



BLOCKO

COINSTACK 설치 메뉴얼

(주)블로코

www.blocko.io

목차

1	설치준비사항	2
2	설치.....	3
3	환경설정	6
4	구동.....	13
5	구동확인	14

이 문서는 Coystack 설치과정을 설명합니다. 이 문서는 Cent OS 7.x 기준으로 작성되었습니다.



BLOCKO

1 설치준비사항

설치를 위해서는 몇가지 필수적인 요소가 있습니다.

하기 조건의 설치여부와 설정사항을 미리 확인하시기 바랍니다.

✓ Git

✓ Gcc

✓ Glide

✓ JDK

✓ Go

■ GO환경변수 (현재 문서기준)

◆ GOROOT=/usr/local/go

◆ GOBIN=/root/go/bin

◆ GOPATH=/root/go

✓ 네트워크 설정

- 방화벽 비활성화 또는 사용포트에 대한 규칙을 추가하여 네트워크상 방해요소가 없도록 하십시오. 사용포트는 사용자에게 의해 변경이 가능하므로 환경설정 후 확정된 포트를 대상으로 합니다.

2 설치

2.1 COINSTACK 설치

Go 환경까지 구성이 완료되었다면 아래 명령어를 이용하여 'coinstackd'를 다운받습니다.

```
# git clone https://github.com/coinstack/coinstackd $GOPATH/src/github.com/coinstack/coinstackd
```

```
[root@localhost go]# git clone https://github.com/coinstack/coinstackd $GOPATH/src/github.com/coinstack/coinstackd
Cloning into '/root/go/src/github.com/coinstack/coinstackd'...
remote: Counting objects: 504, done.
remote: Compressing objects: 100% (427/427), done.
remote: Total 504 (delta 79), reused 492 (delta 67), pack-reused 0
Receiving objects: 100% (504/504), 2.69 MiB | 584.00 KiB/s, done.
Resolving deltas: 100% (79/79), done.
[root@localhost go]#
```

다운로드가 완료되었다면 해당폴더로 이동하여 인스톨을 실행합니다.

```
# cd $GOPATH/src/github.com/coinstack/coinstackd
```

```
# glide install
```

```
# go install ./cmd/...
```

```
[root@localhost go]# cd $GOPATH/src/github.com/coinstack/coinstackd
[root@localhost coinstackd]# glide install
[INFO] Downloading dependencies. Please wait...
[INFO] --> Fetching github.com/bluele/gcache
[INFO] --> Fetching github.com/btcsuite/btcllog
[INFO] --> Fetching github.com/btcsuite/fastsha256
[INFO] --> Fetching github.com/btcsuite/go-flags
[INFO] --> Fetching github.com/btcsuite/go-socks
[INFO] --> Fetching github.com/btcsuite/golangcrypto
[INFO] --> Fetching github.com/btcsuite/seelog
[INFO] --> Fetching github.com/btcsuite/snappy-go
[INFO] --> Fetching github.com/btcsuite/websocket
[INFO] --> Fetching github.com/btcsuite/winsvc
[INFO] --> Fetching github.com/coinstack/btcutil
[INFO] --> Fetching github.com/coinstack/go-hmacauth
.....
[INFO] --> Exporting golang.org/x/net
[INFO] Replacing existing vendor dependencies
[root@localhost coinstackd]#
[root@localhost coinstackd]# go install ./cmd/...
[root@localhost coinstackd]# cd $GOPATH/bin
[root@localhost bin]# ls
addblock btcctl coinstackd contractctl findcheckpoint gencerts gengenesis glide
[root@localhost bin]#
```

위와 같이 인스톨 결과로 '\$GOPATH/bin' 에 실행파일이 생성됩니다. GO 설치환경에 따라 해당 위치가 달라지므로 유의하여 주시기 바랍니다.

2.2 ETCD 설치

'Coinstack' 은 기본적으로 interval mining (PoW)로 동작합니다. Raft 합의알고리즘을 적용하여 Leader-Base 로 동작하려면 아래의 과정을 통해 ETCD 를 추가로 구성하고, 환경설정에서도 반영하는 과정이 필요합니다. 아래는 ETCD 를 설치하는 과정을 설명합니다. Raft 합의알고리즘을 사용하지 않는 경우 해당 과정은 필요하지 않습니다.

아래의 명령어로 'ETCD'를 다운받고 추가명령어를 실행합니다..

```
# git clone https://github.com/etcd-io/etcd.git $GOPATH/src/github.com/coreos/etcd
[root@kimtest1 coreos]# git clone https://github.com/etcd-io/etcd.git $GOPATH/src/github.com/coreos/etcd
Cloning into '/root/go/src/github.com/coreos/etcd'...
remote: Counting objects: 83776, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 83776 (delta 4), reused 12 (delta 4), pack-reused 83759
Receiving objects: 100% (83776/83776), 44.74 MiB | 8.45 MiB/s, done.
Resolving deltas: 100% (53569/53569), done.
[root@kimtest1 coreos]#

# GOPATH should be set
$ echo $GOPATH
/Users/example/go
$ go get -v go.etcd.io/etcd
[root@localhost ~]# echo $GOPATH
/root/go
[root@localhost ~]# go get -v go.etcd.io/etcd_
Fetching https://go.etcd.io/etcd?go-get=1
Parsing meta tags from https://go.etcd.io/etcd?go-get=1 (status code 200)
get "go.etcd.io/etcd": found meta tag get.metaImport{Prefix:"go.etcd.io/etcd", CS:"git", RepoRoot:"https://github.com/etcd-io/etcd"} at https://go.etcd.io/etcd?go-get=1
go.etcd.io/etcd (download)
go.etcd.io/etcd/vendor/github.com/coreos/go-systemd/daemon
go.etcd.io/etcd/vendor/github.com/coreos/pkg/dlopen
go.etcd.io/etcd/vendor/github.com/coreos/go-systemd/util
go.etcd.io/etcd/vendor/github.com/coreos/go-systemd/journal
go.etcd.io/etcd/vendor/github.com/coreos/pkg/capnslog
go.etcd.io/etcd/vendor/gopkg.in/yaml.v2
go.etcd.io/etcd/vendor/github.com/ghodss/yaml
go.etcd.io/etcd/vendor/github.com/beorn7/perks/quantile
go.etcd.io/etcd/vendor/github.com/golang/protobuf/proto
go.etcd.io/etcd/vendor/github.com/prometheus/client_model/go
go.etcd.io/etcd/vendor/github.com/matttproud/golang_protobuf_extensions/pbutil
go.etcd.io/etcd/vendor/github.com/prometheus/common/internal/bitbucket.org/ww/goautoneg
go.etcd.io/etcd/vendor/github.com/prometheus/common/model
go.etcd.io/etcd/vendor/github.com/prometheus/common/expfmt
```

다운로드가 완료되었다면 해당 폴더로 이동하여 인스톨을 실행합니다.

```
# cd $GOPATH/src/github.com/coreos/etcd
# ./build
```

```
[root@localhost bin]# cd $GOPATH/src/github.com/coreos/etcd
[root@localhost etcd]# ./build
[root@localhost etcd]# cd bin
[root@localhost bin]# ls
etcd  etcdctl
[root@localhost bin]#
```

위와 같이 인스톨 결과로 '\$GOPATH/src/github.com/coreos/etcd /bin' 에 실행파일이 생성됩니다.



BLOCKO

3 환경설정

'Coinstack' 을 구동하기 전 관련한 환경을 설정하는 부분입니다. 환경의 변경시에는 'Coinstack' 을 다시 시작해주시기 바랍니다. 먼저 환경을 구성하기 전 노드의 채굴에 사용할 'wallet address'와 'privatekey' 및 'Genesis Block' 이 필요합니다. 해당 정보를 구하는 방법을 기술합니다.

3.1 'WALLET ADDRESS' 및 'PRIVATEKEY' 생성하기

■ 생성방법

- ◆ 동봉된 프로그램인 'KeyGeneratorTool-0.7.0-jar-with-dependencies.jar' 를 사용합니다.

```
$ java -jar KeyGeneratorTool-0.7.0-jar-with-dependencies.jar -o [생성할 파일명]
```

정상적으로 실행되면 해당 폴더에 다음과 같은 3개의 파일이 생성됩니다.

- [생성할 파일명].key, [생성할 파일명].addr, [생성할 파일명].pubkey

- ◆ 결과 파일을 아래의 명령어로 확인하면 'wallet address'와 'privatekey' 를 볼 수 있습니다. 환경설정에 필요하니 해당 정보를 따로 기억해두시기 바랍니다.

```
$ java -jar KeyGeneratorTool-0.7.0-jar-with-dependencies.jar -q [생성할 파일명].key
```

- PrivateKey: L1oHnQGoFafTHAxj1VafW8yAPPu54sZ2sumUm3nQtRTNukL2X345
- PublicKey:
3f1ebb0689a82ab46f3b2a3683fa0c3d1c474fd3d4c17dd5dc48923e9d2eccc16
- Wallet Addr: 1MkKdcmcp3pid6yMAoDyfrx5o6z69sVog

예제 화면

```
[root@localhost coinstack_node]# java -jar KeyGeneratorTool-0.7.0-jar-with-dependencies.jar -o create
Random generated pk and corresponding pubkey and wallet address is save to each file.
private key : create.key
public key : create.pubKey
wallet addr : create.addr
[root@localhost coinstack_node]# ls -al | grep create
-rw-r--r-- 1 root root 34 7월 26 11:07 create.addr
-rw-r--r-- 1 root root 52 7월 26 11:07 create.key
-rw-r--r-- 1 root root 66 7월 26 11:07 create.pubKey
[root@localhost coinstack_node]# java -jar KeyGeneratorTool-0.7.0-jar-with-dependencies.jar -q create.key
Private Key : L4dsE7EDgfXd3o682LQv5D27K2So4QijKt7EBkBTcmQ7Nlin2Fr8
Public Key : 021c7c24e5c0d4886d86a52d50d2f483a969d3fd01eb0f987b06d83c6983e15db4
Wallet Addr : 1ExQ169fS9BXfVhPgjjhD2LbpgGBBfXmF
[root@localhost coinstack_node]#
```

✓

3.2 'GENESIS BLOCK' 생성하기

■ 개요

블록체인은 기본적으로 이전블록의 Hash 를 포함합니다. 최초의 블록에는 이전블록의 개념이 없기 때문에 Genesis block 이란 특별한 이름으로 부릅니다. 여기에 포함되는 이전블록에 해당하는 값을 생성하여 환경설정에 적용합니다. 이 값의 차이로 인해 노드들은 다른 블록네트워크로 인식하게 됩니다.

■ 생성하기

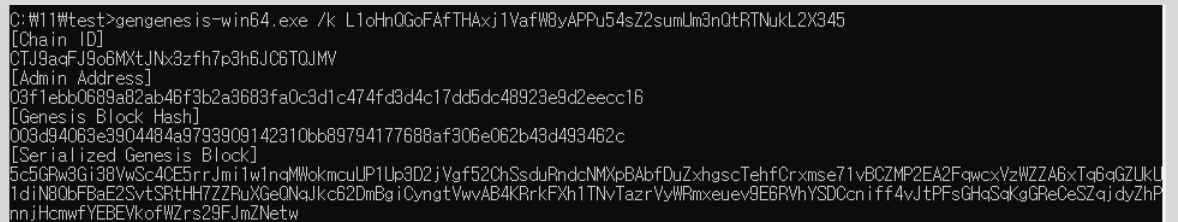
- ◆ 하기 링크를 이용하여 편하신 OS환경에 맞추어 프로그램을 받습니다. 여기서는 Windows 기준으로 실행하였습니다.

- ◆ <http://unit.cloudapp.net/s2/dist/gengenesis/>

- ◆ 위 주소가 접속이 안되는 경우 하기 페이지를 참고하여 생성바랍니다.

<https://github.com/vlamer/genesis-block-generator>

- ◆ 위에서 생성된 'PrivateKey'를 이용하여 'Genesis Block' 를 생성합니다.



```
C:\#11\test>gengenesis-win64.exe /k L1oHn0GoFAfTHAxj1VafW8yAPPu54sZ2sumUm3n0tRTNukL2X345
[Chain ID]
CTJ9adFJ9o6MxtJNx3zf7p3h6JC6TQJMV
[Admin Address]
03f1ebb0689a82ab46f3b2a3683fa0c3d1c474fd3d4c17dd5dc48923e9d2eccc16
[Genesis Block Hash]
009d94063e3904484a9793909142310bb89794177688af306e062b43d493462c
[Serialized Genesis Block]
5c5GRw3G138VwSc4CE5rrJmi1w1ngMWokmcuLP1Up3D2jVgF52ChSsduRncdNMxpBAbfDuZxhgscTehfCrxmse71vBCZMP2EA2FqwcxVzWZZA6xTq6qGZUKL
1diN8ObFBaE2SvtSRtHH7ZZRuXGeQNaJkc62DmBgiCyngtVwvAB4KrkFXh1TNvTazrVyWRmxuev9E6RvYSDCcniff4vJtPFsGHqSgKgGReCeSZqjdyZhP
nnjHcmwfyEBEVkofWZrs29FJmZNetw
```

- ◆ [Serialized Genesis Block] 의 값을 환경에 사용하니 참고하시기 바랍니다.
다른 값도 향후 사용할 수 있으니 따로 보관하시기 바랍니다.

3.3 COINSTACK 환경설정

기본적으로는 'coinstackd'폴더내의 'coinstackd.ini.sample' 파일을 수정하거나 참고하여 작성하시면 됩니다. 단, Raft 사용시에는 'ETCD' 에 대한 환경구성도 추가로 구성하여야 하고, 해당 파일에서도 그에 맞게 '**coordminingendpoint**' 옵션을 활성화하여야 합니다.

하기는 'coinstackd.ini.sample' 파일을 기준으로 변경할 옵션에 대해 설명합니다.

기본기입 내용에서 변경하는 부분은 파란색으로, 유의할 부분은 붉은색으로 표기합니다.

기본적으로 항목 상단에 주석으로 설명이 있으니 참고바랍니다.

■ coinstackd.ini.sample

```
##### Block Parameters #####
# Block 생성여부
generate=1
# mining reward 주소
# sample private key: 생성한 'Privatekey' 를 입력합니다.
miningaddr=생성한 'wallet address' 를 입력합니다.
# 체인의 첫 노드일 경우 bootstrap 동작을 위해 설정. 다른 노드들은 필요없음
privnetbootstrapping=1

##### System Parameters #####
# debugging log level
debuglevel=info
# data directory path : (데이터와 로그가 저장될 위치를 지정합니다.)
datadir="/coinstackd/data"
logdir="/coinstackd/logs"

##### Network Parameters #####
##### Peer Parameters #####
# bitcoin wire protocol port (P2P) : 동기화에 사용할 포트를 입력합니다.
listen=":38333"
# Outbound peer list : (네트워크를 구성할 다른 노드를 {IP}:{wire protocol port} 형태로 입력합니다.
addpeer="192.168.8.101:38333"
addpeer="192.168.8.102:38333"

# enable TLS for peer to peer protocol. disabled by default for backward compatibility
#p2ptls=0
# certificate file path for p2p protocol. required only if p2ptls is on. default is {btcdhome}/psp.cert
#p2pcert=p2p.cert
# key file path for p2p protocol. required only if p2ptls is on. default is {btcdhome}/psp.key
#p2pkey=p2p.key
# skip certificate verification for p2p AND REST api. strongly discouraged in production environment.
#skipp2pverifycert=0

##### REST api Parameters #####
# REST api port. default is 3000 if not configured.
restlisten=3000
```

```
# enable P2P tls
#resttls=0
# certificate file path for REST api. required only if resttls is on
#restcert=.coinstackd/rest.crt
# key file path for REST api. required only if resttls is on
#restkey=.coinstackd/rest.key

##### JSON RPC Parameters #####
# RPC port (client) : JSON RPC 에 사용할 포트와 패스워드, 통신방식을 지정합니다.
rpclisten="127.0.0.1:48333"
# enable TLS for RPC
notls=1
rpcuser=coinstack
rpcpass=coinstack
# certificate file path for RPC. Default filename is rpc.cert. Automatically generated if notls is off (=0)
and cert or key file is missing
#rpccert=.coinstackd/rpc.cert
# key file path for RPC. default filename is rpc.key. Automatically generated if notls is off (=0) and
cert of key file is missing
#rpckey=.coinstackd/rpc.key

##### Consensus Parameters #####
# consensus - mining 주기를 설정한다. 기본은 interval mining 으로 동작. coordminingendpoint 를
설정하면 Raft + interval 로 동작함.
# 기본값: privnetmininginterval=60
# consensus - Raft-variant(distributed lock)
coordminingendpoint="127.0.0.1:5333"

##### Additional Parameters #####
# Block/Tx size 파라미터
# max block size (bytes)
#   blockmaxsize=10000000
# max tx size (bytes)
#   privnetmaxtxsize=5000000
# max data(OP_RETURN) size (bytes)
#   privnetmaxmetasize=1024000
#
# Log 사용 조정 파라미터
```

```
# Logfile size
#   LogMaxSize = 10485760
# Logfile count
#   MaxRolls = 3
#
# genesis block - gengenesis command
#   PrivateNetGenesis      =
Ka2539oj99GQYfMXyhV4zUVemRfjnYcEXHPx3truESZPnfSfuW2EwJ2PWWztDSjsTU5T3M9vnxvmeqwT
vVozuMx7oybBm6Lq6GsJjjCoHH9sYsFZnitzoe4Bibw6vJ6xqGpTaEmS43NKt4nNrcAfUp8ZPJ5LGWxECTXb
Zt9hGWf7BjQdft4xFreAVcvgvooXsc74htfzPzjU3b8Fyb2keVB2WsajB2biJhWfuqtKnQGDhGn6sax6M9UQY
Es1My
#
# node 간 통신 TLS 설정
#   DisableTLS = 1
#
# Memory 설정(orphan 이 많이 생기고 동기화가 실패하는 경우 설정 변경, 단 메모리를
  많이 사용함)
#   BlockMaxSize           = 10485760
#   PrivateNetMaxTxSize    uint32 = 5242880 // less than blocksize * 0.5
#   PrivateNetMaxMetaSize  uint32 = 2097152 // less than PrivateNetMaxTxSize
#
# Watchdog 사용 여부(peer reset 만 하므로 켜 놓는게 좋다)
#   PrivateNetWatchdog = 1
#   WatchdogMessageDeadline = 60 * 360 // network 단절시간에 따라 설정 변경 가능
```

3.4 ETCD 환경설정

ETCD 환경을 구성하기 환경설정입니다. 위한 위에서 언급한대로 Raft 사용시에만 구성합니다. 설치부분에서 설치한 ETCD 폴더중 'bin'폴더로 이동합니다.

```
$ cd $GOPATH/src/github.com/coreos/etcd/bin
```

```
[root@localhost coinstack_node]# cd $GOPATH/src/github.com/coreos/etcd/bin
[root@localhost bin]# ls
etcd  etcdctl
[root@localhost bin]#
```

정상적으로 인스톨되었다면 'etcd' 실행파일이 존재합니다. 파일 확인 후 환경설정용 파일을 생성합니다. 명령어로도 가능하나 내용이 길어져 가독성이 떨어지니 파일을 만들어서 적용하는 것을 권장합니다.

해당 폴더에서 다음 명령어로 파일을 생성합니다. 내용은 ETCD 파일구조를 참고하셔서 생성하시기 바랍니다. #부분은 설명을 위한 주석이니 포함하지 않아도 됩니다. 주요한 사항은 붉은색으로 표기하니 주의하여 주시기 바랍니다.

```
$ vi etcd.conf
```

```
[root@localhost bin]# vi etcd.conf
[root@localhost bin]# cat etcd.conf
#ETCD 멤버명을 입력합니다. 하단 Initial-cluster의 멤버명과 동일하게 적어주시면 됩니다.
name: 'etcd_node1'
#data 및 wal data가 저장되는 폴더를 지정하는 것으로 공란으로 두셔도 됩니다.
```

■ etcd.conf

#ETCD 멤버명을 입력합니다. 하단 Initial-cluster의 멤버명과 동일하게 적어주시면 됩니다.

name: 'etcd_node1'

#data 및 wal data가 저장되는 폴더를 지정하는 것으로 공란으로 두셔도 됩니다.

data-dir:

wal-dir:

#클라이언트 트래픽 수신대기포트를 설정합니다.

#이부분의 포트정보가 coinstack 설정의 **coordminingendpoint** 부분과 동일해야 합니다.

listen-client-urls: <http://0.0.0.0:5333>

#다른 클러스터에 알려줄 클라이언트 정보입니다. 현재 서버 IP를 입력하시면 됩니다.

advertise-client-urls: <http://192.168.8.100:5333>

#피어 트래픽 수신대기포트를 설정합니다.

listen-peer-urls: <http://0.0.0.0:6333>

#다른 클러스터에 알려줄 피어목록입니다. Initial-cluster와 맞도록 설정하시면 됩니다.

initial-advertise-peer-urls: <http://192.168.8.100:6333>

#초기 클러스터 구성정보입니다. {이름=IP:PORT} 형태입니다.

#멤버를 모두 기입하여 구성하고 다른 멤버의 설정파일에도 내용이 동일하게 기입합니다.

initial-cluster:

etcd_node1=http://192.168.8.100:6333,etcd_node2=http://192.168.8.101:6333,etcd_node3=http://192.168.8.102:6333

#클러스터의 토큰 값을 의미하며 변경할 필요는 없습니다.

initial-cluster-token: 'etcd-cluster'

#클러스터의 상태입니다. 초기에는 'new'로 설정하면 됩니다.

#기본은 'new'입니다.

#기존에 존재하는 클러스터에 참여하는 경우 'existing'으로 변경바랍니다.

initial-cluster-state: 'new'



BLOCKO

4 구동

상기의 절차(설치-환경설정)를 모두 마치셨다면 구동할 차례입니다. Raft 를 사용하시는 경우 ETCD 를 먼저 구동하는게 좋습니다. 구동방법은 다음과 같습니다. 되도록 순서대로 구동하시기 바랍니다.

4.1 ETCD 구동

- ETCD 실행파일 위치로 이동합니다.

```
$ cd $GOPATH/src/github.com/coreos/etcd/bin
```

- ETCD 실행파일에 환경파일을 적용시켜 구동합니다.

```
$ ./etcd --config-file etcd.conf
```

* --config-file 옵션으로 etcd 환경설정 파일을 적용시켜 구동합니다.

* background 실행은 screen 또는 & 옵션 등을 이용하시기 바랍니다.

- 예제화면

```
[root@localhost bin]# cd $GOPATH/src/github.com/coreos/etcd/bin
[root@localhost bin]# ./etcd --config-file etcd.conf
2018-07-26 11:18:23.270820 I | etcdmain: Loading server configuration from "etcd.conf".
Other configuration command line flags and environment variables will be ignored if provided.
2018-07-26 11:18:23.270849 I | etcdmain: etcd Version: 3.3.0+git
2018-07-26 11:18:23.270852 I | etcdmain: Git SHA: c5bef4f
2018-07-26 11:18:23.270854 I | etcdmain: Go Version: go1.10.3
```

4.2 COINSTACK 구동

- Coinstack 실행파일 위치로 이동합니다.

(실행파일이 위치한 곳으로 GO설치환경에 따라 다를 수 있습니다.)

```
$ cd $GOBIN
```

- Coinstackd 실행파일에 환경파일을 적용시켜 구동합니다.

```
$ ./coinstackd -C /root/go/src/github.com/coinstack/coinstackd/coinstackd.ini.sample
```

* -C 옵션으로 Coinstack 환경파일을 적용시켜 구동합니다.

* background 실행은 screen 또는 "&> /dev/null &" 를 명령어에 추가하여 구동바랍니다.

- 예제화면

```
[root@localhost bin]# cd $GOBIN
[root@localhost bin]# ./coinstackd -C /root/go/src/github.com/coinstack/coinstackd/coinstackd.ini.sample
11:20:42 2018-07-26 [INF] BTCD: Version 0.12.0-beta
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
```

5 구동확인

5.1 ETCD 구동확인

- ETCD 의 정상 구동상황을 살펴보기 위해서는 아래의 테스트를 수행하시기 바랍니다.
- Curl 명령을 이용하여 정상구동중인지 파악합니다.
\$ curl <http://127.0.0.1:5333/v2/keys/foo>
- 결과값이 반환되면 정상으로 파악하실 수 있습니다. 하기는 예시입니다.
다른 멤버와 통신이 원활히 진행되어 KV store가 구성된 후에 테스트 바랍니다.

```
[root@kimtest1 ~]# curl http://127.0.0.1:5333/v2/keys/foo
{"action": "get", "node": {"key": "/foo", "value": "bar", "modifiedIndex": 9, "createdIndex": 9}}
[root@kimtest1 ~]#
```

5.2 COINSTACK 구동확인

- 기본적으로 지정한 logs 폴더에서 로그를 확인할 수 있습니다.
- 관련하여 블록생성여부를 파악하는 것은 curl로 간단하게 가능합니다.
\$ curl <http://127.0.0.1:3000/blockchain>
- 블록생성간격을 숙지하고 이후 시간으로 조회하면 다음 예시와 같이 블록이 생성된 상황을 확인 가능합니다.

```
[root@kimtest1 ~]# curl http://127.0.0.1:3000/blockchain
{"best_block_hash": "613b7e031de1bbc22462f078dd12374a97a4c34cb019467f0cc072b2e19f005b", "best_height": 145}
[root@kimtest1 ~]# curl http://127.0.0.1:3000/blockchain
{"best_block_hash": "7ceef1fd19e8de9e783fd3c0466979f3af32489d35707dc0463b948e58199ccb", "best_height": 152}
[root@kimtest1 ~]#
```

설치환경에 따라 상이한 문제가 발생할 수 있습니다. 단계별로 이상유무를 확인하시면서 진행하시기 바랍니다.