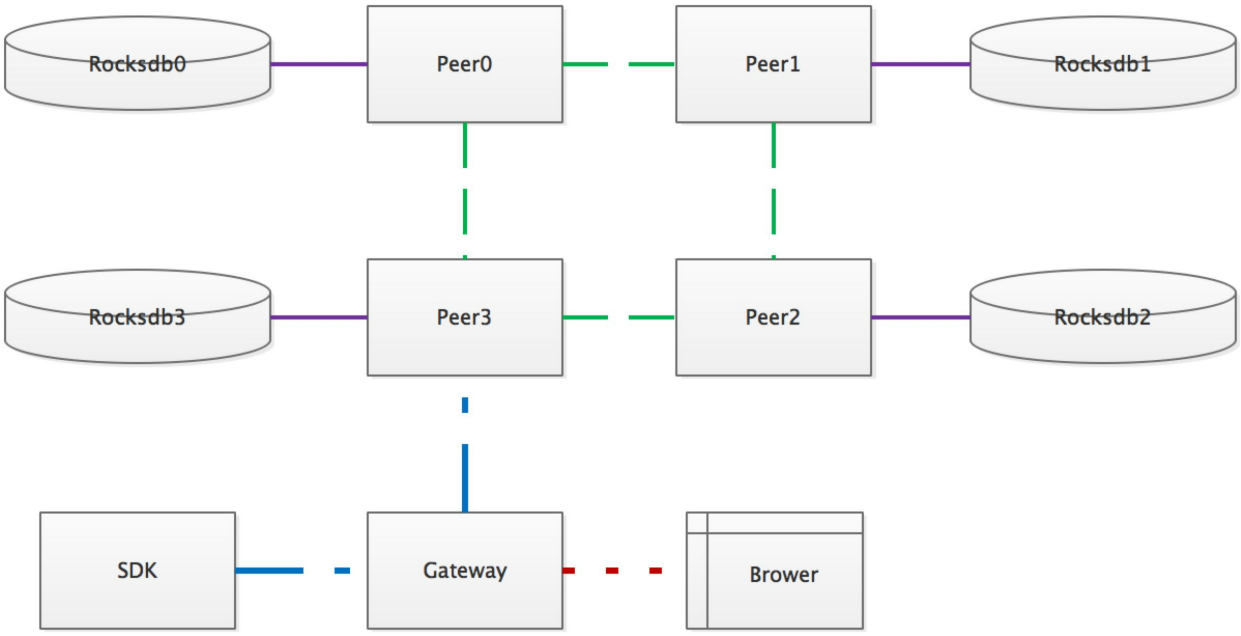
JD Chain安装部署指南

本部署指南基于JD Chain1.0.1.RELEASE版本来构建。

1. 部署环境

1.1 系统部署结构



JD Chain有三类节点：Peer、网关和客户端。peer节点是区块链主节点，参与共识、账本操作等；网关节点（GateWay）与Peer节点通信，负责区块链浏览器及消息传递；客户端与网关通信，可以进行写入或查询账本等操作。

JD Chain默认采用Rocksdb作为存储数据库，每个Peer节点与一个（或多个）Rocksdb实例连接，Rocksdb实例的数量与账本相关，不同Peer节点对应不同的Rocksdb实例。

JD Chain也支持Redis作为其存储数据库，Redis数据库不必和Peer节点安装在同一服务器，只需要配置正确即可（后面配置会说到）。

JD Chain默认共识机制采用的是BFTSmart，该共识算法要求最低四个共识节点，BFTSmart 对作恶节点的支持为：N = 3f+1，即在N个节点共识中可允许f个作恶节点，当节点数增加时可根据该公式进行配置。

本安装部署指南以四个节点为例进行说明。

1.2 服务器配置

1.2.1 硬件配置

为了更好的运行，JD Chain推荐使用 Linux 操作系统，Peer节点自身运行对内存要求较低，但对于Rocksdb存储而言，其对运行环境有较高要求，以下为单Peer节点可正常运行 *亿级交易* 的参考指标：

- 内核2.7.x及以上；
- CPU主频 ≥ 2.0GHz；
- 内存 ≥ 64G；
- 可用硬盘容量 >= 200G

1.2.2 软件配置

JD Chain使用Java开发，其软件需求如下：

- JDK：Version ≥ 1.8。

> 注意：若数据库使用的Redis，则需要对应数据库服务器安装Redis，Redis版本要求 >=3.x。

---

2. 系统安装与配置

2.1 安装包结构

JD Chain默认发布的安装包有两个，一个是Peer节点打包程序，一个是Gateway节点打包程序，两者默认都是用tar.gz（或zip）打包的。

2.1.1 Peer节点安装包

Peer节点打包程序为：jdchain-peer-$version.tar.gz或jdchain-peer-$version.zip，若是tar.gz压缩文件可通过tar命令解压（如下）：

```
tar -xzvf jdchain-peer-$version.tar.gz
```

若是zip文件可通过unzip命令解压（如下）：

```
unzip jdchain-peer-$version.zip
```

> 注意：$version为JD Chain发布的版本号

Peer打包程序解压完后的安装包结构如下：

- bin
  - keygen.sh
  - ledger-init.sh
  - startup.sh
  - shutdown.sh
- config
  - init
    - ledger.init
    - local.conf
    - bftsmart.config（默认）
  - keys
    - %.priv
    - %.pub
    - %.pwd
  - ledger-binding.conf
- docs
- libs
- system

其中目录说明如下：

- bin： 相关命令操作目录；
- config： 对应命令的配置目录，keys路径解压时不存在，会在执行keygen.sh脚本时自动创建；ledger-binding.conf文件解压时不存在，会在成功执行ledger-init.sh脚本后生成；
- docs： 相关文档保存目录；
- libs： 项目运行依赖第三方及非system依赖包保存路径；
- system： 项目运行系统包保存路径；

> 1）注意： *bin* 目录下所有的文件都需要可执行权限，在Linux环境下可简单参考如下命令：

```
cd bin
chmod 777 *
```

> 2）注意：system文件夹包括运行时系统主要加载的jar文件，为系统安全运行定制，开发者无须关心此项。

2.1.2 Gateway节点安装包

Gateway节点打包程序为：jdchain-gateway-$version.tar.gz或jdchain-gateway-$version.zip，其解压完后的安装包结构如下：

- bin
  - startup.sh
  - shutdown.sh
- config
  - gateway.conf
- lib

  其中目录说明如下：
  - bin ： 相关命令操作目录；
  - config ： 对应命令的配置目录；
  - lib ： gateway运行时所需jar包路径；

2.2 注册用户

JD Chain以一个 *密钥对* 标识一个用户，任何区块链操作都必须有用户的签名信息，因此首先需要创建一个用户。
JD Chain提供了创建用户的工具，可直接通过命令行生成一个新用户。切换到bin路径，执行命令：

```
./keygen.sh -n jd-com
```

其中jd-com是生成的密钥对文件前缀，需自定义。
执行命令过程中需要输入密钥对中 *私钥* 加密的密码，并选择是否将该密码编码保存（推荐选择保存，该密码编码会保存至%.pwd文件中）。

> 注意：请妥善保存该密码，切勿丢失！

执行命令后，该 *密钥对* 文件会生成到 *config/keys* 路径，如下所示：

```
-rw-r--r-- 1 root root 71 _25 18:18 jd-com.priv
-rw-r--r-- 1 root root 52 _25 18:26 jd-com.pub
-rw-r--r-- 1 root root 44 _25 18:18 jd-com.pwd
```

其中，jd-com.priv是签名私钥的密文形式内容（以口令哈希作为对称密钥，对签名私钥进行对称加密后的密文数据），jd-com.pub为签名公钥内容，jd-com.pwd是用于加密签名私钥的口令哈希值。

> 注意：%.pwd记录的编码在后续Gateway配置时会用到！

请将所有Peer节点都按照上述流程注册一个独立的用户，在实际生产环境中该过程可能由不同的参与方完成。

2.3 账本初始化

账本初始化是所有参与Peer节点进行共识，并生成初始化账本至数据库的过程。该过程需要所有节点共同参与，同时启动。

2.3.1 初始化配置

config/init 目录下有三个配置文件需要修改：local.conf 、 ledger.init 和 bftsmart.config 。

- local.conf描述账本初始化的本地（即当前节点）配置；
- ledger.init描述账本初始化过程中涉及到的其他参与Peer节点配置信息;
- bftsmart.config为BFTSmart进行共识的相关配置。

2.3.1.1 local.conf配置

```
#当前参与方的 id
local.parti.id=0

#当前参与方的公钥
local.parti.pubkey=endPsK36koyFr1D245Sa9j83vt6pZUdFBJoJRB3xAsWM6cwhRbna

#当前参与方的私钥（密文编码）
local.parti.privkey=177gjsj5PHeCpbAtJE7qnbmhuZMHAEKuMsd45zHkv8F8AWBvTBbff8yRKdCyT3k

#当前参与方的私钥解密密钥(原始口令的一次哈希，Base58格式)，如果不设置，则启动过程中需要从控制台输入
local.parti.pwd=DYu3G8aGTMBW1WrTw76zxQJQU4DHLw9MLyy7peG4LKkY

#账本初始化完成后生成的"账本绑定配置文件"的输出目录
ledger.binding.out=../config

#账本数据库的连接字符，下为rocksdb样例
#rocksdb数据库连接格式：rocksdb://{path}
```

```
#redis数据库连接格式：redis://{ip}:{prot}/{db}
ledger.db.uri=rocksdb:///export/app/peer/rocks.db/rocksdb0.db

#账本数据库的连接口令
ledger.db.pwd=
```

其中local.parti.id从 0 开始编号，且不同Peer节点不能相同；local.parti.privkey为当前节点私钥，即 *config/keys/%.priv* 文件中的内容；其他参数根据实际环境进行配置。

2.3.1.2 ledger.init配置

```
#账本的种子；一段16进制字符，最长可以包含64个字符；可以用字符“-”分隔，以便更容易读取；
ledger.seed=932dfe23-fe23232f-283f32fa-dd32aa76-8322ca2f-56236cda-7136b322-cb323ffe

#账本的描述名称；此属性不参与共识，仅仅在当前参与方的本地节点用于描述用途；
#ledger.name=

#声明的账本创建时间；格式为 “yyyy-MM-dd HH:mm:ss.SSSZ”，表示”年-月-日 时:分:秒:毫秒时区“；例
created-time=2019-10-17 05:21:58.069+0800

#共识服务提供者；必须；
consensus.service-provider=com.jd.blockchain.consensus.bftsmart.BftsmartConsensusPr

#共识服务的参数配置；必须；
#consensus.conf=/export/app/peer/config/init/bftsmart.config
consensus.conf=bftsmart.config

#密码服务提供者列表，以英文逗点“,”分隔；必须；
crypto.service-providers=com.jd.blockchain.crypto.service.classic.ClassicCryptoServ
com.jd.blockchain.crypto.service.sm.SMCryptoService

#参与方的个数，后续以 cons_parti.id 分别标识每一个参与方的配置；
cons_parti.count=4


#第0个参与方的名称；
cons_parti.0.name=
#第0个参与方的公钥文件路径；
cons_parti.0.pubkey-path=
#第0个参与方的公钥内容（由keygen工具生成）；此参数优先于 pubkey-path 参数；
cons_parti.0.pubkey=
#第0个参与方的账本初始服务的主机；
cons_parti.0.initializer.host=127.0.0.1
#第0个参与方的账本初始服务的端口；
cons_parti.0.initializer.port=8800
#第0个参与方的账本初始服务是否开启安全连接；
cons_parti.0.initializer.secure=false

#第1个参与方的名称；
cons_parti.1.name=
```

```
#第1个参与方的公钥文件路径；
cons_parti.1.pubkey-path=
#第1个参与方的公钥内容（由keygen工具生成）；此参数优先于 pubkey-path 参数；
cons_parti.1.pubkey=
#第1个参与方的账本初始服务的主机；
cons_parti.1.initializer.host=127.0.0.1
#第1个参与方的账本初始服务的端口；
cons_parti.1.initializer.port=8810
#第1个参与方的账本初始服务是否开启安全连接；
cons_parti.1.initializer.secure=false
```

账本初始化过程中，需要其他Peer节点的公钥信息进行验证及保存，因此每个节点都需要获取其他Peer节点的公钥信息至本地。配置中cons_parti.N.*中的N需按照实际每个参与方定义的序号配置（该配置为local.conf中的local.parti.id）。其他参数根据实际环境进行配置。

> 注意：上述config中只显示了两个参与方（Peer节点）的配置信息，其他参与方（默认是4个节点）的配置方式一致！

2.3.1.1 bftsmart.config配置

```
system.server.0.network.host=127.0.0.1
system.server.0.network.port=16000

system.server.1.network.host=127.0.0.1
system.server.1.network.port=16010

system.servers.num = 4

system.servers.f = 1

system.initial.view = 0,1,2,3
```

> 注意：上述只是将bftsmart.config文件中主要需要修改的参数显示，以方便进行相关说明，其他参数的修改可参考具体文件中的具体说明

参数具体说明如下：

- system.server.$n.network.host：第n个节点的域名；
- system.server.$n.network.port：第n个节点的共识端口；
- system.servers.num：共识节点总数；
- system.servers.f：共识节点最大支持的作恶节点数；
- system.initial.view：为每个共识节点分配的viewID，每个viewID之间使用英文逗号分隔，viewID不允许重复，且viewID总数需要和system.servers.num一致。

2.3.2 初始化部署

按照 *2.3.1章节* 配置完成后，只需要启动ledger-init.sh即可，可参考如下命令：

```
cd bin
./ledger-init.sh
```

执行命令之后，会在 *config* 目录下生成ledger-binding.conf文件，该文件即账本初始化生成的文件，Peer节点启动时需要依赖该文件。

> 1）注意：因为JD Chain支持多账本形式，若config/ledger-binding.conf文件在初始化之前就存在的话，初始化操作后不会覆盖其中的内容，会以追加的方式写入。若第一次创建账本，建议先将该文件删除再进行初始化！

> 2）注意：Peer节点会定时检测ledger-binding.conf，有新账本加入时会自动进行更新，不需要重启Peer节点！目前默认时间为封5自动更新，即：每个小时的5/15/25/35/45/55分钟会执行；

账本初始化成功后，每个Peer节点对应的Rocksdb都会写入该账本相关的数据。

2.4 Peer节点安装

Peer节点启动依赖于 *config* 目录下ledger-binding.conf的配置，该文件由 *2.3章节* 操作生成，无须做任何修改；application.properties文件中主要用于配置Peer节点对外HTTP端口。

> 注意：application.properties文件可能不存在，可手动创建。该文件中配置适用于SpringBoot2.x版本的相关参数，不限于目前内置的启动端口。

由于Peer节点启动后会自动与其他参与节点进行通信，因此需要同时启动4个Peer节点，只需要执行startup.sh即可，参考命令：

```
cd bin
./startup.sh
```

> 1）注意：startup.sh命令中可修改启动端口，默认为：-p 7080；

> 2）注意：Peer节点会与账本中涉及到的参与方进行通信，当通信不成功（例如有节点尚未启动）时，会自动进行重试，因此多个Peer节点启动可不必完全同时进行。目前默认设置为重试16次操作，每次间隔时间2秒。

2.5 Gateway节点安装

GateWay（网关）节点可以认为是一个过滤节点，交易的提交及账本的查询都需要通过网关节点与Peer节点进行通信。 *Gateway* 程序可独立部署，不需要依赖Peer节点，它的操作环境是 *2.1.2 章节* 中解压后的环境，网关节点启动依赖于配置文件 *config/gateway.conf*。

```
#网关的HTTP服务地址;
http.host=0.0.0.0
#网关的HTTP服务端口;
http.port=8081
#网关的HTTP服务上下文路径，可选;
#http.context-path=

#共识节点的服务地址;
peer.host=127.0.0.1
#共识节点的服务端口;
peer.port=7080
#共识节点的服务是否启用安全证书;
peer.secure=false
#共识节点的服务提供解析器
peer.providers=com.jd.blockchain.consensus.bftsmart.BftsmartConsensusProvider

#数据检索服务对应URL
#若该值不配置或配置不正确，则浏览器模糊查询部分无法正常显示
data.retrieval.url=http://192.168.151.39:10001

#默认公钥的内容（Base58编码数据）;
keys.default.pubkey=
#默认私钥的路径；在 pk-path 和 pk 之间必须设置其一;
keys.default.privkey-path=
#默认私钥的内容（加密的Base58编码数据）；在 pk-path 和 pk 之间必须设置其一;
keys.default.privkey=
#默认私钥的解码密码;
keys.default.privkey-password=
```

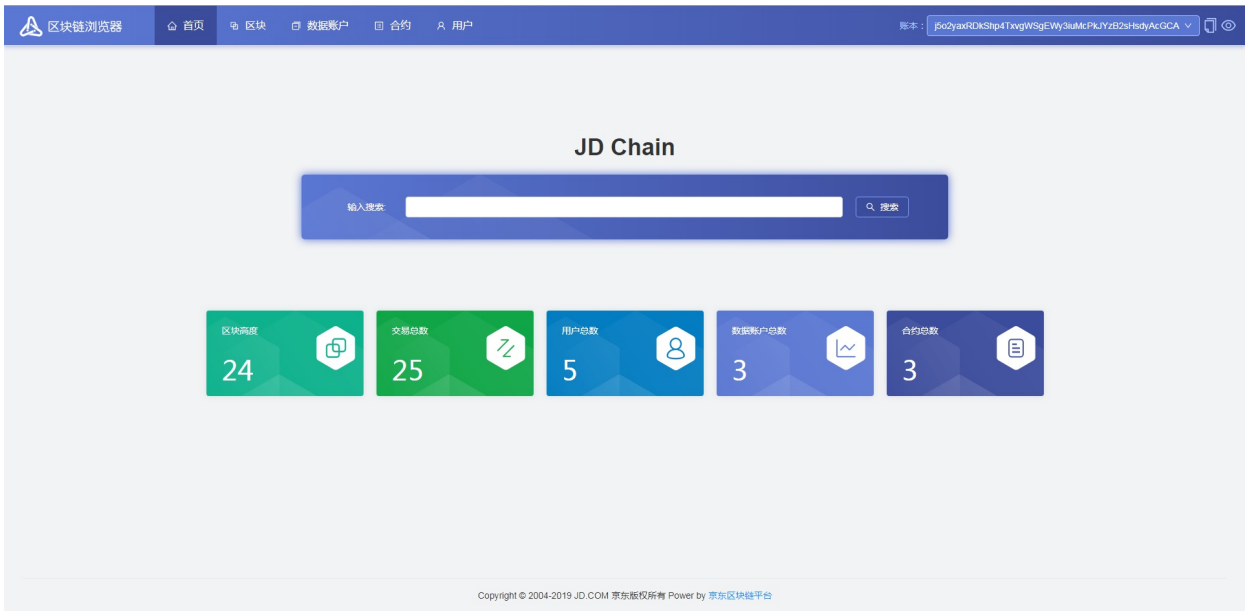
其中keys.default.% 配置是网关的角色配置，目前暂不支持网关自定义角色，因此，网关可选择4个参与方的任何一个作为其配置（主要是密钥对）。

> 注意：keys.default.privkey-password填写内容为*2.2章节*中生成的%.pwd文件中的内容！

启动网关节点只需要执行：startup.sh即可，参考命令：

```
cd bin
./startup.sh
```

网关节点启动成功后，可通过访问区块链浏览器获取其账本相关信息：http://[ip]:[port]
下图为区块链浏览器首页，可供参考：



> 注意：JD Chain网关节点默认自带区块链浏览器！！！