거나 화면 캡처등을 사용하여 별도 저장하여야 합니다. "키 값은 할 때 마다 다르기 때문에 주의하여 복사합니다 "

Hyperledger Fabric admmlncerts 디렉터리 생성 및 운영자 인증서 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp/admincerts
```

```
cp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp/signcerts/cert.pem /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp/admincerts/admin@kmacoorg3-cert.pem
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoorg3 의 kmacoorg3peer01에서 작업합니다.

Hyperledger Fabric 조직운영자의 MSP 생성

```
# MSP 저장 디렉토리 생성
mkdir -p /home/kmari/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/

# MSP 생성 요청
fabric-ca-client enroll -u
http://admin@kmacoorderer:adminpw@10.0.0.100:7054 -H
/home/kmari/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/

# 생성된 MSP의 ca 인증서 파일 이름 변경
mv /home/kmari/hlf/testnet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/msp/cac
erts/10-0-0-100-7054.pem /home/kmari/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/msp/cac
erts/ca.crt

# 생성된 MSP의 개인키 파일 이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/msp/key
store/6a785040dc00bbe895cfbe7ce97b4778ce2c19c2b0500783e597813c4bae0e3
9_sk /home/kmari/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/users/admin@kmacoorderer/msp/key
store/server.key
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoordererorg의 kmaorderer 에서 작업합니다.
3. 빨간색 부분의 개인키는 복사하거나 화면 캡처등을 사용하여 별도 저장하여야 합니다. "키 값은 할 때 마다 다르기 때문에 주의하여 복사합니다 "

Hyperledger Fabric admmlncerts 디렉터리 생성 및 운영자 인증서 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizations/kmacoordererorg/users/admin@k  
macoordererorg/msp/admincerts
```

```
cp /home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizations/kmacoordererorg/users/admin@k  
macoorg1/msp/signcerts/cert.pem  
/home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizations/kmacoordererorg/users/admin@k  
macoordererorg/msp/admincerts/admin@kmacoordererorg-  
cert.pem
```

1. 4개의 서버에서 작업한 계정에 대한 각 조직의 운영자 노드의 MSP를 생성합니다.
2. kmacoordererorg 의 kmaorderer에서 작업합니다.

Hyperledger Fabric Peer 및 Orderer 노드 계정 작업

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/fabric-  
ca-client-config.yaml
```

파일의 id 수정

```
id:  
  name: kmaorg1peer01  
  type: peer  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:
```

확인 다음 작업 입니다.

```
fabric-ca-client register --id.secret=adminpw -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/
```

1. 조직의 운영자에서 Peer 노드의 등록입니다.
2. kmacoorg1 의 kmacoorg1peer01에서 작업합니다.
3. 여기서는 서버 이름 및 피어 이름을 목록에서 확인 후 변경합니다.
4. -H 옵션으로 fabric-ca-client-config.yaml 파일 경로 지정하여 CA 클라이언트 등록합니다.

Hyperledger Fabric Peer 및 Orderer 노드 계정 작업

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2/fabric-  
ca-client-config.yaml
```

파일의 id 수정

```
id:  
  name: kmaorg2peer01  
  type: peer  
  affiliation: kmacoorg2  
  maxenrollments: -1  
  attributes:
```

확인 다음 작업 입니다.

```
fabric-ca-client register --id.secret=adminpw -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2/
```

1. 조직의 운영자에서 Peer 노드의 등록입니다.
2. kmacoorg2 의 kmacoorg2peer01에서 작업합니다.
3. 여기서는 서버 이름 및 피어 이름을 목록에서 확인 후 변경합니다.
4. -H 옵션으로 fabric-ca-client-config.yaml 파일 경로 지정하여 CA 클라이언트 등록합니다.

Hyperledger Fabric Peer 및 Orderer 노드 계정 작업

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg3/fabric-  
ca-client-config.yaml
```

파일의 id 수정

```
id:  
  name: kmaorg3peer01  
  type: peer  
  affiliation: kmacoorg3  
  maxenrollments: -1  
  attributes:
```

확인 다음 작업 합니다.

```
fabric-ca-client register --id.secret=adminpw -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/
```

1. 조직의 운영자에서 Peer 노드의 등록입니다.
2. kmacoorg3 의 kmacoorg3peer01에서 작업합니다.
3. 여기서는 서버 이름 및 피어 이름을 목록에서 확인 후 변경합니다.
4. -H 옵션으로 fabric-ca-client-config.yaml 파일 경로 지정하여 CA 클라이언트 등록합니다.

Hyperledger Fabric Peer 및 Orderer 노드 계정 작업

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/keocpordererorg/users/admin@kmacoordererorg/fabric-ca-client-config.yaml
```

파일의 id 수정

```
id:  
  name: kmaorderer  
  type: orderer  
  affiliation: kmacoordererorg  
  maxenrollments: -1  
  attributes:
```

확인 다음 작업 합니다.

```
fabric-ca-client register --id.secret=adminpw -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoordererorg/users/admin@kmacoordererorg/
```

1. 조직의 운영자에서 Peer 노드의 등록입니다.
2. kmacoordererorg 의 kmaorderer에서 작업합니다.
3. 여기서는 서버 이름 및 피어 이름을 목록에서 확인 후 변경합니다.
4. -H 옵션으로 fabric-ca-client-config.yaml 파일 경로 지정하여 CA 클라이언트 등록합니다.

Hyperledger Fabric Peer 및 Orderer 노드 MSP 생성 작업

```
# MSP를 저장할 디렉토리 생성
mkdir -p /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer01.kmacoorg1/

# MSP 요청
fabric-ca-client enroll -u http://kmaco1peer01:adminpw@10.0.0.100:7054 -H
/home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer01.kmacoorg1/

# ca인증서 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmacoorg1peer01.kmacoorg1/msp/ca
certs/10-0-0-100-7054.pem /home/kmair/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmacorog1peer01.kmacoorg1/msp/ca
certs/ca.crt

# 개인키 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmariorg1/peers/kmacoorg1peer01.kmacoorg1/msp/key
store/ef54938c016be68b046eb90374634ad4f93c22dfd8a8ff6c1fe1d5817de01d25_s
k /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmaco1peer01.kmacoorg1/msp/keyst
ore/server.key
```

1. MSP 생성을 위한 작업을 합니다.
2. kmacoorg1 의 kmacoorg1peer01에서 작업합니다.
3. 앞에서 작업할 때 개인키 복사한 부분을 다시 확인하고 똑같이 넣어야 합니다.
4. 개인키의 값이 틀리면 서비스도 작동하지 않습니다.

Hyperledger Fabric Peer 및 Orderer 노드 MSP 생성 작업

```
# MSP를 저장할 디렉토리 생성
mkdir -p /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer01.kmacoorg2/

# MSP 요청
fabric-ca-client enroll -u http://kmaco2peer01:adminpw@10.0.0.100:7054 -H
/home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer01.kmacoorg2/

# ca인증서 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmacoorg2peer01.kmacoorg2/msp/ca
certs/10-0-0-100-7054.pem /home/kmair/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmacorog2peer01.kmacoorg2/msp/ca
certs/ca.crt

# 개인키 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmariorg2/peers/kmacoorg2peer01.kmacoorg2/msp/key
store/ef54938c016be68b046eb90374634ad4f93c22dfd8a8ff6c1fe1d5817de01d25_s
k /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmaco2peer01.kmacoorg2/msp/keyst
ore/server.key
```

1. MSP 생성을 위한 작업을 합니다.
2. kmacoorg2 의 kmacoorg2peer01에서 작업합니다.
3. 앞에서 작업할 때 개인키 복사한 부분을 다시 확인하고 똑같이 넣어야 합니다.
4. 개인키의 값이 틀리면 서비스도 작동하지 않습니다.

Hyperledger Fabric Peer 및 Orderer 노드 MSP 생성 작업

```
# MSP를 저장할 디렉토리 생성
mkdir -p /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer01.kmacoorg1/

# MSP 요청
fabric-ca-client enroll -u http://kmaco3peer01:adminpw@10.0.0.100:7054 -H
/home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer01.kmacoorg1/

# ca인증서 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmacoorg3peer01.kmacoorg1/msp/ca
certs/10-0-0-100-7054.pem /home/kmair/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmacorog3peer01.kmacoorg1/msp/ca
certs/ca.crt

# 개인키 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmariorg3/peers/kmacoorg3peer01.kmacoorg3/msp/key
store/ef54938c016be68b046eb90374634ad4f93c22dfd8a8ff6c1fe1d5817de01d25_s
k /home/kmar/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaco3peer01.kmacoorg3/msp/keyst
ore/server.key
```

1. MSP 생성을 위한 작업을 합니다.
2. kmacoorg3 의 kmacoorg3peer01에서 작업합니다.
3. 앞에서 작업할 때 개인키 복사한 부분을 다시 확인하고 똑같이 넣어야 합니다.
4. 개인키의 값이 틀리면 서비스도 작동하지 않습니다.

Hyperledger Fabric Peer 및 Orderer 노드 MSP 생성 작업

```
# MSP를 저장할 디렉토리 생성
mkdir -p /home/kmar/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/orderers/kmacoorderer. kmacoorderer
/

# MSP 요청
fabric-ca-client enroll -u http://kmaco1peer01:adminpw@10.0.0.100:7054 -H
/home/kmari/hlf/kmarinet/crypto-config/ordererOrganizations/ kmacoorderer
/orderers/ kmacoorderer. kmacoorderer /

# ca인증서 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-config/ordererOrganizations/ kmacoorderer
/peers/kmacoorg1peer01. kmacoorderer /msp/cacerts/10-0-0-100-7054.pem
/home/kmair/hlf/kmarinet/crypto-
config/ordererOrganizations/kmacoorderer/orderer/kmacoorderer.
kmacoordererorg /msp/cacerts/ca.crt

# 개인키 파일이름 변경
mv /home/kmari/hlf/kmarinet/crypto-config/peerOrganizations/ kmacoorderer
/orderers/kmacoorderer. kmacoordererorg
/msp/keystore/ef54938c016be68b046eb90374634ad4f93c22dfd8a8ff6c1fe1d5817d
e01d25_sk /home/kmar/hlf/kmarinet/crypto-config/ordererOrganizations/
kmacoorderer /orderer/keororderer. kmacoordererorg /msp/keystore/server.key
```

1. MSP 생성을 위한 작업을 합니다.
2. kmacoordererorg 의 kmacoorderer에
서 작업합니다.
3. 앞에서 작업할 때 개인키 복사한 부분
을 다시 확인하고 똑같이 넣어야 합니
다.
4. 개인키의 값이 틀리면 서비스도 작동하
지 않습니다.

Hyperledger Fabric Orderer 서비스 시작

```
vi home/kmari/hlf/kmarinet/configtx.yaml
```

파일 생성
주요 내용은 다음과 같습니다.

1. Orderer 시작을 위해 트랜잭션 생성을 위한 설정파일을 생성합니다.
2. Yaml 파일이기 때문에 간격 및 tab 을 주의해야 하며 잘못할 경우 서비스가 작동하지 않습니다.

Hyperledger Fabric Orderer 서비스 시작

```
#####  
#####  
#  
# ORGANIZATIONS  
#  
# This section defines the organizational identities that can be  
# referenced  
# in the configuration profiles.  
#  
#####  
#####  
Organizations:  
  
- &kmacordererorg  
  Name: kmacordererorg  
  ID: keocpordererorgMSP  
  MSPDir: crypto-  
config/ordererOrganizations/kmacordererorg/msp/
```

1. 앞부분 이어서 작업합니다.

```
- &kmacorog1
  Name: kmacoorg1MSP
  ID: kmacoorg1MSP
  MSPDir: crypto-
config/peerOrganizations/kmacoorg1/msp/
AnchorPeers:
  - Host: kmaorg1peer01
    Port: 7051

- &kmacorog2
  Name: kmacooorg2MSP
  ID: kmacoorg2MSP
  MSPDir: crypto-
config/peerOrganizations/kmacoorg2/msp/
AnchorPeers:
  - Host: kmaorg2peer01
    Port: 7051
```

1. 앞부분 이어서 작업합니다.

```
- &kmacorog3
  Name: kmacoorg3MSP
  ID: kmacoorg3MSP
  MSPDir: crypto-
config/peerOrganizations/kmacoorg3/msp/
  AnchorPeers:
    - Host: kmaorg3peer01
      Port: 7051
```

1. 앞부분 이어서 작업합니다.

Hyperledger Fabric Orderer 서비스 시작

```
#####  
#####  
#  
# ORDERER  
#  
# This section defines the values to encode into a config  
# transaction or  
# genesis block for orderer related parameters.  
#  
#####  
#####  
Orderer: &OrdererDefaults
```

1. 앞부분 이어서 작업합니다.

Hyperledger Fabric Orderer 서비스 시작

OrdererType: kafka

Addresses:

- orderer:7050

BatchTimeout: 1s

BatchSize:

MaxMessageCount: 30

AbsoluteMaxBytes: 99 MB

PreferredMaxBytes: 512 KB

MaxChannels: 0

Kafka:

Brokers:

- kmakafka:9092

Organizations:

1. 앞부분 이어서 작업합니다.

Hyperledger Fabric Orderer 서비스 시작

```
#####  
#####  
#  
# APPLICATION  
#  
# This section defines the values to encode into a config  
# transaction or  
# genesis block for application-related parameters.  
#  
#####  
#####  
Application: &ApplicationDefaults  
  
# Organizations lists the orgs participating on the  
# application side of the  
# network.  
Organizations:
```

1. 앞부분 이어서 작업합니다.

Hyperledger Fabric Orderer 서비스 시작

```
#####  
#####  
#  
#  PROFILES  
#  
#  Different configuration profiles may be encoded here to be  
#  specified as  
#  parameters to the configtxgen tool. The profiles which  
#  specify consortiums  
#  are to be used for generating the orderer genesis block.  
#  With the correct  
#  consortium members defined in the orderer genesis block,  
#  channel creation  
#  requests may be generated with only the org member  
#  names and a consortium  
#  name.  
#  
#####  
#####  
Profiles:
```

1. 앞부분 이어서 작업합니다.

TwoOrgsOrdererGenesis:

Orderer:

<<: *OrdererDefaults

Organizations:

- *kmacoordererorg

Consortiums:

SampleConsortium:

Organizations:

- *kmacoorg1

- *kmacoorg2

- *kmacoorg3

1. 앞부분 이어서 작업합니다.

```
TwoOrgsChannel:  
  Consortium: SampleConsortium  
  Application:  
    <<: *ApplicationDefaults  
    Organizations:  
      - *kmacoorg1  
      - *kmacoorg2  
      - *kmacoorg3
```

1. 앞부분 이어서 작업합니다.

Hyperledger Fabric gensis.block 생성 및 인증서 기관의 MSP 복사

```
# OrdererOrg 기관의 MSP 디렉토리 내에 운영자의 인증서 디렉토리 생성
```

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-config/ordererOrganizations/ordererorg/msp/admincerts/
```

```
# Kpeco Orderer Org 기관의 운영자 인증서 복사
```

```
cp /home/kmari/hlf/kmarinet/crypto-config/ordererOrganizations/ordererorg/users/admin@kmacordererorg/msp/admincerts/admin@kmacordererorg-cert.pem  
/home/kmari/hlf/kmarinet/crypto-config/ordererOrganizations/kmacordererorg/msp/admincerts/
```

1. kmaorderer 노드에서 복사작업을 진행합니다.
2. orderer 노드에는 orderer 노드와 ordererorg 관리자 의 MSP를 모두 포함하고 있습니다.

Hyperledger Fabric gensis.block 생성 및 인증서 기관의 MSP 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/admincerts/  
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/cacerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp/admincerts/admin@kmacoorg1-cert.pem  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/admincerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/cacerts/ca.crt  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/msp/cacerts/
```

1. kmacoorg1 에서 작업을 진행합니다.
2. kmacoorg1의 admin에서 공개키 복사합니다.
3. kmacoorg1 조직의 운영자인증서와 CA 인증서를 저장할 디렉토리 생성합니다.
4. 이제 kmacoorg1에서 복사를 진행합니다.

Hyperledger Fabric genesis.block 생성 및 인증서 기관의 MSP 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/admincerts/  
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/cacerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp/admincerts/admin@kmacoorg2-cert.pem  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/admincerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/cacerts/ca.crt  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/msp/cacerts/
```

1. kmacoorg2 에서 작업을 진행합니다.
2. kmacoorg2의 admin에서 공개키 복사합니다.
3. kmacoorg2 조직의 운영자인증서와 CA 인증서를 저장할 디렉토리 생성합니다.
4. 이제 kmacoorg2에서 복사를 진행합니다.

Hyperledger Fabric gensis.block 생성 및 인증서 기관의 MSP 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/admincerts/  
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/cacerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp/admincerts/admin@kmacoorg3-cert.pem  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/admincerts/  
  
scp /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/cacerts/ca.crt  
kmari@kmacoorderer:/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/msp/cacerts/
```

1. kmacoorg3 에서 작업을 진행합니다.
2. kmacoorg3의 admin에서 공개키 복사합니다.
3. kmacoorg3 조직의 운영자인증서와 CA 인증서를 저장할 디렉토리 생성합니다.
4. 이제 kmacoorg2에서 복사를 진행합니다.

Hyperledger Fabric genesis.block 생성 및 인증서 기관의 MSP 복사

생성

```
configtxgen -profile TwoOrgsOrdererGenesis -outputBlock  
genesis.block
```

생성된 genesis.block 내용 확인

```
configtxgen -inspectBlock genesis.block
```

1. kmacoorger 에서 작업을 진행합니다.
2. genesis block 생성을 합니다.

Hyperledger Fabric gensis.block 생성 및 인증서 기관의 MSP 복사

```
mv genesis.block crypto-  
config/ordererOrganizations/kmarogordererorg/orderers/kmaco  
orderer.kmacoordererorg/genesis.block
```

1. Gensis 복사전 kafka-zookeeper은 구동되어 있어야 합니다.
2. 다음 kafka를 참조하여 사전에 구동합니다.
3. 서비스 구동이 되었다면 이제 노드를 실행합니다.

Hyperledger Fabric gensis.block 생성 및 인증서 기관의 MSP 복사

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizationskmacordererorg/orderers/kmacoor  
derer.kmacordererorg/msp/admincerts/
```

```
cp /home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizations/kmacordererorg/msp/admincerts/  
admin@kmacordererorg-  
cert.pem /home/kmari/hlf/kmarinet/crypto-  
config/ordererOrganizations/kmacordererorg/orderers/kmacoo  
rderer.kmacordererorg/msp/admincerts/
```

1. msp 에서 admincerts 만 추가하빈다.
2. kmaorderer 에서 작업합니다.
3. Orderer 에서 관리자 공개키를 복사합니다.

```
vi /home/kmari/hlf/kmarinet/orderer.yaml
```

#중간부분 부터 확인 후 변경

ListenAddress: kmacorderer

GenesisProfile: TwoOrgsOrdererGenesis

**GenesisFile: crypto-
config/ordererOrganizations/kmaorgordererorg/orderers/km
acoorderer.kmacordererorg/genesis.block**

1. Orderer 실행을 위한 환경 설정입니다. kmarinet 안에 있는 orderer.yaml 파일을 수정합니다.
2. 여기서 수정 위치부분에서 변경 부분만 표시했습니다.
3. Yaml 파일 이기 때문에 띄어쓰기를 조심합니다.

LocalMSPDir: crypto-
config/ordererOrganizations/kmacordererorg/orderers/kma-
coorderer.kmacordererorg/m네

LocalMSPID: keocpordererorgMSP

1. 이이서 작업을 진행합니다.

Hyperledger Fabric kmacorderer 노드 실행

```
vi /home/kmari/hlf/kmarinet/runorderer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/orderer
```

1. kmaorderer 에서 작업합니다.
2. runorderer.sh 파일을 생성합니다.
3. chmod +x runorderer.sh 를 실행합니다.
4. sudo ./runorderer.sh 를 실행합니다.

Hyperledger Fabric Kafka 설치 및 구성 메뉴얼

Hyperledger Fabric kafka-zookeeper 설치 및 구성

생성

```
vi /home/kmari/hlf/kmarinet/docker-compose.yaml
```

주요 내용입니다.

1. kmakafka 에서 작업을 진행합니다.
2. home/kmari/hlf/kmarinet/docker-compose.yaml 파일 생성을 생성합니다.

Hyperledger Fabric kafka-zookeeper 설치 및 구성

```
version: '2'
services:
  zookeeper:
    image: hyperledger/fabric-zookeeper
    ports:
      - "2181:2181"
  kafka0:
    image: hyperledger/fabric-kafka
    ports:
      - "9092:9092"
    environment:
      - KAFKA_ADVERTISED_HOST_NAME=10.0.0.12
      - KAFKA_ADVERTISED_PORT=9092
      - KAFKA_BROKER_ID=0
      - KAFKA_MESSAGE_MAX_BYTES=103809024 # 99*1024*1024
      - KAFKA_REPLICA_FETCH_MAX_BYTES=103809024
      - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
      - KAFKA_NUM_REPLICA_FETCHERS=1
      - KAFKA_DEFAULT_REPLICATION_FACTOR=1
      - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
    depends_on:
      - zookeeper
```

1. 이어서 작업합니다.

Hyperledger Fabric kafka-zookeeper 설치 및 구성

```
docker-compose up
```

#또는

```
sudo docker-compose up
```

#도커 확인 방법입니다.

```
sudo netstat -ltnp
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign
Address	State	PID/Program name		
tcp	0	0	127.0.0.53:53	0.0.0.0:*
				LISTEN 318/system
d-resolve				
tcp	0	0	0.0.0.0:22	0.0.0.0:*
				LISTEN 692/sshd
tcp	0	0	127.0.0.1:631	0.0.0.0:*
				LISTEN 3156/cupsd

.....

1. Kafakc-zookeeper 구동은 Docker Compose 로 시작합니다.
2. 서비스를 시작하고 sudo netstat -ltnp 로 내용을 확인합니다

Hyperledger Fabric Peer 서비 스 시작

Hyperledger Fabric Peer 서비스 시작

```
# admincerts 디렉토리 생성
mkdir -p /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer01.km
acoorg1/msp/admincerts/
```

```
# kmacoorg1 운영자 인증서 복사
cp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1
/msp/admincerts/admin@kmacoorg1-
cert.pem /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmaco1org1/peers/kmaorg1peer1.km
acoorg1/msp/admincerts/
```

1. kmacoorg1peer01에서 작업을 진행합니다.
2. admin 인증서를 복사합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/coer.yaml
```

#중간부분 부터 확인 후 변경

id: kmacoorg1-kmaorg1peer01

listenAddress: kmaorg1peer01:7051

address: kmaorg1peer01:7051

1. kmaco1peer01 실행을 위한 환경 설정입니다. kmarinet 안에 있는 coer.yaml 파일을 수정합니다.
2. 여기서 수정 위치부분에서 변경 부분만 표시했습니다.
3. Yaml 파일 이기 때문에 띄어쓰기를 조심합니다.

```
bootstrap: kmaorg1peer01:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg1org1/peers/kmaorg1pe  
er01.kmacoorg1/msp  
  
localMspId: kmacoorg1MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/runorderer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. kmaorderer 에서 작업합니다.
2. runorderer.sh 파일을 생성합니다.
3. chmod +x runorderer.sh 를 실행합니다.
4. sudo ./runorderer.sh 를 실행합니다.

Hyperledger Fabric Peer 서비스 시작

```
# admincerts 디렉토리 생성
mkdir -p /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer01.km
acoorg2/msp/admincerts/
```

```
# kmacoorg2 운영자 인증서 복사
cp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2
/msp/admincerts/admin@kmacoorg2-
cert.pem /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmaco2org1/peers/kmaorg2peer1.km
acoorg2/msp/admincerts/
```

1. kmacoorg2peer01에서 작업을 진행합니다.
2. admin 인증서를 복사합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/coer.yaml
```

#중간부분 부터 확인 후 변경

```
id: kmacoorg2-kmaorg2peer01
```

```
listenAddress: kmaorg2peer01:7051
```

```
address: kmaorg2peer01:7051
```

1. kmaco2peer01 실행을 위한 환경 설정입니다. kmarinet 안에 있는 coer.yaml 파일을 수정합니다.
2. 여기서 수정 위치부분에서 변경 부분만 표시했습니다.
3. Yaml 파일 이기 때문에 띄어쓰기를 조심합니다.

```
bootstrap: kmaorg2peer01:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg1org1/peers/kmaorg2pe  
er02.kmacoorg2/msp  
  
localMspId: kmacoorg2MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/runorderer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. kmaorderer 에서 작업합니다.
2. runorderer.sh 파일을 생성합니다.
3. chmod +x runorderer.sh 를 실행합니다.
4. sudo ./runorderer.sh 를 실행합니다.

Hyperledger Fabric Peer 서비스 시작

```
# admincerts 디렉토리 생성
mkdir -p /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer01.km
acoorg3/msp/admincerts/
```

```
# kmacoorg3 운영자 인증서 복사
cp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3
/msp/admincerts/admin@kmacoorg3-
cert.pem /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmaco3org1/peers/kmaorg3peer1.km
acoorg3/msp/admincerts/
```

1. kmacoorg3peer01에서 작업을 진행합니다.
2. admin 인증서를 복사합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/coer.yaml
```

#중간부분 부터 확인 후 변경

id: kmacoorg3-kmaorg3peer01

listenAddress: kmaorg3peer01:7051

address: kmaorg3peer01:7051

1. kmaco2peer01 실행을 위한 환경 설정입니다. kmarinet 안에 있는 coer.yaml 파일을 수정합니다.
2. 여기서 수정 위치부분에서 변경 부분만 표시했습니다.
3. Yaml 파일 이기 때문에 띄어쓰기를 조심합니다.

```
bootstrap: kmaorg3peer01:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg3org1/peers/kmaorg3pe  
er01.kmacoorg3/msp  
  
localMspId: kmacoorg3MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric Peer 서비스 시작

```
vi /home/kmari/hlf/kmarinet/runorderer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. kmaorderer 에서 작업합니다.
2. runorderer.sh 파일을 생성합니다.
3. chmod +x runorderer.sh 를 실행합니다.
4. sudo ./runorderer.sh 를 실행합니다.

Hyperledger Fabric 채널 생성 메뉴얼

Hyperledger Fabric 채널 생성

```
vi /home/kmari/hlf/kmarinet/ceate-channel.sh
```

#내용

```
FABRIC_PATH=/home/kmair/hlf/kmarinet  
echo $FABRIC_PATH
```

```
CORE_PEER_LOCALMSPID="kmacoorg1MSP" ₩  
CORE_PEER_TLS_ROOTCERT_FILE=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer01.km  
acoorg1/tls/ca.crt ₩  
CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp ₩  
CORE_PEER_ADDRESS=kmaorg1peer01:7051 ₩  
peer channel create -o orderer:7050 -c ch1 -f ch1.tx
```

1. 조직의 관리자 서버에서 작업을 진행합니다.
2. kmacoorg1의 kmaorg1peer01 에서 작업을 합니다.
3. /home/kmari/hlf/kmarinet/create-channel.sh 스크립트 파일 생성합니다.
4. chmod +x create-channel.sh 를 실행합니다.
5. sudo ./ceate-channelsh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/join-channel-peer01.sh
```

#내용

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmacoorg1peer01:7051  
peer channel join -b ch1.block
```

1. kmacoorg1의 kmacoorg1peer01 에서 작업을 합니다.
2. /home/kmari/hlf/kmarinet/join-channel-peer01.sh 스크립트 파일 생성 합니다.
3. ./join-channel-peer01.sh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/join-channel-peer02.sh
```

#내용

```
FABRIC_PATH=/home/kmair/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg2MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp  
export CORE_PEER_ADDRESS=kmaorg2peer01:7051  
peer channel join -b ch1.block
```

1. kmacoorg2의 kmaorg2peer01 에서 작업을 합니다.
2. /home/kmari/hlf/kmarinet/join-channel-peer02.sh 스크립트 파일 생성합니다.
3. ./join-channel-peer02.sh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/join-channel-peer03.sh
```

#내용

```
FABRIC_PATH=/home/kmair/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg3MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp  
export CORE_PEER_ADDRESS=kmaorg3peer01:7051  
peer channel join -b ch1.block
```

1. kmacoorg3의 kmaorg3peer01 에서 작업을 합니다.
2. /home/kmari/hlf/kmarinet/join-channel-peer03.sh 스크립트 파일 생성합니다.
3. ./join-channel-peer03.sh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
# Anchor 업데이트 트랜잭션 생성
configtxgen -profile TwoOrgsChannel -
outputAnchorPeersUpdate kmacoorg1MSPanchors.tx -
channelID ch1 -asOrg kmacoorg1MSP
$ configtxgen -profile TwoOrgsChannel -
outputAnchorPeersUpdate kmacoorg2MSPanchors.tx -
channelID ch1 -asOrg kmacoorg2MSP
$ configtxgen -profile TwoOrgsChannel -
outputAnchorPeersUpdate kmacoorg3MSPanchors.tx -
channelID ch1 -asOrg kmacoorg3MSP
```

1. 이번에는 kmacoorderer에서 작업을 합니다.
2. Anchor peer 업데이트와 트랜잭션을 생성합니다.

Hyperledger Fabric Peer 채널 참여

```
scp kmacoorg1MSPanchors.tx  
kmari@kmaorg1peer01:/home/kmari/hlf/kmarinet
```

```
scp kmacoorg2MSPanchors.tx  
kmari@kmaorg2peer01:/home/kmari/hlf/kmarinet
```

```
scp kmacoorg3MSPanchors.tx  
kmari@kmaorg3peer01:/home/kmari/hlf/kmarinet
```

1. 생성된 tx를 모두 admin@kmacoorg1 , 2, 3 으로 전송합니다.

Hyperledger Fabric Peer 채널 참여

```
vi kmacoorg1-anchor.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmaogr1peer01:7051  
peer channel create -o orderer:7050 -c ch1 -f  
kmacoorg1MSPanchors.tx
```

1. 이제 anchor peer 업데이트를 하기 위해 kmacoorg1의 kmacoorg1peer01 서버에서 작업을 진행합니다.
2. 스크립트를 생성하여 실행합니다.
3. /home/kmari/hlf/kmarinet/kmacoorg1-anchor.sh 스크립트 파일 생성합니다.
4. chmod +x kmacoorg1-anchor.sh 를 실행합니다.
5. sudo ./kmacoorg1-anchor.sh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
vi kmacoorg2-anchor.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg2MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp  
export CORE_PEER_ADDRESS=kmaorg2peer01:7051  
peer channel create -o orderer:7050 -c ch1 -f  
kmacoorg2MSPanchors.tx
```

1. 이제 anchor peer 업데이트를 하기 위해 kmacoorg2의 kmacoorg2peer01 서버에서 작업을 진행합니다.
2. 스크립트를 생성하여 실행합니다.
3. /home/kmari/hlf/kmarinet/kmacoorg2-anchor.sh 스크립트 파일 생성합니다.
4. chmod +x kmacoorg2-anchor.sh 를 실행합니다.
5. sudo ./kmacoorg2-anchor.sh 를 실행합니다

Hyperledger Fabric Peer 채널 참여

```
vi kmacoorg3-anchor.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg3MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp  
export CORE_PEER_ADDRESS=kmaorg3peer01:7051  
peer channel create -o orderer:7050 -c ch1 -f  
kmacoorg3MSPanchors.tx
```

1. 이제 anchor peer 업데이트를 하기 위해 kmacoorg3의 kmacoorg3peer01 서버에서 작업을 진행합니다.
2. 스크립트를 생성하여 실행합니다.
3. /home/kmari/hlf/kmarinet/kmacoorg3-anchor.sh 스크립트 파일 생성합니다.
4. chmod +x kmacoorg3-anchor.sh 를 실행합니다.
5. sudo ./kmacoorg3-anchor.sh 를 실행합니다

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/  
fabric-ca-client-config.yaml
```

#추가할 내용

```
id:  
  name: kmaorg1peer02  
  type: peer  
  affiliation: kmacoorg1  
  maxenrollments: -1  
  attributes:  
    - name: role  
      value: peer  
      ecert: true
```

1. kmacoorg1의 kmacoorg1peer01 에서 작업을 합니다.
2. kmacoorg1peer02의 계정을 생성하기 위해 yaml 파일을 수정합니다.
3. /home/kmari/hlf/kmarinet/crypto-config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/fabric-ca-client-config.yaml 파일을 수정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
fabric-ca-client register --id.secret=dlwhdrjs -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoo  
rg1/
```

1. -H 옵션으로 파일의 경로를 지정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

#먼저 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.km  
acoorg1/
```

```
fabric-ca-client enroll -u  
http://kmaco1peer01:admin@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.  
kmacoorg1/
```

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.km  
acoorg1/msp/cacerts/10-0-0-100-7054.pem  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.km  
acoorg1/msp/cacerts/ca.crt
```

1. kmacoorg1의 kmacoorg1peer02 에서 작업을 합니다.
2. 서버를 꼭 확인하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
# [개인키] 파일이름 확인 후 복사
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.km
acoorg1/msp/keystore/cb2069c15665e8d19ec9d737370bd55419
309d2ae93e333e698f536c186bd8de_sk /home/
kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.km
acoorg1/msp/keystore/server.key
```

1. 이어서 계속 작업을 합니다.
2. kmacoorg1의 조직관리자가 등록한 계정의 패스워드를 기억하고 등록할 때 꼭 사용해야 합니다.
3. 개인키의 값이 틀리면 서비스도 작동하지 않습니다. – 빨간 부분 참조

Hyperledger Fabric 2번째 Peer 채널 참여

```
# kmacoorg1의 kmaorg1peer01에서 실행
scp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1
/msp/admincerts/admin@kmacoorg1-cert.pem
kmari@kmaco1peer01:/home/kmari/hlf/kmarinet

# kmacoorg1의 kmaorg1peer02에서 실행
mkdir -p /home/ kmari/hlf/kmarinet /crypto-
config/peerOrganizations/org0/peers/kmaorg1peer02.kmacoorg
1/msp/admincerts/

mv /home/ kmari/hlf/kmarinet /admin@kmacoorg1-
cert.pem /home/ kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmacoorg1/peers/kmaorg1peer02.org
0/msp/admincerts/
```

1. kmacoorg1의 kmaorg1peer02 구동을 확인하기 위해서 작업 위치는 kmacoorg1의 kmaorg1peer01에서 작업을 진행 먼저 진행합니다. 이유는 인증서 복사를 먼저 해야 합니다.
2. 복사를 마치고 난 다음 kmaorg1peer02 서버에서 작업을 합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/core.yaml
```

#중간부분 부터 확인 후 변경

id: kmaoorg1-kmaco1peer02

listenAddress: kmaorg1peer02:7051

address: kmaorg1peer02:7051

1. 이어서 kmaorg1peer02 서버에서 계속 작업을 합니다.
2. core.yaml 파일을 수정합니다.
3. Yaml 파일이기 때문에 띄어쓰기를 주의합니다.
4. 보통은 kmaorg1peer01의 core.yaml 파일을 복사해서 내용만 변경하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
bootstrap: kmaorg1peer02:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg1org1/peers/kmaorg1pe  
er02.kmacoorg2/msp  
  
localMspId: kmacoorg1MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kpri/hlf/kmarinet/runPeer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. 스크립트를 생성하여 실행합니다.
2. /home/kmari/hlf/kmarinet/runPeer.sh 스크립트 파일 생성합니다.
3. `chmod +x runPeer.sh` 를 실행합니다.
4. `sudo ./runPeer.sh` 를 실행합니다

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2/  
fabric-ca-client-config.yaml
```

#추가할 내용

```
id:  
  name: kmaorg2peer02  
  type: peer  
  affiliation: kmacoorg2  
  maxenrollments: -1  
  attributes:  
    - name: role  
      value: peer  
      ecert: true
```

1. kmacoorg2의 kmacoorg2peer01 에서 작업을 합니다.
2. kmacoorg2peer02의 계정을 생성하기 위해 yaml 파일을 수정합니다.
3. /home/kmari/hlf/kmarinet/crypto-config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2/fabric-ca-client-config.yaml 파일을 수정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
fabric-ca-client register --id.secret=dlwhdrjs -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoo  
rg2/
```

1. -H 옵션으로 파일의 경로를 지정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

#먼저 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.km  
acoorg2/
```

```
fabric-ca-client enroll -u  
http://kmaco1peer01:admin@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.  
kmacoorg2/
```

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.km  
acoorg2/msp/cacerts/10-0-0-100-7054.pem  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.km  
acoorg2/msp/cacerts/ca.crt
```

1. kmacoorg2의 kmacoorg2peer02 에서 작업을 합니다.
2. 서버를 꼭 확인하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
# [개인키] 파일이름 확인 후 복사
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.km
acoorg2/msp/keystore/cb2069c15665e8d19ec9d737370bd55419
309d2ae93e333e698f536c186bd8de_sk /home/
kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.km
acoorg2/msp/keystore/server.key
```

1. 이어서 계속 작업을 합니다.
2. kmacoorg1의 조직관리자가 등록한 계정의 패스워드를 기억하고 등록할 때 꼭 사용해야 합니다.
3. 개인키의 값이 틀리면 서비스도 작동하지 않습니다. – 빨간 부분 참조

Hyperledger Fabric 2번째 Peer 채널 참여

```
# kmacoorg2의 kmaorg2peer01에서 실행
scp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2
/msp/admincerts/admin@kmacoorg2-cert.pem
kmari@kmaco2peer02:/home/kmari/hlf/kmarinet
```

```
# kmacoorg2의 kmaorg2peer02에서 실행
mkdir -p /home/ kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmaorgorg2/peers/kmaorg2peer02.k
macoorg2/msp/admincerts/
```

```
mv /home/ kmari/hlf/kmarinet /admin@kmacoorg2-
cert.pem /home/ kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmacoorg2/peers/kmaorg2peer02.org
0/msp/admincerts/
```

1. kmacoorg2의 kmaorg2peer02 구동을 확인하기 위해서 작업 위치는 kmacoorg2의 kmaorg2peer01에서 작업을 진행 먼저 진행합니다. 이유는 인증서 복사를 먼저 해야 합니다.
2. 복사를 마치고 난 다음 kmaorg2peer02 서버에서 작업을 합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/core.yaml
```

#중간부분 부터 확인 후 변경

id: kmacoorg2-kmaorg2peer02

listenAddress: kmaorg2peer02:7051

address: kmaorg2peer02:7051

1. 이어서 kmaorg2peer02 서버에서 계속 작업을 합니다.
2. core.yaml 파일을 수정합니다.
3. Yaml 파일이기 때문에 띄어쓰기를 주의합니다.
4. 보통은 kmaorg2peer01의 core.yaml 파일을 복사해서 내용만 변경하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
bootstrap: kmaorg2peer02:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg2org1/peers/kmaorg2pe  
er02.kmacoorg2/msp  
  
localMspId: kmacoorg2MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kpri/hlf/kmarinet/runPeer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. 스크립트를 생성하여 실행합니다.
2. /home/kmari/hlf/kmarinet/runPeer.sh 스크립트 파일 생성합니다.
3. `chmod +x runPeer.sh` 를 실행합니다.
4. `sudo ./runPeer.sh` 를 실행합니다

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/  
fabric-ca-client-config.yaml
```

#추가할 내용

```
id:  
  name: kmaorg3peer02  
  type: peer  
  affiliation: kmacoorg3  
  maxenrollments: -1  
  attributes:  
    - name: role  
      value: peer  
      ecert: true
```

1. kmacoorg3의 kmacoorg3peer01 에서 작업을 합니다.
2. kmacoorg3peer02의 계정을 생성하기 위해 yaml 파일을 수정합니다.
3. /home/kmari/hlf/kmarinet/crypto-config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3/fabric-ca-client-config.yaml 파일을 수정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
fabric-ca-client register --id.secret=dlwhdrjs -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoo  
rg3/
```

1. -H 옵션으로 파일의 경로를 지정합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

#먼저 디렉토리 생성

```
mkdir -p /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.km  
acoorg2/
```

```
fabric-ca-client enroll -u  
http://kmaorg3peer01:admin@10.0.0.100:7054 -H  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.  
kmacoorg3/
```

```
mv /home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.km  
acoorg3/msp/cacerts/10-0-0-100-7054.pem  
/home/kmari/hlf/kmarinet/crypto-  
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.km  
acoorg3/msp/cacerts/ca.crt
```

1. kmacoorg2의 kmacoorg2peer02 에서 작업을 합니다.
2. 서버를 꼭 확인하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
# [개인키] 파일이름 확인 후 복사
mv /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.km
acoorg3/msp/keystore/cb2069c15665e8d19ec9d737370bd55419
309d2ae93e333e698f536c186bd8de_sk /home/
kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.km
acoorg3/msp/keystore/server.key
```

1. 이어서 계속 작업을 합니다.
2. kmacoorg1의 조직관리자가 등록한 계정의 패스워드를 기억하고 등록할 때 꼭 사용해야 합니다.
3. 개인키의 값이 틀리면 서비스도 작동하지 않습니다. – 빨간 부분 참조

Hyperledger Fabric 2번째 Peer 채널 참여

```
# kmacoorg3의 kmaorg3peer01에서 실행
scp /home/kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3
/msp/admincerts/admin@kmacoorg3-cert.pem
kmari@kmaorg3peer01:/home/kmari/hlf/kmarinet
```

```
# kmacoorg3의 kmaorg3peer02에서 실행
mkdir -p /home/ kmari/hlf/kmarinet /crypto-
config/peerOrganizations/kmaorgorg3/peers/kmaorg3peer02.k
macoorg2/msp/admincerts/
```

```
mv /home/ kmari/hlf/kmarinet /admin@kmacoorg3-
cert.pem /home/ kmari/hlf/kmarinet/crypto-
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.org
0/msp/admincerts/
```

1. kmacoorg3의 kmaorg3peer02 구동을 확인하기 위해서 작업 위치는 kmacoorg3의 kmaorg3peer01에서 작업을 진행 먼저 진행합니다. 이유는 인증서 복사를 먼저 해야 합니다.
2. 복사를 마치고 난 다음 kmaorg3peer02 서버에서 작업을 합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kmari/hlf/kmarinet/core.yaml
```

#중간부분 부터 확인 후 변경

id: kmacoorg3-kmaco3peer02

listenAddress: kmaorg3peer02:7051

address: kmaorg3peer02:7051

1. 이어서 kmaorg3peer02 서버에서 계속 작업을 합니다.
2. core.yaml 파일을 수정합니다.
3. Yaml 파일이기 때문에 띄어쓰기를 주의합니다.
4. 보통은 kmaorg3peer01의 core.yaml 파일을 복사해서 내용만 변경하고 작업합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
bootstrap: kmaorg3peer02:7051  
  
useLeaderElection: true  
  
useLeaderElection: true  
  
mspConfigPath: crypto-  
config/peerOrganizations/kmacoorg3/peers/kmaorg3peer02.  
kmacoorg3/msp  
  
localMspId: kmacoorg3MSP
```

1. 이이서 작업을 진행합니다.

Hyperledger Fabric 2번째 Peer 채널 참여

```
vi /home/kpri/hlf/kmarinet/runPeer.sh
```

#내용

```
/home/kmair/hlf/src/github.com/hyperledger/fabric/.build/bin/p  
eer node start
```

1. 스크립트를 생성하여 실행합니다.
2. /home/kmari/hlf/kmarinet/runPeer.sh 스크립트 파일 생성합니다.
3. `chmod +x runPeer.sh` 를 실행합니다.
4. `sudo ./runPeer.sh` 를 실행합니다

체인 코드 설치 메뉴얼

```
vi /home/kmari/hlf/kmarinet/installCCpeer01.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmaorg1peer01:7051  
peer chaincode install -n kmarinetCC -v 1.0 -p  
github.com/hyperledger/fabric/examples/chaincode/go/exa  
mple02/cmd
```

1. kmacoorg1의 kmacoorg1peer01에서 서버에서 진행합니다.
2. admin@kmacoorg1의 MSP 권한으로 설치를 해야 합니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/installCCpeer01.sh
5. chmod +x installCCPeer01.sh 를 실행합니다.
6. ./installCCpeer01.sh 를 실행합니다

```
vi /home/kmari/hlf/kmarinet/installCCpeer02.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg2MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg2/users/admin@kmacoorg2  
/msp  
export CORE_PEER_ADDRESS=kmaorg2peer01:7051  
peer chaincode install -n kmarinetCC -v 1.0 -p  
github.com/hyperledger/fabric/examples/chaincode/go/exa  
mple02/cmd
```

1. kmacoorg2의 kmacoorg2peer01에서 서버에서 진행합니다.
2. admin@kmacoorg2의 MSP 권한으로 설치를 해야 합니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/installCCpeer02.sh
5. chmod +x installCCPeer02.sh 를 실행합니다.
6. ./installCCpeer02.sh 를 실행합니다

```
vi /home/kmari/hlf/kmarinet/installCCpeer03.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg3MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg3/users/admin@kmacoorg3  
/msp  
export CORE_PEER_ADDRESS=kmaorg3peer01:7051  
peer chaincode install -n kmarinetCC -v 1.0 -p  
github.com/hyperledger/fabric/examples/chaincode/go/exa  
mple02/cmd
```

1. kmacoorg3의 kmaorg3peer01에서 서버에서 진행합니다.
2. admin@kmacoorg3의 MSP 권한으로 설치를 해야 합니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/installCCpeer03.sh
5. chmod +x installCCPeer03.sh 를 실행합니다.
6. ./installCCpeer03.sh 를 실행합니다

```
vi /home/kmari/hlf/kmarinet/installCClist.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmacoorg1peer01:7051  
peer chaincode list -C ch1 --installed
```

1. kmacoorg1의 kmaorg1peer01에서 서버에서 진행합니다.
2. 설치된 내용을 확인합니다.
3. 이건 한곳에서만 진행합니다.
4. 스크립트 파일을 생성합니다.
5. /home/kmari/hlf/kmarinet/installCClist.sh
6. chmod +x installCClist.sh 를 실행합니다.
7. ./installCClist.sh 를 실행합니다

```
vi /home/kmari/hlf/kmarinet/instantiateCC.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmaorg1peer01:7051  
peer chaincode instantiate -o kmacoorderer:7050 -C ch1 -n  
kmarinetCC -v 1.0 -c '{"Args":["init","a","200", "b", "300"]}' -  
P "OR ('kmacoorg1MSP.member', 'kmacoorg2MSP.member',  
'kmacoorg3MSP.member')"
```

1. kmacoorg1의 kmaorg1peer01에서 서버에서 진행합니다.
2. 체인코드 인스턴스 생성을 합니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/instantiateC
C.sh
5. chmod +x instantiateCC.sh 를 실행합
니다.
6. ./instantiateCC.sh 를 실행합니다

```
vi /home/kmari/hlf/kmarinet/instantiateCC.sh
```

입력하고 다음 내용을 추가합니다.

```
FABRIC_PATH=/home/kmari/hlf/kmarinet  
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmacoorg1MSP"  
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-  
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1  
/msp  
export CORE_PEER_ADDRESS=kmaorg1peer01:7051  
peer chaincode list -C ch1 --instantiated
```

1. kmacoorg1의 kmaorg1peer01에서 서버에서 진행합니다.
2. 인스턴스 목록을 확인합니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/instantiateCClist.sh
5. chmod +x instantiateCClist.sh 를 실행합니다.
6. ./instantiateCClist.sh 를 실행합니다

체인코드 설치 - 기본 샘플을 기준으로 테스트 용도

```
vi /home/kmari/hlf/kmarinet/query.sh
```

입력하고 다음 내용을 추가합니다.

```
help()
{
    echo "Usage: $0 [variable]"
}
```

```
if [ $# -ne 1 ]
then
    help
    exit 0
fi
```

```
FABRIC_PATH=/home/kmari/hlf/kmarinet
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmaorg1org1MSP"
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/msp
export CORE_PEER_ADDRESS=kmaorg1peer01:7051
peer chaincode query -C ch1 -n kmarinetCC -c "{W"ArgsW":[W"queryW",W"$1W"]}"
```

1. kmacoorg1의 kmaorg1peer01에서 서버에서 진행합니다.
2. 분산원장 데이터 읽기 테스트를 해봅니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/query.sh
5. chmod +x query.sh를 실행합니다.
6. ./query.sh a 를 실행합니다
7. 쿼리에서 a 를 스크립트 인자(매개변수)로 넣어서 확인할 수 있습니다.

체인코드 설치 - 기본 샘플을 기준으로 테스트 용도

```
vi /home/kmari/hlf/kmarinet/query.sh
```

입력하고 다음 내용을 추가합니다.

```
help()
{
    echo "Usage: $0 [variable]"
}
```

```
if [ $# -ne 1 ]
then
    help
    exit 0
fi
```

```
FABRIC_PATH=/home/kmari/hlf/kmarinet
echo $FABRIC_PATH
```

```
export CORE_PEER_LOCALMSPID="kmaorg1org1MSP"
export CORE_PEER_MSPCONFIGPATH=$FABRIC_PATH/crypto-
config/peerOrganizations/kmacoorg1/users/admin@kmacoorg1/msp
export CORE_PEER_ADDRESS=kmaorg1peer01:7051
peer chaincode invoke -o kmacoorderer:7050 -C ch1 -n kmarinetCC -c
'{"Args":["invoke","a","b","50"]}'
```

1. kmacoorg1의 kmaorg1peer01에서 서버에서 진행합니다.
2. 분산원장 데이터 기록 테스트를 해봅니다.
3. 스크립트 파일을 생성합니다.
4. /home/kmari/hlf/kmarinet/invoke.sh
5. chmod +x invoke.sh를 실행합니다.
6. ./invoke.sh 를 실행합니다
7. ./query.sh a 또는 ./query.sh b
8. 쿼리에서 a 를 스크립트 인자(매개변수)로 넣어서 확인할 수 있습니다.