



# 梧桐链基础版 节点 部署说明



苏州同济区块链研究院  
Suzhou Tongji Blockchain Research Institute

# 目录

- 一、 介绍.....1
- 二、 操作系统.....1
- 三、 网络规划.....2
- 四、 服务器时间同步.....2
- 五、 **Docker** 环境配置 .....3
- 六、 节点必备文件解析.....4
- 七、 生成证书.....6
- 八、 启动节点具体步骤.....8
- 九、 启动监控服务具体步骤.....9
- 十、 检测部署是否成功.....10
- 十一、 节点常见问题及解决方法 .....10



## 一、 介绍

梧桐链平台是主要针对企业、机构的区块链应用场景开发的联盟链区块链系统平台。

梧桐链基础版提供基本的区块链服务，适合低成本快速部署区块链应用。支持多种加密算法、多种共识算法；支持高性能自主智能合约引擎；提供对区块链系统的治理和运维支持，可对整个网络运行状态进行实时监控。

梧桐链的节点部署说明主要用于描述整个节点网络的软硬件组成、系统架构、以及其安装部署方法、配置方法等信息。通过本文档可以对整体系统进行全新部署，或者针对某个部分进行重新部署。

## 二、 操作系统

### 1. 最低配置

配置选项	配置要求
操作系统	Linux 操作系统内核 3.1 以上，64 位，推荐使用 ubuntu 16.04.3
内存	$\geq 2\text{G}$
CPU	$\geq 2\text{C}$
硬盘	$\geq 100\text{G}$
网络	百兆及以上

注：本文所示操作均以 ubuntu 16.04 为例

### 2. 推荐配置

配置选项	配置要求
操作系统	Linux 操作系统内核 3.1 以上，64 位，推荐使用 ubuntu 16.04.3
内存	$\geq 8\text{G}$
CPU	$\geq 4\text{C}$
硬盘	$\geq 1\text{TB}$
网络	百兆及以上

注：本文所示操作均以 ubuntu 16.04 为例



### 三、 网络规划

服务器	端口规划	默认端口	说明
梧桐链节点	P2P 通讯	TCP 60000	节点间通讯
	gRPC 服务监听	TCP 9000	SDK 与节点通讯
	服务器健康监控	TCP 9090	健康监控服务程序
SDK 服务器	HTTP 服务	TCP 8080	http 服务
CA 服务	gRPC 服务监听	TCP 9092	节点与 CA 服务通讯

梧桐链系统网络架构示意图如下所示：

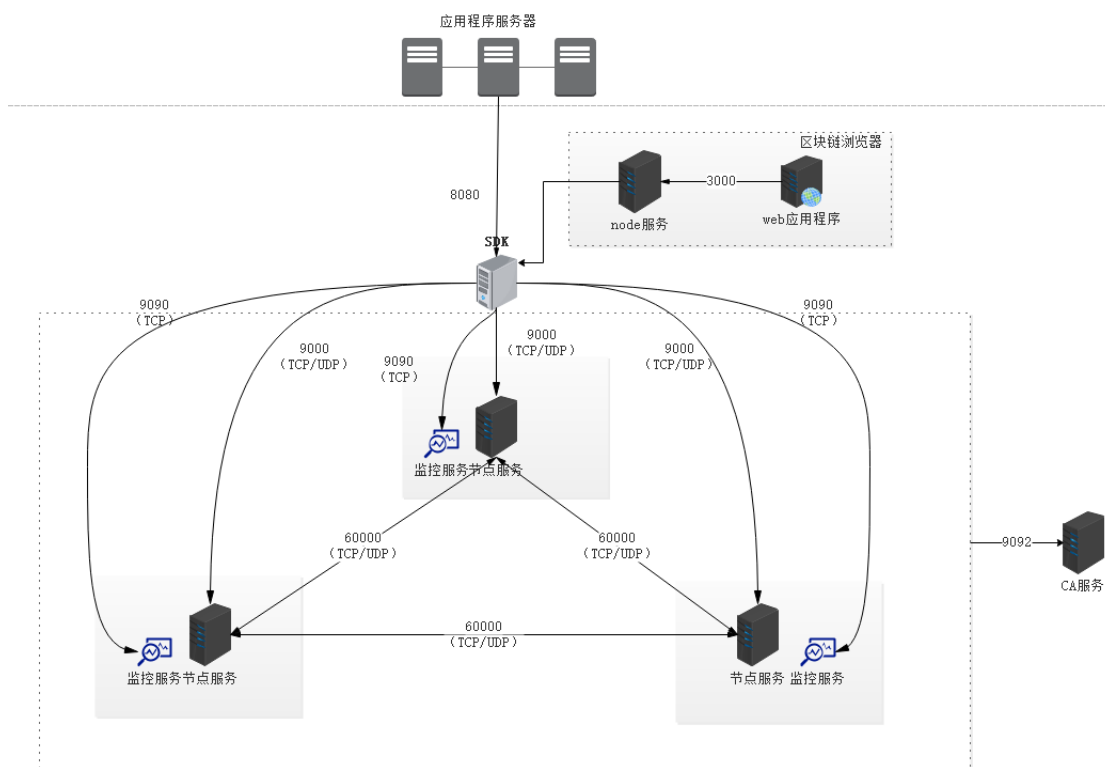


图 1 梧桐链及应用系统节点示意图

### 四、 服务器时间同步

选取局域网中的一台机器 A 作为时间服务器，其余服务器（节点，SDK）同步该台机器的时间。防火墙须开启 123/UDP 端口



1. 安装 ntpd, ntpdate  
Ubuntu:  
apt install ntp  
apt install ntpdate
2. 开启 123 端口  
vi /etc/services  
//新增两行  
ntp 123/tcp  
ntp 123/udp
3. 删除本地时间并设置时区为上海  
rm -rf /etc/localtime  
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
4. 时间服务器 A 开启 ntpd 服务  
service ntp start
5. 修改服务器 A 配置  
vim /etc/ntp.conf  
//加入以下配置  
server 127.127.1.0  
fudge 127.127.1.0 stratum 10
6. 重启 A 服务器 ntpd, 需等待五至六分钟  
service ntp restart
7. 其他服务器关闭 ntpd 服务, 并配置 ntpdate  
service ntp stop  
//x.x.x.x 为服务器 A 的 ip 地址  
ntpdate -u x.x.x.x  

8. 设置定时任务  
crontab -e  
//每天 23 点同步服务器时间  
0 23 \* \* \* /usr/sbin/ntpdate -u x.x.x.x

## 五、 Docker 环境配置

1. 解压 docker 安装包  
tar zxvf docker-18.03.0-ce.tgz
2. 进入 docker 解压目录



```
cd docker
```

3. 将 docker 目录下文件拷贝到/usr/local/bin/目录下，该目录一般都位于 PATH 下

```
cp * /usr/local/bin
```

4. 后台启动 docker

```
dockerd &
```

5. 验证 docker 是否安装成功

```
docker -v
```

6. 进入含有 chaincode.tar 的目录

```
cd ..
```

```
ls
```

7. 执行镜像安装

```
docker load < chaincode.tar
```

8. 确认是否安装镜像成功

```
docker images
```

成功则显示如下

```
root@ubuntu:~/zxx# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tjfoc/tjfoc-ccenv   1.0.1              9f54f0a46960       4 months ago       750MB
```

## 六、 节点必备文件解析

节点文件目录：

```
peer
├── conf
│   ├── base.yaml
│   └── config.yaml
├── crypt
│   ├── ca.pem
│   ├── cert.pem
│   └── key.pem
└── peer
```

1. 配置文件 base.yaml （目录./peer/conf/base.yaml）

Log:

Level: "Info" #默认日志输出等级，可选值: Error, Warning, Info, Debug

LogDir: "logs" #日志的输出目录

LogFile: "log.log" #日志文件名

RaftLevel: "Info" #Raft 模块的日志级别



DockerLevel: "Error" #Docker 模块的日志级别

#### Rpc:

Port: 9000 #rpc 服务的端口号

Timeout: 50 #超时时间, 单位为毫秒

TLS-enabled: false #是否使用 TLS 证书

TLSCaPath: "./tls/ca.pem" #TLS 安全传输层 CA 根证书路径

TLSKeyPath: "./tls/key.pem" #TLS 安全传输层私钥路径

TLSCertPath: "./tls/cert.pem" #TLS 安全传输层证书路径

TLSServerName: "test.example.com" #TLS 安全传输层服务器名称

#### Node:

NodeCertPath: "./cert/nodeCert.pem" #节点认证证书

#### StorePath:

Path: "peer.db" #存储记录交易过程, 所有上链数据

#### Crypt:

KeyType: "sm2" #加密算法, 支持 sm2, ecc

HashType: "sm3" #hash 算法, 支持 sm3, sha256

KeyPath: "./crypt/key.pem" #非对称加密的私钥

#### Docker: #docker 通信端口及内核版本

Ip: "0.0.0.0" #docker 的 IP

Port: "7051" #docker 的端口

Endpoint: unix:///var/run/docker.sock #docker 的相关默认配置

BaseImageVersion: "1.0.1" #镜像的版本

Enabled: false #是否使用 docker

#### CA:

OnlineCAEnabled: true #在线 CA 认证是否启用

CAAddress: "your ip:9092" #CA 的 IP 地址

## 2. 配置文件 config.yaml (目录./peer/config.yaml)

#### Self: #本机节点地址和端口

Id: "158" #为自身在集群中的编号, 不可重复

ShownName: "peer158" #节点的显示名称

Addr: "ip:60000" #节点内网 IP 和端口

#### Members: #整个集群需要加入节点的所有主机的地址。

Peers:



- Id: "157" # 节点在集群中的编号  
ShownName: "peer157" # 节点的显示名称  
InAddr: "ip: 60000" # 入站 IP+ 端口  
OutAddr: "ip: 60000" # 进出站 IP 一致时本字段可以省略  
- Id: "158" # 节点在集群中的编号  
ShownName: "peer158" # 节点的显示名称  
InAddr: "ip: 60000" # 入站 IP+ 端口  
OutAddr: "ip: 60000" # 进出站 IP 一致时本字段可以省略

### 3. 端口

只要可以使用并且没有被占用的端口号都可以任意配置。

## 七、 生成证书

梧桐链目前提供证书生成工具生成证书(目前只支持 sm2 加密, 如需使用 ecc 加密, 请使用 openssl 等工具生成)

### 1. Linux 环境

- 1) 将下载的证书生成工具移动到 Linux 服务器上
- 2) 赋予权限

chmod +x certtool

```
root@ubuntu:~/certtool# ls
certtool
root@ubuntu:~/certtool# chmod +x certtool
root@ubuntu:~/certtool# ls
certtool
```

### 3) 生成证书

命令:

./certtool -d 36500 -n 3 -c test.example.com

参数说明:

./certtool -d [有效期] -n [密钥对数量] -c [域名]

-d 是代表证书有效期, 默认值 36500

-n 是生成不包括根证书 ca 的证书及公私钥数量, 默认值 3

-c 证书的域名, 默认值 test.example.com

注意:

每次生成的证书随机, ca 根证书不同, 需按需求生成相对应的密钥对, 否则生成的普通密钥对的根证书不对应。

如下图生成根证书和根证书密钥对, 以及其余三对证书和密钥对。





```
root@ubuntu:~/certtool# ./certtool -d 36500 -n 3 -c test.example.com
Version:
=====SM2=====
create ca cert!
create node cert!
*****success!*****
=====SM2=====
=====SM2=====
create node cert!
*****success!*****
=====SM2=====
=====SM2=====
create node cert!
*****success!*****
=====SM2=====
root@ubuntu:~/certtool# ls
ca cert1 cert2 cert3 certtool
```

如下图显示每个文件夹下都包含一个证书 `cert.pem` 文件（根证书为 `ca.pem`），一个私钥 `key.pem` 文件，一个公钥 `pubkey.pem` 文件。

```
root@ubuntu:~/certtool# ls ca
ca.pem key.pem pubkey.pem
root@ubuntu:~/certtool# ls cert1
cert.pem key.pem pubkey.pem
```

## 2. Windows 环境

- 1) 打开 windows 系统的命令提示符
- 2) 进入证书生成工具 `certtool.exe` 的文件夹下

```
C:\Users\Administrator\Desktop\certtool>ls
certtool.exe
```

- 3) 生成证书

命令：

`certtool.exe -d [有效期] -n [密钥对数量] -c [域名]`

参数说明：

`certtool.exe -d 36500 -n 3 -c test.com`

-d 是代表证书有效期，默认值 36500

-n 是生成不包括根证书 `ca` 的证书及公私钥数量，默认值 3

-c 证书的域名，默认值 `test.example.com`

注意：









每次生成的证书随机，`ca` 根证书不同，需按需求生成相对应的密钥对，否则生成的普通密钥对的根证书不对应。

如下图生成根证书和根证书密钥对，以及其余三对证书和密钥对。



```
C:\Users\Administrator\Desktop\certtool>certtool.exe -d 36500 -n 3 -c test.com
Version:
=====SM2=====
create ca cert!
create node cert!
*****success!*****
=====SM2=====
=====SM2=====
create node cert!
*****success!*****
=====SM2=====
=====SM2=====
create node cert!
*****success!*****
=====SM2=====
```

如下图显示每个文件夹下都包含一个证书 **cert.pem**（根证书为 **ca.pem**），一个私钥 **key.pem**，一个公钥 **pubkey.pem**。

 ca	2018/11/16 14:21	文件夹	
 cert1	2018/11/16 14:21	文件夹	
 cert2	2018/11/16 14:21	文件夹	
 cert3	2018/11/16 14:21	文件夹	
 certtool.exe	2018/11/16 14:16	应用程序	3,611 KB
 ca.pem	2018/11/16 14:21	PEM 文件	1 KB
 key.pem	2018/11/16 14:21	PEM 文件	1 KB
 pubkey.pem	2018/11/16 14:21	PEM 文件	1 KB
 cert.pem	2018/11/16 14:21	PEM 文件	1 KB
 key.pem	2018/11/16 14:21	PEM 文件	1 KB
 pubkey.pem	2018/11/16 14:21	PEM 文件	1 KB

## 八、 启动节点具体步骤

### 1. 证书生成

节点必须配备节点验证证书、签名私钥。节点 TLS 验证证书可选。

#### 1) 节点验证证书（cert 文件夹，目录：./peer/cert/）

- A. 按照步骤[七、生成证书](#)生成多套证书密钥对，每个节点都使用同一套节点验证的根证书生成的密钥对。
- B. 将生成的证书移动到 cert 文件夹中。  
cert 文件夹必备 nodeCert.pem（证书）。



- 2) 节点签名私钥 (cert 文件夹, 目录: ./peer/crypt)
  - A. 按照步骤[七、生成证书](#)生成多套证书密钥对, 每个节点都使用同一套节点签名的根证书。
  - B. 将生成的文件移动到 crypt 文件夹中。  
crypt 文件夹必备 key.pem (私钥)。
- 3) TLS 验证证书 (crypt 文件夹, 目录: ./peer/crypt)
  - C. 按照步骤[七、生成证书](#)生成多套证书密钥对, 每个节点都使用同一套 TLS 验证的根证书。
  - D. 将生成的文件移动到 crypt 文件夹中。  
crypt 文件夹必备 ca.pem (CA 证书), cert.pem (证书), key.pem (私钥)。

**注意:**

证书文件名应当与节点配置文件中节点证书配置一致, 可以修改文件名, 也可以修改配置文件。

修改文件名:

```
mv [当前文件名] [配置文件中的文件名]
```

```
mv cert.pem nodeCert.pem
```

2. 更改配置文件 (配置文件解析具体[六、节点必备文件解析](#)。更改日志等级、是否启用证书、节点 id 和地址、加密方式等)

```
cd conf
```

```
vim base.yaml
```

3. 启动

- 1) ./peer

启动节点, 从 DB 里读取节点列表, 如果没有 DB, 从配置文件读取。

- 2) ./peer start -c

只从配置文件中读取节点列表

- 3) ./peer start -r

只从网络中读取节点列表

## 九、启动监控服务具体步骤

1. 文件目录

```
./peer/appserver
```

2. 后台启动命令

```
nohup ./ appserver &
```

3. 注意



## 十、检测部署是否成功

```

root@ubuntu:/home/user/chaincode# ./peer_v20180424
#####112e30
2018-04-26 13:07:35.041 CST [chaincode_support] HandleChaincodeStream -> INFO 001 ===== entry HandleChaincodeStream
2018-04-26 13:07:35.041 CST [chaincode_support] HandleChaincodeStream -> INFO 002 Current context deadline = 0001-01-01 00:00:00 +0000 UTC, ok = false
2018-04-26 13:07:35.042 CST [chaincode_support] beforeRegisterEvent -> INFO 003 Unmarshal chaincodeID: name=" "
2018-04-26 13:07:35.045 CST [chaincode_support] launchAndWaitForRegister -> WARN 004 xxx in case , ok:true
2018-04-26 13:07:35.109 CST [raft] NewRaft -> INFO 005 localId: c4ca4238a0b923820dc6509a6f7f
2018-04-26 13:07:35.109 CST [raft] NewRaft -> INFO 006 peerId: cfc020849d565f66e7df9f987
2018-04-26 13:07:35.109 CST [raft] NewRaft -> INFO 007 peerId: c81e728d9d4c2f636f867f89c1d
Wutong Chain Peer
Copyright 2018 Suzhou Tongji Fintech Research Institute
http://www.tj-fintech.com/wtl/
version: 0.0.1

```

## 十一、节点常见问题及解决方法

报错信息:

```
2018-04-26 01:52:43.078 PDT [worldstate] requireCompare -> ERRO 00b wd:Send RequireMes error:dial tcp :60000: getsockopt: connection refused
```

解决方法:

需开启集群其他节点。其他节点未开启,该节点显示连接不上其他节点,至少需要两个节点算法才能进行投票。

2. WARN: connection refused, type5

报错信息:

```
2018-04-26 02:09:16.076 PDT [raft] doAppendLogRequest -> DEBU 037 enter peer:p_196
2018-04-26 02:09:16.079 PDT [raft] sendPeerMessage -> WARN 038 send peer:p_196, error:dial tcp :20000: getsockopt: connection refused, type:5
```

解决方法:

该节点配置文件错误，节点连接不上 IP:20000 这个节点，说明集群中无该节点记录，需要修改配置文件正确配置节点集群中的节点。

3. WARN: connection refused, type4

报错信息:

```
2018-04-26 02:16:48.377 PDT [raft] sendPeerMessage -> WARN 0f9 send peer_ip: error:dial tcp 60000: getsockopt: connection refused, type:4
2018-04-26 02:16:48.578 PDT [raft] sendPeerMessage -> WARN 0fa send peer_ip: error:dial tcp 60000: getsockopt: connection refused, type:4
2018-04-26 02:16:48.780 PDT [raft] sendPeerMessage -> WARN 0fb send peer_ip: error:dial tcp 60000: getsockopt: connection refused, type:4
```

解决方法:

节点连接不上 IP:60000 这个节点，说明该节点已断开或未开启，请重新开启该节点。



#### 4. vote self term

报错信息:

```
2018-04-26 02:05:21.591 PDT [raft] runCandidate -> INFO 010 vote self term : 1
2018-04-26 02:05:21.593 PDT [raft] sendPeerMessage -> WARN 011 send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:21.594 PDT [raft] sendPeerMessage -> WARN 012 send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:22.592 PDT [raft] run -> INFO 013 new state: 1
2018-04-26 02:05:22.593 PDT [raft] runCandidate -> INFO 014 vote self term : 1
2018-04-26 02:05:22.595 PDT [raft] sendPeerMessage -> WARN 015 send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:22.596 PDT [raft] sendPeerMessage -> WARN 016 send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:23.595 PDT [raft] run -> INFO 017 new state: 1
2018-04-26 02:05:23.596 PDT [raft] runCandidate -> INFO 018 vote self term : 1
2018-04-26 02:05:23.598 PDT [raft] sendPeerMessage -> WARN 019 send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:23.599 PDT [raft] sendPeerMessage -> WARN 01a send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:24.597 PDT [raft] run -> INFO 01b new state: 1
2018-04-26 02:05:24.599 PDT [raft] runCandidate -> INFO 01c vote self term : 1
2018-04-26 02:05:24.601 PDT [raft] sendPeerMessage -> WARN 01d send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:24.601 PDT [raft] sendPeerMessage -> WARN 01e send peer error:dial tcp 60000: getsockopt: connection refused, type:1
2018-04-26 02:05:25.602 PDT [raft] run -> INFO 01f new state: 1
2018-04-26 02:05:25.602 PDT [raft] runCandidate -> INFO 020 vote self term : 1
```

解决方法:

其他节点断开, 该节点接受不到其他节点请求, 节点算法只能投票给自己, 须开启其他节点。