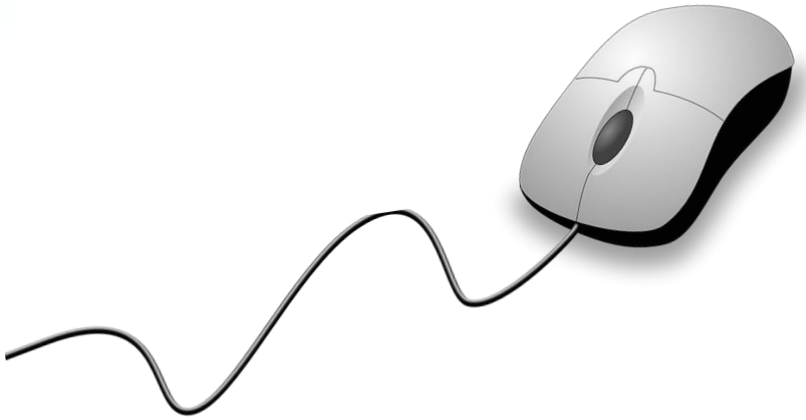
공개SW 솔루션 설치 & 활용 가이드

기타 -> 블록체인



HYPERLEDGER

블록쉴드 v1.0

제대로 배워보자

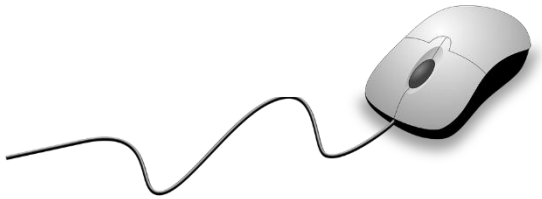
How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center



CONTENTS

1. 개요

<table>
<tr><td>소개</td><td colspan="3">• IBM HyperLedger Fabric의 Private 블록체인 네트워크 엔진을 기반으로 멀티호스트 블록체인 네트워크를 구성하고 오픈소스 기반의 모니터링 툴, 컨테이너 관리툴을 병합하여 docker 블록체인 네트워크를 쉽게 관리할 수 있도록 제공하는 솔루션</td></tr>
<tr><td>주요기능</td><td colspan="3">• 허가형 블록체인 네트워크 구성
• 블록 체인 모니터링 관리도구 툴
• 도커 컨테이너 관리도구 툴</td></tr>
<tr><td>대분류</td><td>• 기타</td><td>소분류</td><td>• 블록체인</td></tr>
<tr><td rowspan="2">라이선스형태</td><td rowspan="2">• EPL 1.0
(https://github.com/sonatype/nexus-public/blob/master/LICENSE.txt)</td><td>사전설치 솔루션</td><td>• 없음</td></tr>
<tr><td>버전</td><td>• 3.29.x (2020년 12월 기준)</td></tr>
<tr><td>특징</td><td colspan="3">• 직관적이고 사용자 친화적 UI
• WEB UI를 통한 블록 트랜잭션 모니터링
• WEB UI를 통한 컨테이너 관리 기능</td></tr>
<tr><td>공식 홈페이지</td><td colspan="3">• http://www.exotech.kr</td></tr>
</table>

1. 개요

블록쉴드 v1.0 | 허가형 블록체인 네트워크 설정 및 관리 솔루션

- ✓ 블록쉴드는 IBM Hyperledger Fabric 블록체인 네트워크를 구성하고 구성한 블록 네트워크를 모니터링
- ✓ 별도의 App 커스터마이즈 개발 제공
- ✓ Docker 기반의 블록체인 네트워크를 관리할 수 있도록 오픈소스 컨테이너 툴을 커스터마이즈 하여 제공
- ✓ 블록체인 네트워크 구성, 모니터링, 컨테이너 관리까지 통합하여 제공하는 오픈소스 프로젝트

2. 기능요약

- 블록체인 네트워크 구성
  - ✓ Fabric 1.4 기반의 블록체인 네트워크 구성
  - ✓ 멀티호스트 설정
  - ✓ Artifact 생성 툴
  - ✓ Restful API
- 블록 네트워크 모니터링 기능
  - ✓ 채널별 트랜잭션 모니터링
  - ✓ 트랜잭션 별 상세 정보 확인
  - ✓ 블록의 현재 해시정보 확인
- 컨테이너 관리도구
  - ✓ 엔드 포인트 기능으로 멀티호스트되고 있는 컨테이너 관리
  - ✓ 컨테이너 시작/정지/멈춤/재시작 등 컨테이너 실행 관리 기능
  - ✓ Docker-compose 작성/배포 기능으로 신규 컨테이너 생성 기능

3. 실행환경

- 호스트 운영 체제
  - ✓ Linux Ubuntu 18.04.5 LTS
- 전용 운영체제 사용자 계정
  - ✓ 최소 CPU 4core, 권장 CPU 8core 이상
- Memory 요구 사항

	호스트 물리적 / RAM
최소 (기본값)	16GB
최고	제한 없음

- DISK
  - ✓ SSD 필수

4. 설치 및 실행

세부 목차

4. 설치 및 실행

4.1 블록체인 네트워크 구성

1) 기본 패키지 설치

```
$ sudo apt-get install -y golang-1.10
$ sudo apt-get install -y curl
$ sudo apt-get install -y docker.io docker-compose
$ sudo apt-get install git –y
$ sudo apt-get install npm –y
$ sudo apt-get install libltdl-dev
```

2) 환경변수 설정 - GO

```
$ mkdir works
$ vi .profile
====
export GOROOT=/usr/lib/go-1.10
export GOPATH=/home/사용자계정/works
export BLOCKSHIELD_CA_HOME=$GOPATH/src/github.com/hyperledger/fabric-ca
export BLOCKSHIELD _CA_SERVER_HOME=/home/사용자계정/works/poc
export BLOCKSHIELD _CA_CLIENT_HOME=/home/사용자계정/works/poc
export BLOCKSHIELD _CFG_PATH=$GOPATH/poc
export PATH=$GOROOT/bin:$GOPATH/bin:$PATH
====
$ source .profile
$ go version
```

2) 환경변수 설정 – hoots 설정

```
$ vi /etc/hosts
====
ip주소 node1    orderer1   peer0.org1        zk1      kafka1   ca-org1
ip주소 node2    orderer2    peer1.org1         zk2      kafka2
ip주소 node3    orderer3     peer0         zk3      kafka3   ca-org2
ip주소 node4    peer1.org2      kafka4
ip주소 node5    peer2.org1
ip주소 node6    peer2.org2
====
```

3) 각 서버의 엔진파일 업로드

▶ 엔진파일을 준비하여 각 서버에 업로드 한 후 압축을 풀어줍니다.

구분	파일명	압축해제 경로
Node1번 서버	node1_Main_Engine.tgz	/home/사용자계정/works/poc
Node2번 서버	Node2.tgz	/home/사용자계정/works/poc
Node3번 서버	Node3.tgz	/home/사용자계정/works/poc
Node4번 서버	Node4.tgz	/home/사용자계정/works/poc
Node5번 서버	Node5.tgz	/home/사용자계정/works/poc
Node6번 서버	Node6.tgz	/home/사용자계정/works/poc

3) 각 서버의 엔진파일 업로드

▶ 압축을 해제하면 <그림> 과 같은 엔진 파일을 볼 수 있습니다. 각 파일의 역할은 <표> 를 참고합니다..

```
channel-artifacts/
configtx.yaml
crypto-config/
crypto-config.yaml
docker-compose-ca-org1.yaml
docker-compose-kafka1.yaml
docker-compose-orderer1.yaml
docker-compose-peer0-org1.yaml
docker-compose-zk1.yaml
restful_api/
step1-start-zk1.sh*
step2-start-kafka1.sh*
step3-start-ca-org1.sh*
step4-start-orderer1.sh*
step5-start-peer0-org1.sh*
Tool_channelArtifacts/
```

구분	역할	구분	역할
configtx.yaml	- 블록체인 네트워크 profile이 설정하는 파일 - genesis.bloc 성능 설정파일	step1-start-zk1.sh	Zookeeper 컨테이너 실행하는 쉘 스크립트
docker-compose-ca-org1.yaml	- CA인증기관 역할을 하는 컨테이너를 생성하는 docker-compose 파일	step2-start-kafka1.sh	kafka 컨테이너 실행하는 쉘 스크립트
docker-compose-kafka1.yaml	- kafka 역할을 하는 컨테이너를 생성하는 docker-compose 파일	step3-start-ca-org1.sh	CA 컨테이너 실행하는 쉘 스크립트
docker-compose-orderer1.yaml	- orderer 역할을 하는 컨테이너를 생성하는 docker-compose 파일	step4-start-orderer1.sh	Orderer 컨테이너 실행하는 쉘 스크립트
docker-compose-peer0-org1.yaml	- peer 역할을 하는 컨테이너를 생성하는 docker-compose 파일	step5-start-peer0-org1.sh	Peer 컨테이너 실행하는 쉘 스크립트
docker-compose-zk1.yaml	- zookeeper 역할을 하는 컨테이너를 생성하는 docker-compose 파일	-	-

4) 단계별 스크립트 실행

▶ 각 노드 서버에 엔진 압축 파일을 정상적으로 실행하고 나면 단계별로 컨테이너를 생성하는 스크립트를 실행하여 기본 블록체인 네트워크를 구축한다.

1) STEP1 - zookeeper 구동

서버	스크립트 파일 및 명령어
Node1	$ cd /home/사용자계정/works/poc $ ./step1-start-zk1.sh
Node2	$ cd /home/사용자계정/works/poc $ ./step1-start-zk2.sh
Node3	$ cd /home/사용자계정/works/poc $ ./step1-start-zk2.sh

2) STEP2 - kafka 구동

서버	스크립트 파일 및 명령어
Node1	$ cd /home/사용자계정/works/poc $ ./step2-start-kafka1.sh
Node2	$ cd /home/사용자계정/works/poc $ ./step2-start-kafka2.sh
Node3	$ cd /home/사용자계정/works/poc $ ./step2-start-kafka3.sh
Node4	$ cd /home/사용자계정/works/poc $ ./step2-start-kafka4.sh

4) 단계별 스크립트 실행

▶ 각 노드 서버에 엔진 압축 파일을 정상적으로 실행하고 나면 단계별로 컨테이너를 생성하는 스크립트를 실행하여 기본 블록체인 네트워크를 구축한다.

3) STEP3 - CA 구동

서버	스크립트 파일 및 명령어
Node1	$ cd /home/사용자계정/works/poc $ ./step3-start-ca-org1.sh
Node3	$ cd /home/사용자계정/works/poc $ ./step3-start-ca-org2.sh

4) STEP4- ORDERER 구동

서버	스크립트 파일 및 명령어
Node1	$ cd /home/사용자계정/works/poc $ ./step4-start-orderer1.sh
Node2	$ cd /home/사용자계정/works/poc $ ./step4-start-orderer2.sh
Node3	$ cd /home/사용자계정/works/poc $ ./step4-start-orderer3.sh

4) 단계별 스크립트 실행

5) STEP5 - PEER 구동

서버	스크립트 파일 및 명령어
Node1	$ cd /home/사용자계정/works/poc $ ./ step5-start-peer0-org1.sh
Node2	$ cd /home/사용자계정/works/poc $ ./ step5-start-peer1-org1.sh
Node3	$ cd /home/사용자계정/works/poc $ ./step5-start-peer0-org2.sh
Node4	$ cd /home/사용자계정/works/poc $ ./step5-start-peer1-org2.sh
Node5	$ cd /home/사용자계정/works/poc $ ./step5-start-peer2-org1.sh
Node6	$ cd /home/사용자계정/works/poc $ ./step5-start-peer2-org2.sh

6) STEP6 - 각 노드 서버를 터미널로 접속하여 정상적으로 컨테이너들이 실행되어 있는지 확인합니다.

$ docker ps

```
e7f362b1e1aa        hyperledger/fabric-peer:1.4.0
    0.0.0.0:7051-7053->7051-7053/tcp                                peer0.org1.sayit.kr
17c2a7b7c691        hyperledger/fabric-orderer:1.4.0
    0.0.0.0:7050->7050/tcp                                          orderer1.sayit.kr
4fcd8bdc6ad2        hyperledger/fabric-ca:1.4.0
    0.0.0.0:7054->7054/tcp                                          ca-org1
23326a975e52        hyperledger/fabric-kafka
    0.0.0.0:9092->9092/tcp, 9093/tcp                                kafka1
c284e85833d6        hyperledger/fabric-zookeeper
    0.0.0.0:2181->2181/tcp, 0.0.0.0:2888->2888/tcp, 0.0.0.0:3888->3888/tcp   zk1
"peer node start"       2 days ago          Up 27 hours
"orderer"               2 days ago          Up 2 days
"sh -c 'fabric-ca-se…"  2 days ago          Up 2 days
"/docker-entrypoint.…"  2 days ago          Up 2 days
"/docker-entrypoint.…"  2 days ago          Up 2 days
```

<그림. 컨테이너들이 정상적으로 실행된 화면>

5) API 서버 실행

▶ 블록체인 네트워크에 명령을 실행할 수 있는 api 서버를 구동합니다. api 서버는 노드 1번 서버를 메인으로 설정하여 구성합니다.

```
$ cd /home/사용자계정/works/poc/restful_api
$  ./runApp.sh
```

6) 채널 구성

▶ 블록체인 네트워크가 정상적으로 실행되는지를 모니터링 하기 위하여 별도의 터미널을 열어 채널을 구성합니다. 채널 생성을 용이하게 할 수 있도록 별도 채널 생성 스크립트 파일을 제공합니다.

```
$ cd /home/사용자계정/works/poc/restful_api
$ ./ step1_channel_create_join_mychannel.sh
```

▶ 위 과정은 조직간 통신하려는 채널을 구성하고, 생성된 채널에 peer 노드를 조인한 후 각 조직 노드의 anchor 를 설정하는 프로세스를 전처리해주는 스크립트로, 단계가 성공적으로 진행되면 {success : true 또는 false} 값을 표시하여 네트워크 관리를 용이하게 합니다.

7) 체인코드 설치 및 초기화

▶ 가장 중요한 스마트 컨트렉트에 해당되는 체인코드를 채널에 설치하고 초기화하는 과정입니다.
스마트 컨트렉트는 개별 사이트에 맞게 개발합니다.

1) 체인코드 설치 및 초기화

```
$ cd /home/사용자계정/works/poc/restful_api/api_script
$ ./install_cfrmcc.sh
$ ./init_ordercc.sh mychannel
// dev-peer0.org1.sayit.kr-ordercc-v0 컨테이너가 생성되었는지 확인
$ docker ps
```

1) 라이브러리 종속성

▶ 블록쉴드 탐색기를 설치하고 실행하는데 필요한 소프트웨어 종속성입니다.

- Nodejs 8.11.x (v9.x는 아직 지원되지 않음)
- PostgreSQL 9.5 이상
- jq
- Ubuntu 또는 MacOS와 같은 Linux 기반 운영 체제
- Docker CE 18.09.2 이상
- Docker-compose 1.14.0

2) 패키지 압축 해제

```
$ tar xvfz blockshield-explorer.tgz
$ cd blockshield-explorer
```

3) 데이터베이스 설정

▶ PostgreSQL 데이터베이스 설정을 업데이트하도록 explorerconfig.json을 수정하십시오.

```
"postgreSQL": {
    "host": "127.0.0.1",
    "port": "5432",
    "database": "explorer",
    "username": "사용자",
    "passwd":  " 비밀번호"
}
```

3) 데이터베이스 설정

▶ PostgreSQL 데이터베이스 설정을 업데이트하도록 explorerconfig.json을 수정하십시오.

```
$ chmod -R 775 db/
$ cd blockshield-explorer/app/persistence/fabric/postgreSQL/db
$ sudo -u postgres ./createdb.sh
```

4) 블록쉴드 네트워크 정보 설정

```
$ cd blockshield-explorer/app/platform/fabric
```

▶ 패브릭 네트워크 연결 프로파일을 정의하도록 config.json을 수정하십시오.

```
{
    "network-configs": {
        "first-network": {
            "name": "firstnetwork",
            "profile": "./connection-profile/first-network.json",
            "enableAuthentication": false
        }
    },
    "license": "Apache-2.0"
}
```

4) 블록쉴드 네트워크 정보 설정

- "first-network"는 연결 프로필의 이름이며 다른 이름으로 변경할 수 있습니다.
- "name"은 패브릭 네트워크에 부여하려는 이름이며 "name"키 값만 변경할 수 있습니다.
- "프로필"은 연결 프로필의 위치입니다. "프로필"키 값만 변경할 수 있습니다.
- JSON 파일 first-network.json에서 연결 프로파일을 수정하십시오.
- first-network.json 파일에서 "fabric-path"를 fabric 네트워크 디스크 경로로 변경하십시오 :

```
/blockshield-explorer/app/platform/fabric/connection-profile/first-network.json
```

5) 블록쉴드 탐색기 빌드

```
$ cd blockshield-explorer
$ npm install
$ cd blockshield-explorer/app/test
$ npm install
$ npm run test
$ cd client/
$ npm install
$ npm run test:ci -- -u --coverage
$ npm run build
```

6) 블록쉴드 탐색기 실행

- 동기화 특성을 업데이트하도록 explorerconfig.json 수정
- 동기화 유형 (로컬 또는 호스트), 플랫폼, blocksSyncTime (분) 세부 사항.

▶ 동기화 프로세스 구성
- 동기화 프로세스가 다른 위치에서 실행중인 경우 Explorer explorerconfig.json에서 동일한 구성을 확인하십시오

```
" sync " : {
    " type " : " host " (또는  local)
}
```

1) 탐색기 엔진 start

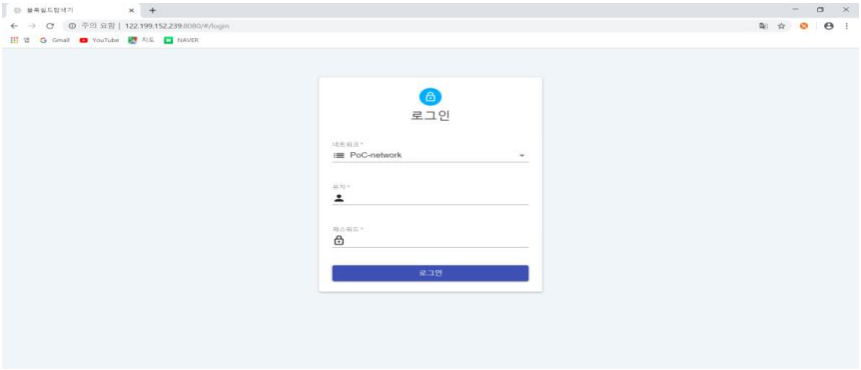
```
$ cd blockshield-explorer/
$ ./start.sh
```

2) 탐색기 엔진 stop

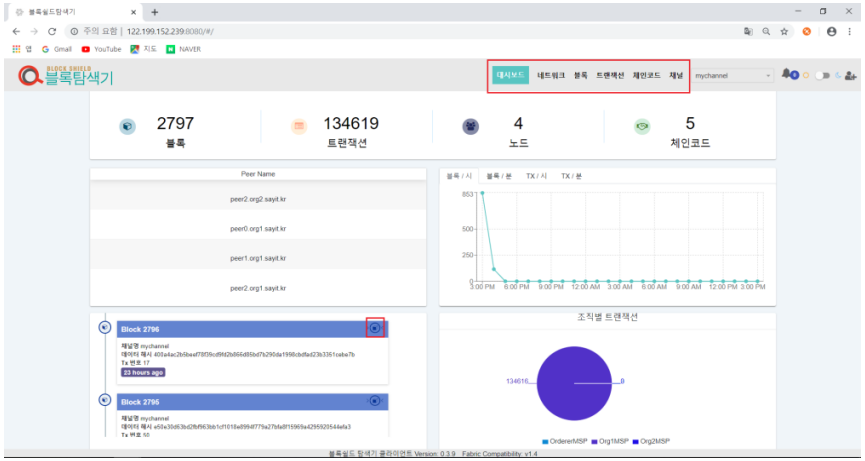
```
$ cd blockshield-explorer/
$ ./stop.sh
```

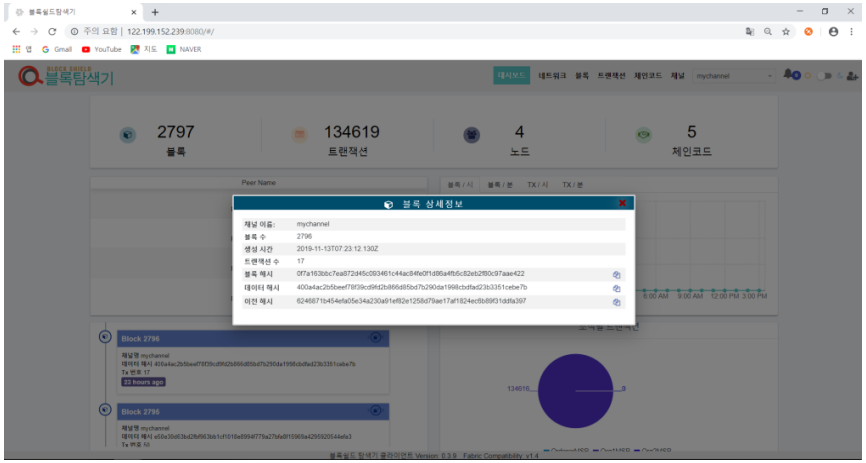
7) 블록쉴드 탐색기 인터페이스 - 로그인

▶ 처음 Build 후 접속하는 관리자계정의 ID는 'admin', 패스워드는 'adminpw' 입니다.



7) 블록쉴드 탐색기 인터페이스 - 대시보드





1) Yarn 설치

```
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
$ sudo apt update && sudo apt install yarn
$ sudo apt update && sudo apt install --no-install-recommends yarn
```

2) 관리툴 빌드

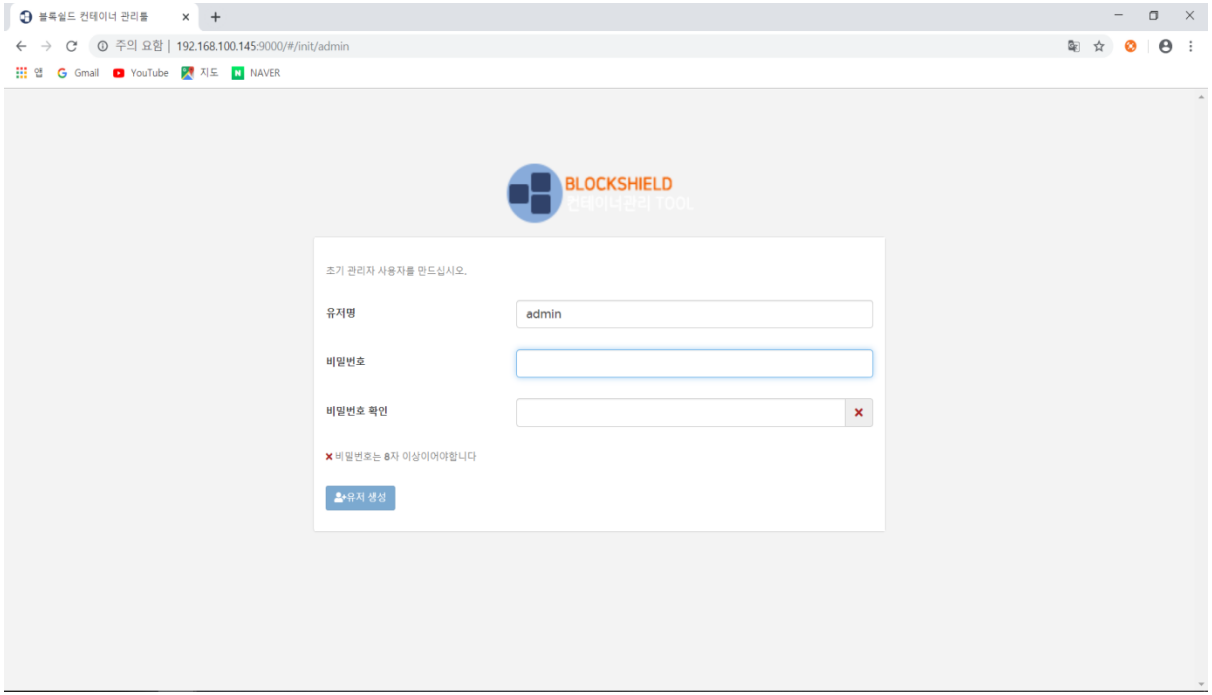
```
$ tar xvfz blockshield_tainer.tgz
$ cd blockshield_tainer
$ yarn
$ yarn start
```

```
$ netstat –lntp
```

▶ '9000' 포트가 열린 것을 확인

3) 컨테이너 관리툴 웹 UI-로그인

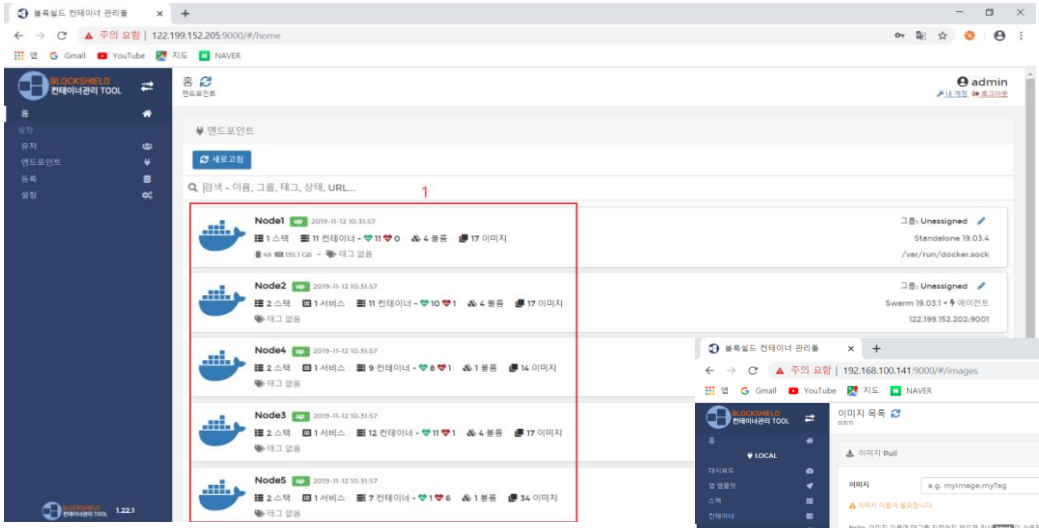
▶ http : // localhost : 9000 로 액세스

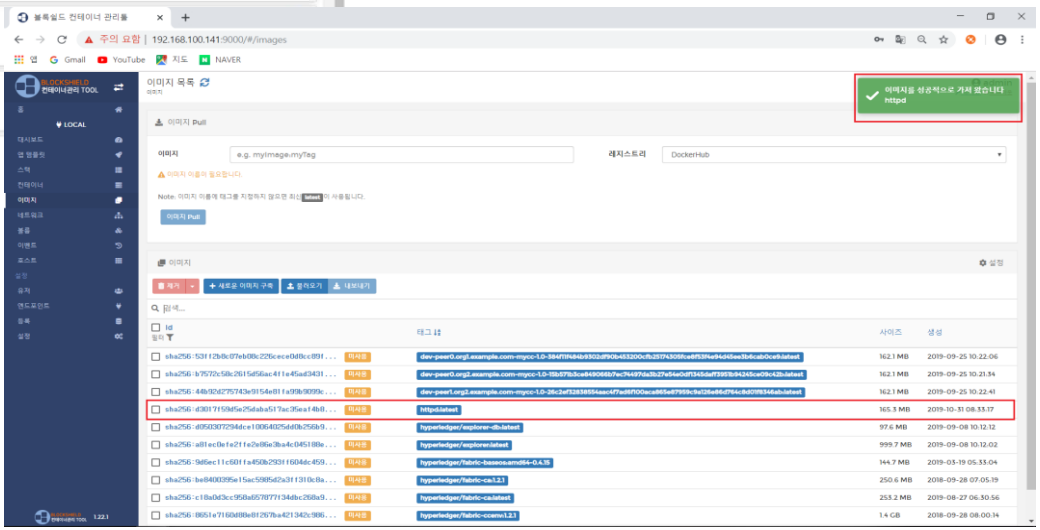


4. 설치 및 실행

4.3 컨테이너 관리 툴 구성

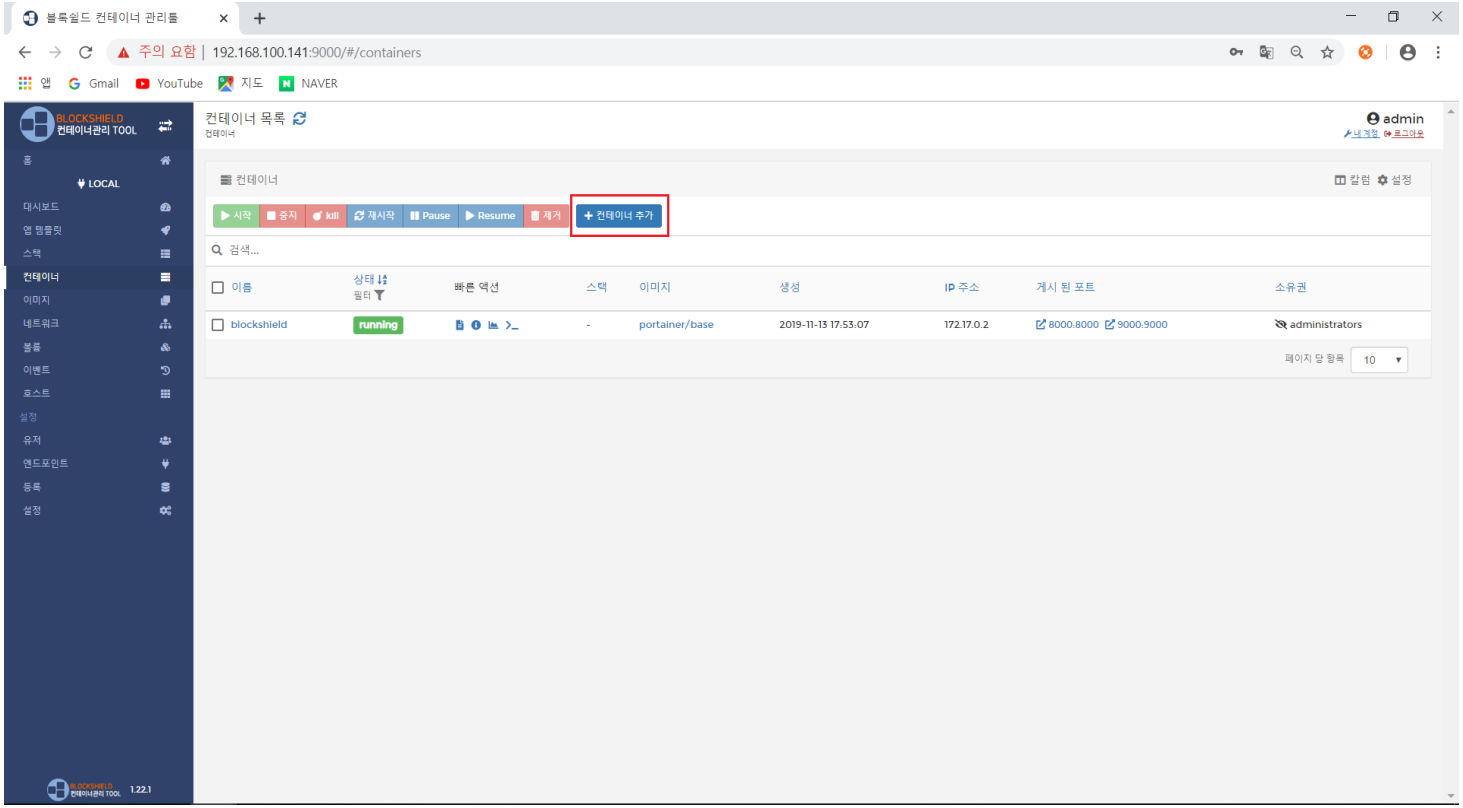
3) 컨테이너 관리툴 웹 UI-대시보드





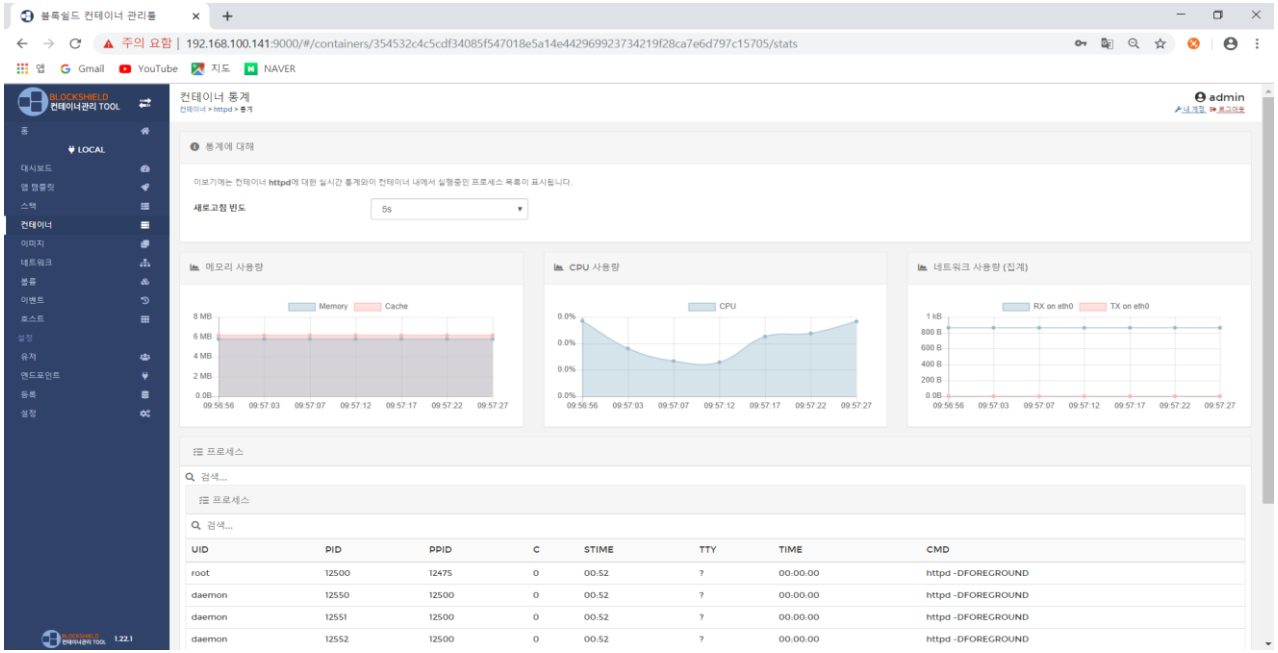
3) 컨테이너 관리툴 웹 UI-컨테이너 관리 기능

▶ 원격의 노드 컨테이너들을 관리할 수 있도록 관리도구에서 '정지-시작-멈춤-재시작' 등의 관리 기능 제공



3) 컨테이너 관리툴 웹 UI-컨테이너 상태 및 통계 모니터링

- 자원 사용량 (메모리, CPU, 네트워크 Input / Output) : 최소 5초 단위로 갱신
- 해당 컨테이너 로그
- 해당 컨테이너 쉘프롬프트 연결
- 컨테이너의 상세 정보



Open Source Software Installation & Application Guide