#說明：##build\_chain.sh 作用--------帮助用户快读搭建FISCO BCOS联盟链

#!/bin/bash ##說明：指定shell脚本解释器的路径

set -e

# default value ##說明：默認參數設置

ca\_file= #CA key

node\_num=1

ip\_file=

ip\_param=

use\_ip\_param=

agency\_array=

group\_array=

ip\_array=

output\_dir=nodes

port\_start=(30300 20200 8545) ##說明：使用端口

state\_type=storage

storage\_type=rocksdb

conf\_path="conf"

bin\_path=

make\_tar=

debug\_log="false"

log\_level="info"

logfile=${PWD}/build.log

listen\_ip="127.0.0.1" ##說明：監聽地址

bcos\_bin\_name=fisco-bcos ##說明：名稱

guomi\_mode=

docker\_mode=

gm\_conf\_path="gmconf/"

current\_dir=$(pwd)

consensus\_type="pbft"

TASSL\_CMD="${HOME}"/.tassl

enable\_parallel=true

auto\_flush="true"

# trans timestamp from seconds to milliseconds ##說明：交易時間戳-秒 到 毫秒

timestamp=$(($(date '+%s')\*1000))

chain\_id=1

compatibility\_version=""

default\_version="2.1.0"

macOS=""

x86\_64\_arch="true"

download\_timeout=60

cdn\_link\_header="https://www.fisco.com.cn/cdn/fisco-bcos/releases/download" ##說明：CDN加速

help() { ##說明：使用幫助

echo $1

cat << EOF

Usage:

-l <IP list> [Required] "ip1:nodeNum1,ip2:nodeNum2" e.g:"192.168.0.1:2,192.168.0.2:3"

-f <IP list file> [Optional] split by line, every line should be "ip:nodeNum agencyName groupList". eg "127.0.0.1:4 agency1 1,2"

-e <FISCO-BCOS binary path> Default download fisco-bcos from GitHub. If set -e, use the binary at the specified location

-o <Output Dir> Default ./nodes/

-p <Start Port> Default 30300,20200,8545 means p2p\_port start from 30300, channel\_port from 20200, jsonrpc\_port from 8545

-i <Host ip> Default 127.0.0.1. If set -i, listen 0.0.0.0

-v <FISCO-BCOS binary version> Default get version from https://github.com/FISCO-BCOS/FISCO-BCOS/releases. If set use specificd version binary

-s <DB type> Default rocksdb. Options can be rocksdb / mysql / external, rocksdb is recommended

-d <docker mode> Default off. If set -d, build with docker

-c <Consensus Algorithm> Default PBFT. If set -c, use Raft

-m <MPT State type> Default storageState. if set -m, use mpt state

-C <Chain id> Default 1. Can set uint.

-g <Generate guomi nodes> Default no

-z <Generate tar packet> Default no

-t <Cert config file> Default auto generate

-T <Enable debug log> Default off. If set -T, enable debug log

-F <Disable log auto flush> Default on. If set -F, disable log auto flush

-h Help

e.g

$0 -l "127.0.0.1:4"

EOF

exit 0

}

LOG\_WARN() ##說明：配置日誌（警示）

{

local content=${1}

echo -e "\033[31m[WARN] ${content}\033[0m"

}

LOG\_INFO() ##說明：配置日誌（信息）

{

local content=${1}

echo -e "\033[32m[INFO] ${content}\033[0m"

}

get\_value()

{

local var\_name=${1}

var\_name=var\_${var\_name//./}

local res=$(eval echo '$'"${var\_name}")

echo ${res}

}

set\_value()

{

local var\_name=${1}

var\_name=var\_${var\_name//./}

local var\_value=${2}

eval "${var\_name}=${var\_value}"

}

exit\_with\_clean() ##配置退出並清理

{

local content=${1}

echo -e "\033[31m[ERROR] ${content}\033[0m"

if [ -d "${output\_dir}" ];then

rm -rf ${output\_dir}

fi

exit 1

}

parse\_params() ##說明：解析參數

{

while getopts "f:l:o:p:e:t:v:s:C:iczhgmTFd" option;do

case $option in

f) ip\_file=$OPTARG

use\_ip\_param="false"

;;

l) ip\_param=$OPTARG

use\_ip\_param="true"

;;

o) output\_dir=$OPTARG;;

i) listen\_ip="0.0.0.0";;

v) compatibility\_version="$OPTARG";;

p) port\_start=(${OPTARG//,/ })

if [ ${#port\_start[@]} -ne 3 ];then LOG\_WARN "start port error. e.g: 30300,20200,8545" && exit 1;fi

;;

e) bin\_path=$OPTARG;;

m) state\_type=mpt;;

s) storage\_type=$OPTARG

if [ -z "${storage\_type}" ];then

LOG\_WARN "${storage\_type} is not supported storage."

exit 1;

fi

;;

t) CertConfig=$OPTARG;;

c) consensus\_type="raft";;

C) chain\_id=$OPTARG

if [ -z $(grep '^[[:digit:]]\*$' <<< "${chain\_id}") ];then

LOG\_WARN "${chain\_id} is not a positive integer."

exit 1;

fi

;;

T) debug\_log="true"

log\_level="debug"

;;

F) auto\_flush="false";;

z) make\_tar="yes";;

g) guomi\_mode="yes";;

d) docker\_mode="yes"

if [ ! -z "${macOS}" ];then LOG\_WARN "Docker desktop of macOS can't support docker mode of FISCO BCOS!" && exit 1;fi

;;

h) help;;

esac

done

}

print\_result() ##說明：輸出結果

{

echo "================================================================"

[ -z ${docker\_mode} ] && [ -f "${bin\_path}" ] && LOG\_INFO "FISCO-BCOS Path : $bin\_path"

[ ! -z ${docker\_mode} ] && LOG\_INFO "Docker tag : latest"

[ ! -z $ip\_file ] && LOG\_INFO "IP List File : $ip\_file"

# [ ! -z $ip\_file ] && LOG\_INFO -e "Agencies/groups : ${#agency\_array[@]}/${#groups[@]}"

LOG\_INFO "Start Port : ${port\_start[\*]}"

LOG\_INFO "Server IP : ${ip\_array[\*]}"

LOG\_INFO "State Type : ${state\_type}"

LOG\_INFO "RPC listen IP : ${listen\_ip}"

LOG\_INFO "Output Dir : ${output\_dir}"

LOG\_INFO "CA Key Path : $ca\_file"

[ ! -z $guomi\_mode ] && LOG\_INFO "Guomi mode : $guomi\_mode"

echo "================================================================"

if [ "${listen\_ip}" == "127.0.0.1" ];then LOG\_WARN "RPC listens 127.0.0.1 will cause nodes' JSON-RPC and Channel service to be inaccessible form other machines";fi

LOG\_INFO "Execute the following command to get FISCO-BCOS console"

echo " bash <(curl -s https://raw.githubusercontent.com/FISCO-BCOS/console/master/tools/download\_console.sh)"

echo "================================================================"

LOG\_INFO "All completed. Files in ${output\_dir}"

}

check\_env() { ##說明：檢查環境

[ ! -z "$(openssl version | grep 1.0.2)" ] || [ ! -z "$(openssl version | grep 1.1)" ] || [ ! -z "$(openssl version | grep reSSL)" ] || {

echo "please install openssl!"

#echo "download openssl from https://www.openssl.org."

echo "use \"openssl version\" command to check."

exit 1

}

if [ ! -z "$(openssl version | grep reSSL)" ];then

export PATH="/usr/local/opt/openssl/bin:$PATH"

fi

if [ "$(uname)" == "Darwin" ];then

macOS="macOS"

fi

if [ "$(uname -m)" != "x86\_64" ];then

x86\_64\_arch="false"

fi

}

# TASSL env ##說明：配置SSL

check\_and\_install\_tassl()

{

if [ ! -f "${HOME}/.tassl" ];then

curl -LO https://github.com/FISCO-BCOS/LargeFiles/raw/master/tools/tassl.tar.gz

LOG\_INFO "Downloading tassl binary ..."

tar zxvf tassl.tar.gz

chmod u+x tassl

mv tassl ${HOME}/.tassl

fi

}

check\_name() { #說明：名稱規范檢查

local name="$1"

local value="$2"

[[ "$value" =~ ^[a-zA-Z0-9.\_-]+$ ]] || {

exit\_with\_clean "$name name [$value] invalid, it should match regex: ^[a-zA-Z0-9.\_-]+\$"

}

}

file\_must\_exists() { #說明：檢查文件

if [ ! -f "$1" ]; then

exit\_with\_clean "$1 file does not exist, please check!"

fi

}

dir\_must\_exists() { #說明：檢查目錄

if [ ! -d "$1" ]; then

exit\_with\_clean "$1 DIR does not exist, please check!"

fi

}

dir\_must\_not\_exists() { #說明：檢查目錄

if [ -e "$1" ]; then

LOG\_WARN "$1 DIR exists, please clean old DIR!"

exit 1

fi

}

gen\_chain\_cert() { ##說明：生成鏈證書

local path="${1}"

name=$(basename "$path")

echo "$path --- $name"

dir\_must\_not\_exists "$path"

check\_name chain "$name"

chaindir=$path

mkdir -p $chaindir

openssl genrsa -out $chaindir/ca.key 2048

openssl req -new -x509 -days 3650 -subj "/CN=$name/O=fisco-bcos/OU=chain" -key $chaindir/ca.key -out $chaindir/ca.crt

mv cert.cnf $chaindir

}

gen\_agency\_cert() { ##說明：生成機構證書

local chain="${1}"

local agencypath="${2}"

name=$(basename "$agencypath")

dir\_must\_exists "$chain"

file\_must\_exists "$chain/ca.key"

check\_name agency "$name"

agencydir=$agencypath

dir\_must\_not\_exists "$agencydir"

mkdir -p $agencydir

openssl genrsa -out $agencydir/agency.key 2048

openssl req -new -sha256 -subj "/CN=$name/O=fisco-bcos/OU=agency" -key $agencydir/agency.key -config $chain/cert.cnf -out $agencydir/agency.csr

openssl x509 -req -days 3650 -sha256 -CA $chain/ca.crt -CAkey $chain/ca.key -CAcreateserial\

-in $agencydir/agency.csr -out $agencydir/agency.crt -extensions v4\_req -extfile $chain/cert.cnf

cp $chain/ca.crt $chain/cert.cnf $agencydir/

rm -f $agencydir/agency.csr

echo "build $name agency cert successful!"

}

gen\_cert\_secp256k1() { ##說明：生成證書secp256k1

agpath="$1"

certpath="$2"

name="$3"

type="$4"

openssl ecparam -out $certpath/${type}.param -name secp256k1

openssl genpkey -paramfile $certpath/${type}.param -out $certpath/${type}.key

openssl pkey -in $certpath/${type}.key -pubout -out $certpath/${type}.pubkey

openssl req -new -sha256 -subj "/CN=${name}/O=fisco-bcos/OU=${type}" -key $certpath/${type}.key -config $agpath/cert.cnf -out $certpath/${type}.csr

openssl x509 -req -days 3650 -sha256 -in $certpath/${type}.csr -CAkey $agpath/agency.key -CA $agpath/agency.crt\

-force\_pubkey $certpath/${type}.pubkey -out $certpath/${type}.crt -CAcreateserial -extensions v3\_req -extfile $agpath/cert.cnf

# openssl ec -in $certpath/${type}.key -outform DER | tail -c +8 | head -c 32 | xxd -p -c 32 | cat >$certpath/${type}.private

cat ${agpath}/agency.crt >> $certpath/${type}.crt

rm -f $certpath/${type}.csr $certpath/${type}.pubkey $certpath/${type}.param

}

gen\_cert() { ##說明：生成證書

if [ "" == "$(openssl ecparam -list\_curves 2>&1 | grep secp256k1)" ]; then

exit\_with\_clean "openssl don't support secp256k1, please upgrade openssl!"

fi

agpath="${1}"

agency=$(basename "$agpath")

ndpath="${2}"

local cert\_name="${3}"

node=$(basename "$ndpath")

dir\_must\_exists "$agpath"

file\_must\_exists "$agpath/agency.key"

check\_name agency "$agency"

dir\_must\_not\_exists "$ndpath"

check\_name node "$node"

mkdir -p $ndpath

gen\_cert\_secp256k1 "$agpath" "$ndpath" "$node" "${cert\_name}"

#nodeid is pubkey

openssl ec -in $ndpath/node.key -text -noout | sed -n '7,11p' | tr -d ": \n" | awk '{print substr($0,3);}' | cat >$ndpath/node.nodeid

# openssl x509 -serial -noout -in $ndpath/node.crt | awk -F= '{print $2}' | cat >$ndpath/node.serial

cp $agpath/ca.crt $ndpath

cd $ndpath

}

generate\_gmsm2\_param() ##說明：生成國密gmsm2參數

{

local output=$1

cat << EOF > ${output}

-----BEGIN EC PARAMETERS-----

BggqgRzPVQGCLQ==

-----END EC PARAMETERS-----

EOF

}

gen\_chain\_cert\_gm() { ##說明：生成鏈（國密）證書gm

local path="${1}"

name=$(basename "$path")

echo "$path --- $name"

dir\_must\_not\_exists "$path"

check\_name chain "$name"

chaindir=$path

mkdir -p $chaindir

generate\_gmsm2\_param "gmsm2.param"

$TASSL\_CMD genpkey -paramfile gmsm2.param -out $chaindir/gmca.key

$TASSL\_CMD req -config gmcert.cnf -x509 -days 3650 -subj "/CN=$name/O=fiscobcos/OU=chain" -key $chaindir/gmca.key -extensions v3\_ca -out $chaindir/gmca.crt

cp gmcert.cnf gmsm2.param $chaindir

if $(cp gmcert.cnf gmsm2.param $chaindir)

then

echo "build chain ca succussful!"

else

echo "please input at least Common Name!"

fi

}

gen\_agency\_cert\_gm() { ##說明：生成機構（國密）證書gm

local chain="${1}"

local agencypath="${2}"

name=$(basename "$agencypath")

dir\_must\_exists "$chain"

file\_must\_exists "$chain/gmca.key"

check\_name agency "$name"

agencydir=$agencypath

dir\_must\_not\_exists "$agencydir"

mkdir -p $agencydir

$TASSL\_CMD genpkey -paramfile $chain/gmsm2.param -out $agencydir/gmagency.key

$TASSL\_CMD req -new -subj "/CN=$name/O=fiscobcos/OU=agency" -key $agencydir/gmagency.key -config $chain/gmcert.cnf -out $agencydir/gmagency.csr

$TASSL\_CMD x509 -req -CA $chain/gmca.crt -CAkey $chain/gmca.key -days 3650 -CAcreateserial -in $agencydir/gmagency.csr -out $agencydir/gmagency.crt -extfile $chain/gmcert.cnf -extensions v3\_agency\_root

cp $chain/gmca.crt $chain/gmcert.cnf $chain/gmsm2.param $agencydir/

rm -f $agencydir/gmagency.csr

}

gen\_node\_cert\_with\_extensions\_gm() { ##說明：生成節點證書及擴展（國密）gm

capath="$1"

certpath="$2"

name="$3"

type="$4"

extensions="$5"

$TASSL\_CMD genpkey -paramfile $capath/gmsm2.param -out $certpath/gm${type}.key

$TASSL\_CMD req -new -subj "/CN=$name/O=fiscobcos/OU=agency" -key $certpath/gm${type}.key -config $capath/gmcert.cnf -out $certpath/gm${type}.csr

$TASSL\_CMD x509 -req -CA $capath/gmagency.crt -CAkey $capath/gmagency.key -days 3650 -CAcreateserial -in $certpath/gm${type}.csr -out $certpath/gm${type}.crt -extfile $capath/gmcert.cnf -extensions $extensions

rm -f $certpath/gm${type}.csr

}

gen\_node\_cert\_gm() { ##說明：生成節點（國密）證書gm

if [ "" = "$(openssl ecparam -list\_curves 2>&1 | grep secp256k1)" ]; then

exit\_with\_clean "openssl don't support secp256k1, please upgrade openssl!"

fi

agpath="${1}"

agency=$(basename "$agpath")

ndpath="${2}"

node=$(basename "$ndpath")

dir\_must\_exists "$agpath"

file\_must\_exists "$agpath/gmagency.key"

check\_name agency "$agency"

mkdir -p $ndpath

dir\_must\_exists "$ndpath"

check\_name node "$node"

mkdir -p $ndpath

gen\_node\_cert\_with\_extensions\_gm "$agpath" "$ndpath" "$node" node v3\_req

gen\_node\_cert\_with\_extensions\_gm "$agpath" "$ndpath" "$node" ennode v3enc\_req

#nodeid is pubkey ##說明：nodeid是公鑰

$TASSL\_CMD ec -in $ndpath/gmnode.key -text -noout | sed -n '7,11p' | sed 's/://g' | tr "\n" " " | sed 's/ //g' | awk '{print substr($0,3);}' | cat > $ndpath/gmnode.nodeid

#serial ##說明：gmnode序列

if [ "" != "$($TASSL\_CMD version | grep 1.0.2)" ];

then

$TASSL\_CMD x509 -text -in $ndpath/gmnode.crt | sed -n '5p' | sed 's/://g' | tr "\n" " " | sed 's/ //g' | sed 's/[a-z]/\u&/g' | cat > $ndpath/gmnode.serial

else

$TASSL\_CMD x509 -text -in $ndpath/gmnode.crt | sed -n '4p' | sed 's/ //g' | sed 's/.\*(0x//g' | sed 's/)//g' |sed 's/[a-z]/\u&/g' | cat > $ndpath/gmnode.serial

fi

cp $agpath/gmca.crt $agpath/gmagency.crt $ndpath

cd $ndpath

}

generate\_config\_ini() ##說明：系統配置初始化(rpc/p2p/證書黑名單/證書白名單/群組/網絡安全/存儲安全/鏈/版本兼容性/日誌)

{

local output=${1}

local ip=${2}

local offset=$(get\_value ${ip//./}\_count)

local node\_groups=(${3//,/ })

local prefix=""

if [ -n "$guomi\_mode" ]; then

prefix="gm"

fi

cat << EOF > ${output}

[rpc]

listen\_ip=${listen\_ip}

channel\_listen\_port=$(( offset + port\_start[1] ))

jsonrpc\_listen\_port=$(( offset + port\_start[2] ))

[p2p]

listen\_ip=0.0.0.0

listen\_port=$(( offset + port\_start[0] ))

;enable\_compress=true

; nodes to connect

$ip\_list

[certificate\_blacklist]

; crl.0 should be nodeid, nodeid's length is 128

;crl.0=

[certificate\_whitelist]

; cal.0 should be nodeid, nodeid's length is 128

;cal.0=

[group]

group\_data\_path=data/

group\_config\_path=${conf\_path}/

[network\_security]

; directory the certificates located in

data\_path=${conf\_path}/

; the node private key file

key=${prefix}node.key

; the node certificate file

cert=${prefix}node.crt

; the ca certificate file

ca\_cert=${prefix}ca.crt

[storage\_security]

enable=false

key\_manager\_ip=

key\_manager\_port=

cipher\_data\_key=

[chain]

id=${chain\_id}

[compatibility]

; supported\_version should nerver be changed

supported\_version=${compatibility\_version}

[log]

enable=true

log\_path=./log

; info debug trace

level=${log\_level}

; MB

max\_log\_file\_size=200

flush=${auto\_flush}

log\_flush\_threshold=100

EOF

}

generate\_group\_genesis() ##說明：生成群組（共識初始化）

{

local output=$1

local index=$2

local node\_list=$3

cat << EOF > ${output}

[consensus]

; consensus algorithm type, now support PBFT(consensus\_type=pbft) and Raft(consensus\_type=raft)

consensus\_type=${consensus\_type}

; the max number of transactions of a block

max\_trans\_num=1000

; the node id of consensusers

${node\_list}

[state]

; support mpt/storage

type=${state\_type}

[tx]

; transaction gas limit

gas\_limit=300000000

[group]

id=${index}

timestamp=${timestamp}

EOF

}

function generate\_group\_ini() ##說明：生成群組（初始化）

{

local output="${1}"

cat << EOF > ${output}

[consensus]

; the ttl for broadcasting pbft message

;ttl=2

; min block generation time(ms), the max block generation time is 1000 ms

;min\_block\_generation\_time=500

;enable\_dynamic\_block\_size=true

[storage]

; storage db type, rocksdb / mysql / external, rocksdb is recommended

type=${storage\_type}

; max cache memeory, MB

max\_capacity=32

max\_forward\_block=10

; only for external

max\_retry=60

topic=DB

; only for mysql

db\_ip=127.0.0.1

db\_port=3306

db\_username=

db\_passwd=

db\_name=

[tx\_pool]

limit=150000

[tx\_execute]

enable\_parallel=${enable\_parallel}

EOF

}

generate\_cert\_conf() ##說明：生成證書配置(ca/req/v3\_req/ v4\_req)

{

local output=$1

cat << EOF > ${output}

[ca]

default\_ca=default\_ca

[default\_ca]

default\_days = 365

default\_md = sha256

[req]

distinguished\_name = req\_distinguished\_name

req\_extensions = v3\_req

[req\_distinguished\_name]

countryName = CN

countryName\_default = CN

stateOrProvinceName = State or Province Name (full name)

stateOrProvinceName\_default =GuangDong

localityName = Locality Name (eg, city)

localityName\_default = ShenZhen

organizationalUnitName = Organizational Unit Name (eg, section)

organizationalUnitName\_default = fisco-bcos

commonName = Organizational commonName (eg, fisco-bcos)

commonName\_default = fisco-bcos

commonName\_max = 64

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v4\_req ]

basicConstraints = CA:TRUE

EOF

}

generate\_script\_template() ##說明：生成script模板

{

local filepath=$1

mkdir -p $(dirname $filepath)

cat << EOF > "${filepath}"

#!/bin/bash

SHELL\_FOLDER=\$(cd \$(dirname \$0);pwd)

LOG\_ERROR() {

content=\${1}

echo -e "\033[31m[ERROR] \${content}\033[0m"

}

LOG\_INFO() {

content=\${1}

echo -e "\033[32m[INFO] \${content}\033[0m"

}

EOF

chmod +x ${filepath}

}

generate\_cert\_conf\_gm() ##說明：生成證書配置gm(new\_oids/ca/CA\_default/policy\_match/policy\_anything/reqreq\_distinguished\_name/usr\_cert/v3\_req/v3enc\_req/v3\_agency\_root/v3\_ca)

{

local output=$1

cat << EOF > ${output}

HOME = .

RANDFILE = $ENV::HOME/.rnd

oid\_section = new\_oids

[ new\_oids ]

tsa\_policy1 = 1.2.3.4.1

tsa\_policy2 = 1.2.3.4.5.6

tsa\_policy3 = 1.2.3.4.5.7

####################################################################

[ ca ]

default\_ca = CA\_default # The default ca section

####################################################################

[ CA\_default ]

dir = ./demoCA # Where everything is kept

certs = $dir/certs # Where the issued certs are kept

crl\_dir = $dir/crl # Where the issued crl are kept

database = $dir/index.txt # database index file.

#unique\_subject = no # Set to 'no' to allow creation of

# several ctificates with same subject.

new\_certs\_dir = $dir/newcerts # default place for new certs.

certificate = $dir/cacert.pem # The CA certificate

serial = $dir/serial # The current serial number

crlnumber = $dir/crlnumber # the current crl number

# must be commented out to leave a V1 CRL

crl = $dir/crl.pem # The current CRL

private\_key = $dir/private/cakey.pem # The private key

RANDFILE = $dir/private/.rand # private random number file

x509\_extensions = usr\_cert # The extentions to add to the cert

name\_opt = ca\_default # Subject Name options

cert\_opt = ca\_default # Certificate field options

default\_days = 365 # how long to certify for

default\_crl\_days= 30 # how long before next CRL

default\_md = default # use public key default MD

preserve = no # keep passed DN ordering

policy = policy\_match

[ policy\_match ]

countryName = match

stateOrProvinceName = match

organizationName = match

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ policy\_anything ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

####################################################################

[ req ]

default\_bits = 2048

default\_md = sm3

default\_keyfile = privkey.pem

distinguished\_name = req\_distinguished\_name

x509\_extensions = v3\_ca # The extentions to add to the self signed cert

string\_mask = utf8only

# req\_extensions = v3\_req # The extensions to add to a certificate request

[ req\_distinguished\_name ]

countryName = CN

countryName\_default = CN

stateOrProvinceName = State or Province Name (full name)

stateOrProvinceName\_default =GuangDong

localityName = Locality Name (eg, city)

localityName\_default = ShenZhen

organizationalUnitName = Organizational Unit Name (eg, section)

organizationalUnitName\_default = fisco

commonName = Organizational commonName (eg, fisco)

commonName\_default = fisco

commonName\_max = 64

[ usr\_cert ]

basicConstraints=CA:FALSE

nsComment = "OpenSSL Generated Certificate"

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid,issuer

[ v3\_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature

[ v3enc\_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE

keyUsage = keyAgreement, keyEncipherment, dataEncipherment

[ v3\_agency\_root ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer

basicConstraints = CA:true

keyUsage = cRLSign, keyCertSign

[ v3\_ca ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer

basicConstraints = CA:true

keyUsage = cRLSign, keyCertSign

EOF

}

generate\_node\_scripts() ##說明：生成節點的script

{

local output=$1

local docker\_tag="v${compatibility\_version}"

generate\_script\_template "$output/start.sh"

local ps\_cmd="\$(ps aux|grep \${fisco\_bcos}|grep -v grep|awk '{print \$2}')"

local start\_cmd="nohup \${fisco\_bcos} -c config.ini >>nohup.out 2>&1 &"

local stop\_cmd="kill \${node\_pid}"

local pid="pid"

local log\_cmd="tail -n20 nohup.out"

local check\_success="\$(${log\_cmd} | grep running)"

if [ ! -z ${docker\_mode} ];then

ps\_cmd="\$(docker ps |grep \${SHELL\_FOLDER//\//} | grep -v grep|awk '{print \$1}')"

start\_cmd="docker run -d --rm --name \${SHELL\_FOLDER//\//} -v \${SHELL\_FOLDER}:/data --network=host -w=/data fiscoorg/fiscobcos:${docker\_tag} -c config.ini"

stop\_cmd="docker kill \${node\_pid} 2>/dev/null"

pid="container id"

log\_cmd="tail -n20 \$(docker inspect --format='{{.LogPath}}' \${SHELL\_FOLDER//\//})"

check\_success="success"

fi

cat << EOF >> "$output/start.sh"

fisco\_bcos=\${SHELL\_FOLDER}/../${bcos\_bin\_name}

cd \${SHELL\_FOLDER}

node=\$(basename \${SHELL\_FOLDER})

node\_pid=${ps\_cmd}

if [ ! -z \${node\_pid} ];then

echo " \${node} is running, ${pid} is \$node\_pid."

exit 0

else

${start\_cmd}

sleep 1.5

fi

try\_times=4

i=0

while [ \$i -lt \${try\_times} ]

do

node\_pid=${ps\_cmd}

success\_flag=${check\_success}

if [[ ! -z \${node\_pid} && ! -z "\${success\_flag}" ]];then

echo -e "\033[32m \${node} start successfully\033[0m"

exit 0

fi

sleep 0.5

((i=i+1))

done

echo -e "\033[31m Exceed waiting time. Please try again to start \${node} \033[0m"

${log\_cmd}

exit 1

EOF

generate\_script\_template "$output/stop.sh"

cat << EOF >> "$output/stop.sh"

fisco\_bcos=\${SHELL\_FOLDER}/../${bcos\_bin\_name}

node=\$(basename \${SHELL\_FOLDER})

node\_pid=${ps\_cmd}

try\_times=10

i=0

if [ -z \${node\_pid} ];then

echo " \${node} isn't running."

exit 0

fi

[ ! -z \${node\_pid} ] && ${stop\_cmd} > /dev/null

while [ \$i -lt \${try\_times} ]

do

sleep 0.6

node\_pid=${ps\_cmd}

if [ -z \${node\_pid} ];then

echo -e "\033[32m stop \${node} success.\033[0m"

exit 0

fi

((i=i+1))

done

echo " Exceed maximum number of retries. Please try again to stop \${node}"

exit 1

EOF

generate\_script\_template "$output/scripts/load\_new\_groups.sh"

cat << EOF >> "$output/scripts/load\_new\_groups.sh"

cd \${SHELL\_FOLDER}/../

NODE\_FOLDER=\$(pwd)

fisco\_bcos=\${NODE\_FOLDER}/../${bcos\_bin\_name}

node=\$(basename \${NODE\_FOLDER})

node\_pid=${ps\_cmd}

if [ ! -z \${node\_pid} ];then

echo "\${node} is trying to load new groups. Check log for more information."

touch config.ini.append\_group

exit 0

else

echo "\${node} is not running, use start.sh to start all group directlly."

fi

EOF

generate\_script\_template "$output/scripts/reload\_whitelist.sh"

cat << EOF >> "$output/scripts/reload\_whitelist.sh"

check\_cal\_line()

{

line=\$1;

if [[ \$line =~ cal.[0-9]\*=[0-9A-Fa-f]{128,128}\$ ]]; then

echo "true";

else

echo "false";

fi

}

check\_cal\_lines()

{

# print Illegal line

config\_file=\$1

error="false"

for line in \$(grep -v "^[ ]\*[;]" \$config\_file | grep "cal\."); do

if [[ "true" != \$(check\_cal\_line \$line) ]]; then

LOG\_ERROR "Illigal whitelist line: \$line"

error="true"

fi

done

if [[ "true" == \$error ]]; then

LOG\_ERROR "[certificate\_whitelist] reload error for illigal lines"

exit 1

fi

}

check\_duplicate\_key()

{

config\_file=\$1;

dup\_key=\$(grep -v '^[ ]\*[;]' \$config\_file |grep "cal\."|awk -F"=" '{print \$1}'|awk '{print \$1}' |sort |uniq -d)

if [[ "" != \$dup\_key ]]; then

LOG\_ERROR "[certificate\_whitelist] has duplicate keys:"

LOG\_ERROR "\$dup\_key"

exit 1

fi

}

check\_whitelist()

{

config\_file=\$1

check\_cal\_lines \$config\_file

check\_duplicate\_key \$config\_file

}

check\_whitelist \${SHELL\_FOLDER}/../config.ini

cd \${SHELL\_FOLDER}/../

NODE\_FOLDER=\$(pwd)

fisco\_bcos=\${NODE\_FOLDER}/../${bcos\_bin\_name}

node=\$(basename \${NODE\_FOLDER})

node\_pid=${ps\_cmd}

if [ ! -z \${node\_pid} ];then

echo "\${node} is trying to reset certificate whitelist. Check log for more information."

touch config.ini.reset\_certificate\_whitelist

exit 0

else

echo "\${node} is not running, use start.sh to start and enable whitelist directlly."

fi

EOF

}

genTransTest() ##說明：生成交易測試

{

local output=$1

local file="${output}/.transTest.sh"

generate\_script\_template "${file}"

cat << EOF > "${file}"

# This script only support for block number smaller than 65535 - 256

ip\_port=http://127.0.0.1:$(( port\_start[2] ))

trans\_num=1

target\_group=1

version=

if [ \$# -ge 1 ];then

trans\_num=\$1

fi

if [ \$# -ge 2 ];then

target\_group=\$2

fi

getNodeVersion()

{

result="\$(curl -X POST --data '{"jsonrpc":"2.0","method":"getClientVersion","params":[],"id":1}' \${ip\_port})"

version="\$(echo \${result} | cut -c250- | cut -d \" -f3)"

}

block\_limit()

{

result=\$(curl -s -X POST --data '{"jsonrpc":"2.0","method":"getBlockNumber","params":['\${target\_group}'],"id":83}' \${ip\_port})

if [ \$(echo \${result} | grep -i failed | wc -l) -gt 0 ] || [ -z \${result} ];then

echo "getBlockNumber error!"

exit 1

fi

blockNumber=\$(echo \${result}| cut -d \" -f 10)

printf "%04x" \$((\$blockNumber+0x100))

}

send\_a\_tx()

{

limit=\$(block\_limit)

random\_id="\$(date +%s)\$(printf "%09d" \${RANDOM})"

if [ \${#limit} -gt 4 ];then echo "blockLimit exceed 0xffff, this scripts is unavailable!"; exit 0;fi

if [ "\${version}" == "2.0.0-rc1" ];then

txBytes="f8f0a02ade583745343a8f9a70b40db996fbe69c63531832858\${random\_id}85174876e7ff8609184e729fff82\${limit}94d6f1a71052366dbae2f7ab2d5d5845e77965cf0d80b86448f85bce000000000000000000000000000000000000000000000000000000000000001bf5bd8a9e7ba8b936ea704292ff4aaa5797bf671fdc8526dcd159f23c1f5a05f44e9fa862834dc7cb4541558f2b4961dc39eaaf0af7f7395028658d0e01b86a371ca0e33891be86f781ebacdafd543b9f4f98243f7b52d52bac9efa24b89e257a354da07ff477eb0ba5c519293112f1704de86bd2938369fbf0db2dff3b4d9723b9a87d"

else

txBytes="f8eca003eb675ec791c2d19858c91d0046821c27d815e2e9c15\${random\_id}0a8402faf08082\${limit}948c17cf316c1063ab6c89df875e96c9f0f5b2f74480b8644ed3885e0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000a464953434f2042434f53000000000000000000000000000000000000000000000101801ba09edf7c0cb63645442aff11323916d51ec5440de979950747c0189f338afdcefda02f3473184513c6a3516e066ea98b7cfb55a79481c9db98e658dd016c37f03dcf"

fi

#echo \$txBytes

curl -s -X POST --data '{"jsonrpc":"2.0","method":"sendRawTransaction","params":['\${target\_group}', "'\$txBytes'"],"id":83}' \${ip\_port}

}

send\_many\_tx()

{

for j in \$(seq 1 \$1)

do

echo 'Send transaction: ' \$j

send\_a\_tx \${ip\_port}

done

}

getNodeVersion

echo "Use version:\${version}"

send\_many\_tx \${trans\_num}

EOF

}

generate\_server\_scripts() ##說明：生成服務的script

{

local output=$1

genTransTest "${output}"

generate\_script\_template "$output/start\_all.sh"

# echo "ip\_array=(\$(ifconfig | grep inet | grep -v inet6 | awk '{print \$2}'))" >> "$output/start\_all.sh"

# echo "if echo \${ip\_array[@]} | grep -w \"${ip}\" &>/dev/null; then echo \"start node\_${ip}\_${i}\" && bash \${SHELL\_FOLDER}/node\_${ip}\_${i}/start.sh; fi" >> "${output\_dir}/start\_all.sh"

cat << EOF >> "$output/start\_all.sh"

dirs=(\$(ls -l \${SHELL\_FOLDER} | awk '/^d/ {print \$NF}'))

for directory in \${dirs[\*]}

do

if [[ -f "\${SHELL\_FOLDER}/\${directory}/config.ini" && -f "\${SHELL\_FOLDER}/\${directory}/start.sh" ]];then

echo "try to start \${directory}"

bash \${SHELL\_FOLDER}/\${directory}/start.sh &

fi

done

wait

EOF

generate\_script\_template "$output/stop\_all.sh"

cat << EOF >> "$output/stop\_all.sh"

dirs=(\$(ls -l \${SHELL\_FOLDER} | awk '/^d/ {print \$NF}'))

for directory in \${dirs[\*]}

do

if [[ -d "\${SHELL\_FOLDER}/\${directory}" && -f "\${SHELL\_FOLDER}/\${directory}/stop.sh" ]];then

echo "try to stop \${directory}"

bash \${SHELL\_FOLDER}/\${directory}/stop.sh &

fi

done

wait

EOF

}

parse\_ip\_config() ##說明：解析IP配置

{

local config=$1

n=0

while read line;do

ip\_array[n]=$(echo ${line} | awk '{print $1}')

agency\_array[n]=$(echo ${line} | awk '{print $2}')

group\_array[n]=$(echo ${line} | awk '{print $3}')

if [ -z "${ip\_array[$n]}" -o -z "${agency\_array[$n]}" -o -z "${group\_array[$n]}" ];then

exit\_with\_clean "Please check ${config}, make sure there is no empty line!"

fi

((++n))

done < ${config}

}

download\_bin() ##說明：下載FISCO-BCOS可執行文件

{

if [ "${x86\_64\_arch}" != "true" ];then exit\_with\_clean "We only offer x86\_64 precompiled fisco-bcos binary, your OS architecture is not x86\_64. Please compile from source."; fi

bin\_path=${output\_dir}/${bcos\_bin\_name}

package\_name="fisco-bcos.tar.gz"

[ ! -z "${macOS}" ] && package\_name="fisco-bcos-macOS.tar.gz"

[ ! -z "$guomi\_mode" ] && package\_name="fisco-bcos-gm.tar.gz"

if [[ ! -z "$guomi\_mode" && ! -z ${macOS} ]];then

exit\_with\_clean "We don't provide binary of GuoMi on macOS. Please compile source code and use -e option to specific fisco-bcos binary path"

fi

Download\_Link="https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/v${compatibility\_version}/${package\_name}"

LOG\_INFO "Downloading fisco-bcos binary from ${Download\_Link} ..."

if [ $(curl -IL -o /dev/null -s -w %{http\_code} ${cdn\_link\_header}/v${compatibility\_version}/${package\_name}) == 200 ];then

curl -LO ${Download\_Link} --speed-time 20 --speed-limit 102400 -m ${download\_timeout} || {

LOG\_INFO "Download speed is too low, try ${cdn\_link\_header}/v${compatibility\_version}/${package\_name}"

curl -LO ${cdn\_link\_header}/v${compatibility\_version}/${package\_name}

}

else

curl -LO ${Download\_Link}

fi

tar -zxf ${package\_name} && mv fisco-bcos ${bin\_path} && rm ${package\_name}

chmod a+x ${bin\_path}

}

check\_bin() ##說明：檢查FISCO-BCOS可執行文件

{

echo "Checking fisco-bcos binary..."

bin\_version=$(${bin\_path} -v)

if [ -z "$(echo ${bin\_version} | grep 'FISCO-BCOS')" ];then

exit\_with\_clean "${bin\_path} is wrong. Please correct it and try again."

fi

if [[ ! -z ${guomi\_mode} && -z $(echo ${bin\_version} | grep 'gm') ]];then

exit\_with\_clean "${bin\_path} isn't gm version. Please correct it and try again."

fi

if [[ -z ${guomi\_mode} && ! -z $(echo ${bin\_version} | grep 'gm') ]];then

exit\_with\_clean "${bin\_path} isn't standard version. Please correct it and try again."

fi

echo "Binary check passed."

}

main() ##說明：main()總入口

{

output\_dir="$(pwd)/${output\_dir}"

[ -z $use\_ip\_param ] && help 'ERROR: Please set -l or -f option.'

if [ "${use\_ip\_param}" == "true" ];then

ip\_array=(${ip\_param//,/ })

elif [ "${use\_ip\_param}" == "false" ];then

if ! parse\_ip\_config $ip\_file ;then

exit\_with\_clean "Parse $ip\_file error!"

fi

else

help

fi

dir\_must\_not\_exists ${output\_dir}

mkdir -p "${output\_dir}"

if [ -z "${compatibility\_version}" ];then

compatibility\_version=$(curl -s https://api.github.com/repos/FISCO-BCOS/FISCO-BCOS/releases | grep "tag\_name" | grep "\"v2\.[0-9]\.[0-9]\"" | sort -u | tail -n 1 | cut -d \" -f 4 | sed "s/^[vV]//")

fi

# in case network is broken ##說明：當網絡故障時的處理

if [ -z "${compatibility\_version}" ];then

compatibility\_version="${default\_version}"

fi

# download fisco-bcos and check it ##說明：下載 fisco-bcos 並檢查

if [ -z ${docker\_mode} ];then

if [[ -z ${bin\_path} ]];then

download\_bin

else

check\_bin

fi

fi

if [ -z ${CertConfig} ] || [ ! -e ${CertConfig} ];then

# CertConfig="${output\_dir}/cert.cnf"

generate\_cert\_conf "cert.cnf"

else

cp ${CertConfig} .

fi

if [ "${use\_ip\_param}" == "true" ];then

for i in $(seq 0 ${#ip\_array[\*]});do

agency\_array[i]="agency"

group\_array[i]=1

done

fi

# prepare CA ##說明：CA證書的預處理

echo "=============================================================="

if [ ! -e "$ca\_file" ]; then

echo "Generating CA key..."

dir\_must\_not\_exists ${output\_dir}/chain

gen\_chain\_cert ${output\_dir}/chain >${logfile} 2>&1 || exit\_with\_clean "openssl error!"

mv ${output\_dir}/chain ${output\_dir}/cert

if [ "${use\_ip\_param}" == "false" ];then

for agency\_name in ${agency\_array[\*]};do

if [ ! -d ${output\_dir}/cert/${agency\_name} ];then

gen\_agency\_cert ${output\_dir}/cert ${output\_dir}/cert/${agency\_name} >${logfile} 2>&1

fi

done

else

gen\_agency\_cert ${output\_dir}/cert ${output\_dir}/cert/agency >${logfile} 2>&1

fi

ca\_file="${output\_dir}/cert/ca.key"

fi

if [ -n "$guomi\_mode" ]; then

check\_and\_install\_tassl

generate\_cert\_conf\_gm "gmcert.cnf"

echo "Generating Guomi CA key..."

dir\_must\_not\_exists ${output\_dir}/gmchain

gen\_chain\_cert\_gm ${output\_dir}/gmchain >${output\_dir}/build.log 2>&1 || exit\_with\_clean "openssl error!" #生成secp256k1算法的CA密钥

mv ${output\_dir}/gmchain ${output\_dir}/gmcert

gen\_agency\_cert\_gm ${output\_dir}/gmcert ${output\_dir}/gmcert/agency >${output\_dir}/build.log 2>&1

ca\_file="${output\_dir}/gmcert/ca.key"

fi

echo "=============================================================="

echo "Generating keys ..."

nodeid\_list=""

ip\_list=""

count=0

server\_count=0

groups=

groups\_count=

for line in ${ip\_array[\*]};do

ip=${line%:\*}

num=${line#\*:}

if [ -z $(echo $ip | grep -E "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$") ];then

LOG\_WARN "Please check IP address: ${ip}, if you use domain name please ignore this."

fi

[ "$num" == "$ip" ] || [ -z "${num}" ] && num=${node\_num}

echo "Processing IP:${ip} Total:${num} Agency:${agency\_array[${server\_count}]} Groups:${group\_array[server\_count]}"

[ -z "$(get\_value ${ip//./}\_count)" ] && set\_value ${ip//./}\_count 0

for ((i=0;i<num;++i));do

echo "Processing IP:${ip} ID:${i} node's key" >> ${logfile}

local node\_coount=$(get\_value ${ip//./}\_count)

node\_dir="${output\_dir}/${ip}/node${node\_coount}"

[ -d "${node\_dir}" ] && exit\_with\_clean "${node\_dir} exist! Please delete!"

while :

do

gen\_cert ${output\_dir}/cert/${agency\_array[${server\_count}]} ${node\_dir} "node" >${logfile} 2>&1

mkdir -p ${conf\_path}/

mv \*.\* ${conf\_path}/

#private key should not start with 00

cd ${output\_dir}

privateKey=$(openssl ec -in "${node\_dir}/${conf\_path}/node.key" -text 2> /dev/null| sed -n '3,5p' | sed 's/://g'| tr "\n" " "|sed 's/ //g')

len=${#privateKey}

head2=${privateKey:0:2}

if [ "64" != "${len}" ] || [ "00" == "$head2" ];then

rm -rf ${node\_dir}

continue;

fi

if [ -n "$guomi\_mode" ]; then

gen\_node\_cert\_gm ${output\_dir}/gmcert/agency ${node\_dir} >${output\_dir}/build.log 2>&1

mkdir -p ${gm\_conf\_path}/

mv ./\*.\* ${gm\_conf\_path}/

#private key should not start with 00

cd ${output\_dir}

privateKey=$($TASSL\_CMD ec -in "${node\_dir}/${gm\_conf\_path}/gmnode.key" -text 2> /dev/null| sed -n '3,5p' | sed 's/://g'| tr "\n" " "|sed 's/ //g')

len=${#privateKey}

head2=${privateKey:0:2}

if [ "64" != "${len}" ] || [ "00" == "$head2" ];then

rm -rf ${node\_dir}

continue;

fi

fi

break;

done

if [ -n "$guomi\_mode" ]; then

cat ${output\_dir}/gmcert/agency/gmagency.crt >> ${node\_dir}/${gm\_conf\_path}/gmnode.crt

cat ${output\_dir}/gmcert/gmca.crt >> ${node\_dir}/${gm\_conf\_path}/gmnode.crt

#move origin conf to gm conf

rm ${node\_dir}/${conf\_path}/node.nodeid

cp ${node\_dir}/${conf\_path} ${node\_dir}/${gm\_conf\_path}/origin\_cert -r

fi

if [ -n "$guomi\_mode" ]; then

nodeid="$(cat ${node\_dir}/${gm\_conf\_path}/gmnode.nodeid)"

else

nodeid="$(cat ${node\_dir}/${conf\_path}/node.nodeid)"

fi

if [ -n "$guomi\_mode" ]; then

#remove original cert files

rm ${node\_dir:?}/${conf\_path} -rf

mv ${node\_dir}/${gm\_conf\_path} ${node\_dir}/${conf\_path}

fi

if [ "${use\_ip\_param}" == "false" ];then

node\_groups=(${group\_array[server\_count]//,/ })

for j in ${node\_groups[@]};do

if [ -z "${groups\_count[${j}]}" ];then groups\_count[${j}]=0;fi

echo "groups\_count[${j}]=${groups\_count[${j}]}" >> ${logfile}

groups[${j}]=$"${groups[${j}]}node.${groups\_count[${j}]}=${nodeid}

"

((++groups\_count[j]))

done

else

nodeid\_list=$"${nodeid\_list}node.${count}=${nodeid}

"

fi

ip\_list=$"${ip\_list}node.${count}="${ip}:$(( $(get\_value ${ip//./}\_count) + port\_start[0] ))"

"

set\_value ${ip//./}\_count $(( $(get\_value ${ip//./}\_count) + 1 ))

((++count))

done

sdk\_path="${output\_dir}/${ip}/sdk"

if [ ! -d ${sdk\_path} ];then

gen\_cert ${output\_dir}/cert/${agency\_array[${server\_count}]} "${sdk\_path}" "sdk" >${logfile} 2>&1

# FIXME: delete the below unbelievable ugliest operation in future

cp sdk.crt node.crt

cp sdk.key node.key

# FIXME: delete the upside unbelievable ugliest operation in future

rm node.nodeid

cp ${output\_dir}/cert/ca.crt ${sdk\_path}/

cd ${output\_dir}

fi

((++server\_count))

done

# clean

for line in ${ip\_array[\*]};do

ip=${line%:\*}

set\_value ${ip//./}\_count 0

done

echo "=============================================================="

echo "Generating configurations..."

cd ${current\_dir}

server\_count=0

for line in ${ip\_array[\*]};do

ip=${line%:\*}

num=${line#\*:}

[ "$num" == "$ip" ] || [ -z "${num}" ] && num=${node\_num}

echo "Processing IP:${ip} Total:${num} Agency:${agency\_array[${server\_count}]} Groups:${group\_array[server\_count]}"

for ((i=0;i<num;++i));do

echo "Processing IP:${ip} ID:${i} config files..." >> ${logfile}

local node\_coount=$(get\_value ${ip//./}\_count)

node\_dir="${output\_dir}/${ip}/node${node\_coount}"

generate\_config\_ini "${node\_dir}/config.ini" ${ip} "${group\_array[server\_count]}"

if [ "${use\_ip\_param}" == "false" ];then

node\_groups=(${group\_array[${server\_count}]//,/ })

for j in ${node\_groups[@]};do

generate\_group\_genesis "$node\_dir/${conf\_path}/group.${j}.genesis" "${j}" "${groups[${j}]}"

generate\_group\_ini "$node\_dir/${conf\_path}/group.${j}.ini"

done

else

generate\_group\_genesis "$node\_dir/${conf\_path}/group.1.genesis" "1" "${nodeid\_list}"

generate\_group\_ini "$node\_dir/${conf\_path}/group.1.ini"

fi

generate\_node\_scripts "${node\_dir}"

set\_value ${ip//./}\_count $(( $(get\_value ${ip//./}\_count) + 1 ))

done

generate\_server\_scripts "${output\_dir}/${ip}"

if [ -z ${docker\_mode} ];then cp "$bin\_path" "${output\_dir}/${ip}/fisco-bcos"; fi

if [ -n "$make\_tar" ];then cd ${output\_dir} && tar zcf "${ip}.tar.gz" "${ip}" && cd ${current\_dir};fi

((++server\_count))

done

rm ${logfile}

if [ -f "${output\_dir}/${bcos\_bin\_name}" ];then rm ${output\_dir}/${bcos\_bin\_name};fi

if [ "${use\_ip\_param}" == "false" ];then

echo "=============================================================="

for l in $(seq 0 ${#groups\_count[@]});do

if [ ! -z "${groups\_count[${l}]}" ];then echo "Group:${l} has ${groups\_count[${l}]} nodes";fi

done

fi

}

##說明：執行以下之方法

check\_env ##說明：1、檢查總運行環境

parse\_params $@ ##說明：2、始初化系統參數

main ##說明：3、總執行入口方法

print\_result ##說明：4、輸出執行結果