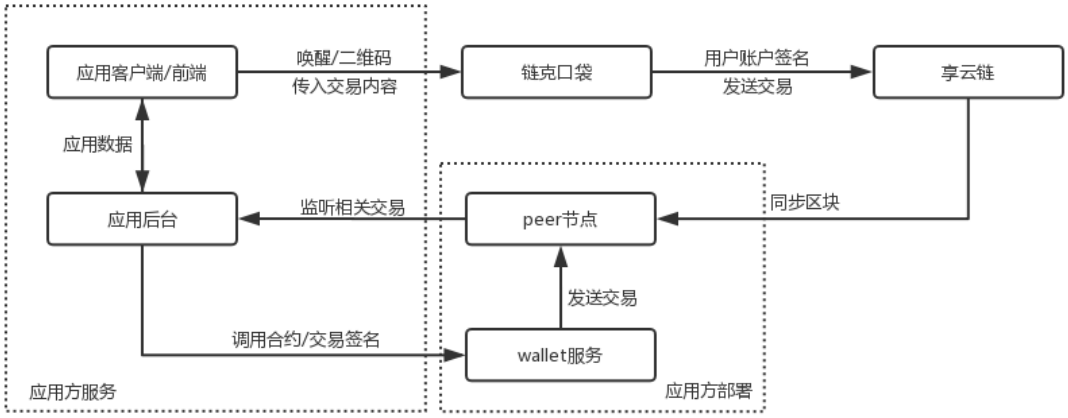
享云链接入快速入门

通过以下步骤，用户可以快速学习和接入享云链，并发送区块链交易到所启动的节点上，从而对享云链的整体使用有一个大致的理解，并且能够结合自己的应用使用享云链。更多的概念解释和操作步骤可以参考[操作指南]和[技术参考]文档。

教程文档主要分为以下四个部分：启动享云链节点、发送享云链交易（普通交易与合约交易）、同步享云链区块、应用与链克口袋交互。用户可以根据自己的需求参考相关的步骤。

一般基于享云链的应用部署结构如图



开发者可遵循以下顺序实现应用开发和享云链接入：

1. git clone linchain 项目
2. 初始化和启动测试环境节点连接
3. 启动wallet服务
4. 开发合约应用或转账应用，部署合约
5. 应用端后台使用wallet服务发送后台合约交易或后台转账交易

6. 应用端前端或客户端使用用户参数按照链克口袋协议拼接交易内容，生成二维码或者唤醒链克口袋的链接
7. 用户使用链克口袋扫码或者应用端唤醒链克口袋，用户支付链克执行交易
8. 应用后台监听peer节点交易，更新数据

测试环境测试通过后，可启动正式环境节点与wallet服务接入，部署正式环境合约。

启动享云链节点

关于启动享云链节点的步骤，以 linkchain github 文档中的启动步骤为准。
下面以启动本地单节点测试网络为例：

克隆项目

```
$ git clone https://github.com/lianxiangcloud/linkchain.git
```

构建镜像

```
$ cd linkchain
$ sudo docker image build -t lkbuilder .
```

构建镜像过程中，会默认执行一次项目编译

构建成功后，可以查看到镜像信息

```
$ sudo docker images
REPOSITORY                          TAG                 IMAGE ID
CREATED              SIZE
lkbuilder                           latest              3e70915811c4        About
a minute ago   2.24GB
```

启动容器

```
$ sudo docker run -ti lkbuilder
```

拉取最新代码

```
$ git pull
```

执行编译打包

```
$ ./build.sh
```

编译成功后在/src/pack/lkchain/bin/目录能看到编译后的文件：

```
$ ll /src/pack/lkchain/bin/
```

```
total 89668
-rwxr-xr-x 1 root root 49804544 Aug 27 02:49 lkchain
```

查询后版本号，以实际git提交最新版本为准

```
$ /src/pack/lkchain/bin/lkchain version
```

```
linkchain version: 0.1.0, gitCommit:7f5d2a3e
```

运行单节点测试网络

进入docker容器内

`$ sudo docker run -ti lkbuilder`

初始化

`$ sh /src/scripts/sandbox_start.sh init peer ~/blockdata/`

```
committee contract code nil!!!
validators white list contract code nil!!!
genesisBlock stateHash
0x0d8827403cb36d8d176cbf6257915f1b5274ba11ff2891b06a0263946ebf0b57
genesisBlock trieRoot
0x0000000000000000000000000000000000000000000000000000000000000000
genesisBlock ChainID:chainID
block.Hash:0x26cb0291c88674df8614a93eb0e1b5e23b82e3117f18dade10acb0cf7c597b2d
```

启动节点:

`$ sh /src/scripts/sandbox_start.sh peer ~/blockdata/`

```
start lkchain ...
pid: 390
```

测试RPC:

`$ curl -H 'Content-Type: application/json' -d '{"jsonrpc":"2.0","id":"0","method":"eth_blockNumber","params":[]}' http://127.0.0.1:41000`

```
{"jsonrpc":"2.0","id":"0","result":"0x0"}
```

查看Log:

`$ tail ~/blockdata/peer_logs/lkchain.log`

```
DEBUG 2019-08-27 03:04:44.797 status report
 module=mempool specGoodTxs=0 goodTxs=0 futureTxs=0
DEBUG 2019-08-27 03:04:44.819 Broadcasting proposal heartbeat message
 module=consensus height=3 round=0 sequence=1
DEBUG 2019-08-27 03:04:46.820 Broadcasting proposal heartbeat message
 module=consensus height=3 round=0 sequence=2
DEBUG 2019-08-27 03:04:48.821 Broadcasting proposal heartbeat message
 module=consensus height=3 round=0 sequence=3
DEBUG 2019-08-27 03:04:49.797 status report
 module=mempool specGoodTxs=0 goodTxs=0 futureTxs=0
DEBUG 2019-08-27 03:04:49.865 dialOutLoop
 module=conManager maxDialOutNums=3 needDynDials=3
DEBUG 2019-08-27 03:04:49.865 ReadRandomNodes
 module=httpTable tab.seeds=[]
DEBUG 2019-08-27 03:04:49.865 after dialRandNodesFromCache
module=conManager needDynDials=3
DEBUG 2019-08-27 03:04:49.865 dialNodesFromNetLoop
module=conManager needDynDials=3
DEBUG 2019-08-27 03:04:50.822 Broadcasting proposal heartbeat message
 module=consensus height=3 round=0 sequence=4
```

启动wallet服务

进入钱包启动脚本目录
`$ cd /src/wallet/sbin`

钱包默认连接本地的peer节点，如果上一步已经启动了一个本地的测试peer，那么现在可以直接启动钱包连接这个peer

启动钱包进程
`$ ./wallet.sh start`

新建账户，密码为12345678（实际使用中不能泄露账户密码和私钥文件）

```
$ curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","method":"personal_newAccount","params":["12345678","this is
my test account"],"id":67}' -H 'Content-Type:application/json'

{"jsonrpc":"2.0","id":67,"result":"0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784"}
```

解锁钱包，测试账户的密码是"12345678"

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","method":"personal_unlockAccount","params":
["0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784","12345678",3600],"id":67}' -H
'Content-Type:application/json'
{"jsonrpc":"2.0","id":67,"result":true}
```

通过节点服务发送区块链交易

通过上面的步骤或自己通过其他方式启动了享云链节点以后，就可以向区块链发送交易来实现自己的区块链应用了。这里的交易泛指通过享云链做链克转账的交易，和通过部署或执行享云链智能合约来读写数据的交易。

构造交易

所有的交易结构都遵循以下结构体：

```
tx: {
    from,
    to,
    value,
    gas,
    data,
    nonce
}
```

对于链上链克转账的交易，解释为from账户发起交易，向to账户转入value数量的链克，data为空，另外享云链为普通交易扩展了一个remark字段，用于一些额外信息的记录。
对于链上执行合约的交易，解释为由from账户发起合约执行，执行合约账户to的某个方法data（方法加调用参数的编码），如果所执行的方法接收链克，可以通过value转入链克。
gas 为执行交易限制消耗的最大手续费数量。
nonce 为账户在链上执行的第几笔交易。
gas的估算和nonce值都可以通过wallet服务的接口获取。

下面构造一个普通的转账交易，合约的交易在操作指南中做详细介绍。

普通交易（从内置账户转账给新建账户）

```
{
    from: '0xa73810e519e1075010678d706533486d8ecc8000',
    to: '0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784',
    value: '0xde0b6b3a7640000'
}
```

使用from账户（内置账户） 0xa73810e519e1075010678d706533486d8ecc8000 向 to账户（新建的测试账户） 0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784 转入 0xde0b6b3a7640000(转十进制 10^18 wei = 1链克)

使用账户对交易做签名

每一笔交易发送到链上，都需要经过发送者私钥签名，其他节点在接收到签名的交易时可以验证交易的合法性。下面是用 wallet 服务提供的接口为上述两笔交易做签名。(用户在实际操作时需要将from账户替换成自己生成的账户地址)

1. 获取预估手续费数量，执行 `ltk_estimateGas` 接口

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_estimateGas","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","to":"0x3c4b41b3b769932ce2
a48746a7e9dcd9f7d5c784","value":"0xde0b6b3a7640000"}]}' -H 'Content-Type:
application/json'

{"jsonrpc":"2.0","id":"0","result":"0x7a120"}
```

2. 获取账户nonce值，执行 `ltk_getTransactionByHash` 接口

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionCount","params":
["0xa73810e519e1075010678d706533486d8ecc8000","latest"]}' -H 'Content-Type:
application/json'

{"jsonrpc":"2.0","id":"0","result":"0x0"}
```

3. 解锁账户，调用 `personal_unlockAccount` 接口

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","method":"personal_unlockAccount","params":
["0xa73810e519e1075010678d706533486d8ecc8000","1234",3600],"id":67}' -H
'Content-Type:application/json'

{"jsonrpc":"2.0","id":67,"result":true}
```

4. 使用私钥对交易做签名，调用 `ltk_signTransaction` 接口

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_signTransaction","params":
[{"from":"0xa73810e519e1075010678d706533486d8ecc8000","to":"0x3c4b41b3b769932ce2
a48746a7e9dcd9f7d5c784","value":"0xde0b6b3a7640000","nonce":"0x0","gas":"0x7a120
","gasPrice":"0x174876e800"}]}' -H 'Content-Type: application/json'

{"jsonrpc":"2.0","id":"0","result":
{"raw":"0xf86f8085174876e8008307a120943c4b41b3b769932ce2a48746a7e9dcd9f7d5c78488
0de0b6b3a76400008082e3e6a07fa9e0fea5022012ff8490abc53686b2638caf764d5a44b8ae1aec
901db4e64ea05a01a0cbf195a8abd063176277e10b5914c61a1b54a22cbc27764e787aedbc50","t
x":
{"nonce":"0x0","gasPrice":"0x174876e800","gas":"0x7a120","to":"0x3c4b41b3b769932
ce2a48746a7e9dcd9f7d5c784","value":"0xde0b6b3a7640000","input":"0x","v":"0xe3e6"
,"r":"0x7fa9e0fea5022012ff8490abc53686b2638caf764d5a44b8ae1aec901db4e64e","s":"0
x5a01a0cbf195a8abd063176277e10b5914c61a1b54a22cbc27764e787aedbc50","hash":"0x0cb
dfbad48ca93343d458df74bc63a9743e81844199c3fb49c45524d91fbcede"}}}
```

以上就是构造一个交易和使用账户对交易做签名的过程，最终得到的结构体里的raw就是最终发送到链上所需的数据。

发送交易

调用 `ltk_sendRawTransaction` 接口将上一步签出的raw发送交易到链上

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_sendRawTransaction","params":
["0xf86f8085174876e8008307a120943c4b41b3b769932ce2a48746a7e9dcd9f7d5c784880de0b6
b3a76400008082e3e6a07fa9e0fea5022012ff8490abc53686b2638caf764d5a44b8ae1aec901db4
e64ea05a01a0cbf195a8abd063176277e10b5914c61a1b54a22cbc27764e787aedbc50"]}' -H
'Content-Type: application/json'

{"jsonrpc":"2.0","id":"0","result":"0x0cbdfbad48ca93343d458df74bc63a9743e8184419
9c3fb49c45524d91fbcede"}
```

交易发送到链上确认的过程是异步的，发送后会生成一个交易hash，最终的确认结果需要根据hash到链上查询上链状态才能确认是否执行成功。

验证结果

上链结果可以通过多种方式验证，比如`ltk_getTransactionReceipt`/`ltk_getTransactionByHash`等接口，下面使用`ltk_getTransactionReceipt`示例：

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getTransactionReceipt","params":
["0x0cbdfbad48ca93343d458df74bc63a9743e81844199c3fb49c45524d91fbcede"]}' -H
'Content-Type: application/json'

{"jsonrpc":"2.0","id":"0","result":
{"blockHash":"0x5e2f552028904ca89b1e82c55374b950ecb08f6bb32b78ca537f382e9a7657da
","blockNumber":"0x2","contractAddress":null,"cumulativeGasUsed":"0x7a120","from
":"0xa73810e519e1075010678d706533486d8ecc8000","gasUsed":"0x7a120","logs":
[],"logsBloom":"0x00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000","status":"0x1","to":"0x3c4b4
1b3b769932ce2a48746a7e9dcd9f7d5c784","tokenAddress":"0x0000000000000000000000000
000000000000000","transactionHash":"0x0cbdfbad48ca93343d458df74bc63a9743e8184419
9c3fb49c45524d91fbcede","transactionIndex":"0x0"}}
```

根据返回结果status=0x1，可知执行成功。
也可以查询一下转入账户的余额变化：

```
curl -s -X POST http://127.0.0.1:18082 -d
'{"jsonrpc":"2.0","id":"0","method":"ltk_getAccountInfo","params":
["0x0000000000000000000000000000000000000000",
"0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784"]}' -H 'Content-Type:
application/json'

{"jsonrpc":"2.0","id":"0","result":{"eth_account":
{"address":"0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784","balance":"0xde0b6b3a764
0000","nonce":"0x0"},"utxo_accounts":
[{"address":"bzTSG8ki1WhzCueTaEu1gUSonEjDotsBg5fPfSbAoKZfGoeGbnhvvs6a1j7uADEmLGZ
8d7Biey22ksXzdi1g3biiEnBgPU","index":"0x0","balance":"0x0"}],"total_balance":"0x
de0b6b3a7640000","token":"0x0000000000000000000000000000000000000000"}}
```

节点控制台操作

上面的步骤是通过wallet提供的服务发送的交易，这种方式利于程序化服务；另外一些一次性的操作也可以通过节点控制台来实现。

连接控制台

测试节点服务启动后，可以通过rpc连接节点Geth控制台服务，默认为端口11000。

```
$ cd /src/pack/lkchain/bin
$ ./lkchain  attach http://127.0.0.1:11000

Welcome to the Geth JavaScript console!

 modules: debug:1.0 eth:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0

>
```

通过控制台发送交易

1. 查询账户余额

```
> eth.getBalance("0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784")
1000000000000000000
```

2. 解锁账户

```
> personal.unlockAccount("0xa73810e519e1075010678d706533486d8ecc8000")
Unlock account 0xa73810e519e1075010678d706533486d8ecc8000
Passphrase:
true
```

3. 发送交易

```
> eth.sendTransaction({from: "0xa73810e519e1075010678d706533486d8ecc8000", to:
"0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784", value: web3.toWei(1)})
"0x52c8a3e69238135c61087f0c9cdf25e9d04ba7483849840a7ee2d7a1c270732c"
```

4. 查询账户余额

```
> eth.getBalance("0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784")
2000000000000000000
```

通过链克口袋发送转账交易

链克口袋是享云链提供的用户端账户与资产管理的客户端软件，用户可以通过链克口袋App保管自己的账户、发起转账、以及扫码或外部应用唤醒执行合约。
当前使用链克口袋执行交易的方式分为以下几种：

1. 使用交易内容直接生成二维码方式

直接使用交易内容结构拼接字符串生成二维码，对于简单的交易来说（如备注转账），拼接后的字符串长度小于160个字符，生成的二维码对于大多数移动设备来说可以快速识别成功。如果交易内容过长，拼接后的字符串长度大于160个字符，会导致生成的二维码难以识别，这种情况建议使用第二种二维码活码方式。

2. 应用端唤醒方式

应用端可以通过构造唤醒链克口袋的链接，把交易内容传入到链克口袋中来执行交易签名。

下面将继续使用前面构造的交易结构，并且新增一个remark字段用于标记应用方的交易订单信息。

```
{
    to: '0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784',
    value: '0xde0b6b3a7640000',
    remark: '2c09cadc2f1040009be98bbfe48062e3'
}
```

from账户是通过链克口袋执行时选中的账户。

直接生成二维码转账

二维码字符串拼接规则：
`base64(ptitlubancommon://transfer?to=tx.to&value=number(tx.value)&remark=encodeURL(tx.remark))`

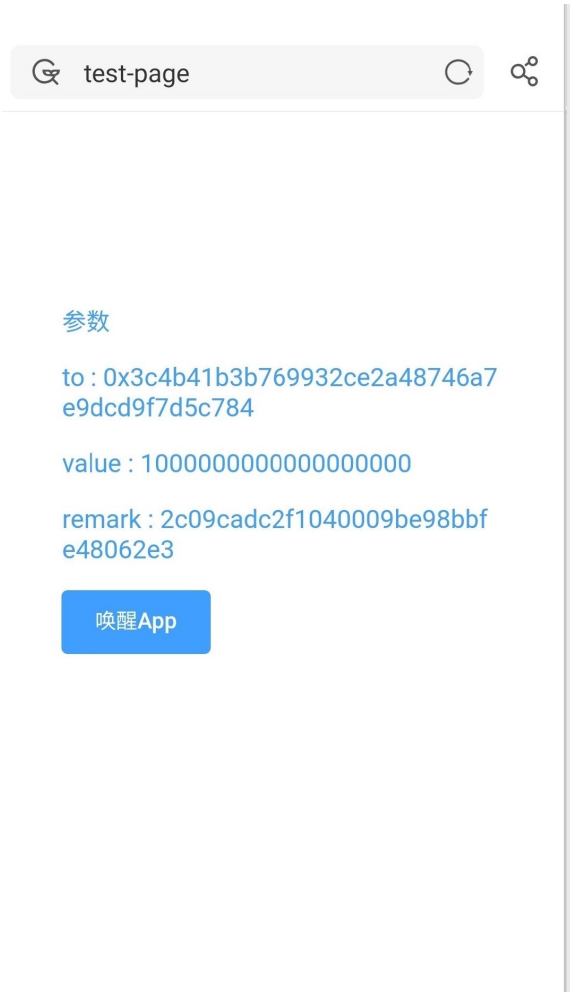
可以得到以下的

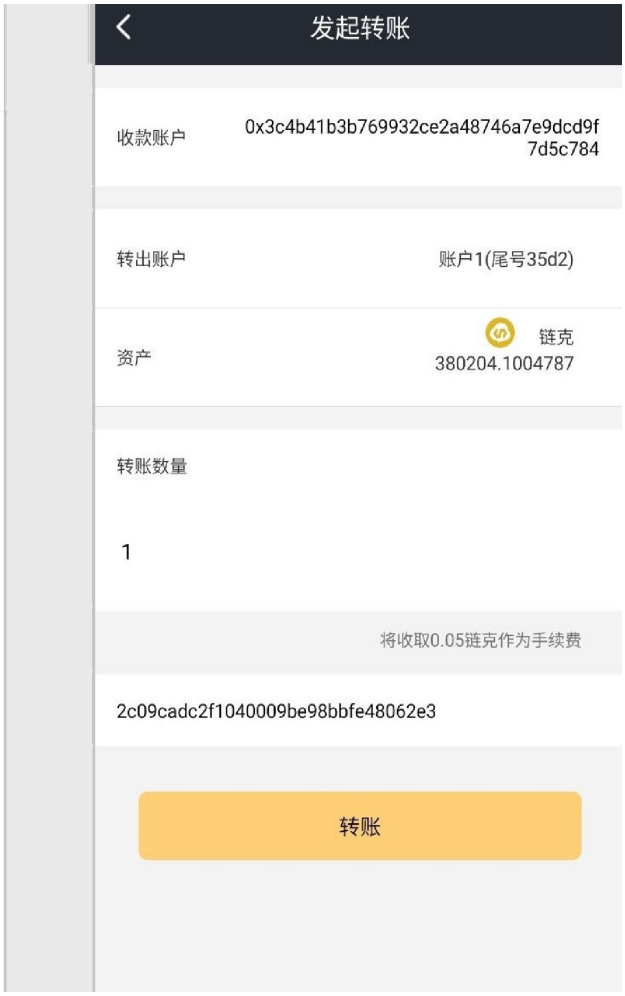
```
base64(ptitlubancommon://transfer?
to=0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784&value=1000000000000000000&remark=2
c09cadc2f1040009be98bbfe48062e3)

cHRpdGx1YmFuY29tbW9uOi8vdHJhbnNmZXI/dG89MHgzYzRiNDFiM2I3Njk5MzJjZTJhNDg3NDZhN2U5
ZGNkOWY3ZDVjNzg0JnZhbHVlPTEwMDAwMDAwMDAwMDAwMDAwMDAmcmVtYXJrPTJjMDljYWRjMmYxMDQw
MDA5YmU5OGJiZmU0ODA2MmUz
```

二维码如图

应用端唤醒方式

通过第三方应用或第三方页面构造请求，请求链接为
`ptitlubancommon://transfer?to=0x3c4b41b3b769932ce2a48746a7e9dcd9f7d5c784&value=1000000000000000000&remark=2c09cadc2f1040009be98bbfe48062e3`





点击后将会唤醒链克口袋。