使用chaincode可能是相关行业人士要注意的知识。在这里，我们将详细介绍使用柴火取暖时需要启动的地方，并拓展一些相关知识与大家分享，希望能给你带来帮助！

首先，你的程序没有&#039；这似乎不符合你的意思。如果chainLinkcode是二进制或者十六进制，左右移动还是有意义的，但问题是十进制。如果我是你

链节代码=链节代码/1000；链接代码=链链接代码00；

如果是为着节省空间，定义为无符号short，那应该没必要，其实全是4字节的int，处理器运行速度更快。

构建您第一个网络提供了一个结构网络示例。示例网络由两个组织组成，每个组织维护两个对等节点，并演示了基于链码查询两个帐户余额和转账的操作。

第一网络中有一个启动脚本byfn.sh，它使用内置的Docker镜像来快速启动网络。这个脚本将启动一个订购者节点和属于两个不同组织的四个对等节点。，它还将启动一个cli运行脚本，该脚本将向通道添加对等节点，部署并实例化链代码，并根据部署的链代码驱动事务执行。以下是byfn.sh

的帮助文档

执行up-oetcdraft启动脚本，指定使用raft共识算法

通过cryptogen工具生成组织成员、身份证书、密钥等文件。。调用configtxgen工具生成节点和通道配置文件，包括订购方节点上系统通道的genesis块文件genesis.block。、新申请通道配置事务文件channel.tx、组织锚节点配置更新事务文件Org1MSPanchors.tx和Org2MSPanchors.tx等。

用户执行network_setup.sh脚本来启动结构网络，这将调用networkUp函数。该命令将检查网络实体的证书是否生成，如果没有，第一个教师将创建相关的证书目录。

生成证书主要执行了密码生成配置=https://yrb114。com/crypto-config。YAML

根据crypto-config.yaml，生成网络成员的组织结构和身份证书、签名私钥等对应文件并保存在默认的crypto-config目录下，身份证书等文件在对应的目录msp/下。，TLS证书和密钥文件保存在tls/中。

Crypto-
config.YAML主要包括fabric排序节点的证书配置和fabric组织的证书配置。

模板定义节点的配置模式

规范另一种配置模式

统计节点总数

主机名完全限定的域名命名格式

域名

用户添加到管理员的用户帐户数量

Specs模式配置五个排序节点

模板模式配置两个组织，每个组织有两套公私钥和证书，普通用户数为1

。

OrdererOrganizations目录：包含ID证书。pem文件、签名私钥文件_sk文件和TLS证书(certificate。crt文件和密钥。关键文件)，订购方组织类型为

。

对等组织目录：包含身份证书。pem文件，签名私钥文件_sk文件，TLS证书(证书。crt文件和密钥。密钥文件)的对等组织类型

。(XY001)这些Dokumentewurdendocker-compose工具，基于docker-compose-cil.yaml，docker-compose-
base.yaml等待文件，使用目录作为将卷装入容器的指定目录。

调用replacePrivateKey，基于docker-compose-e2e-template.YAML文件创建一个新的配置文件docker-compose-e2e.yaml。。进入Org1的CA目录，获取私钥的文件名为PRIV_KEY，替换docker-compose-
e2e.yaml文件的CA1_PRIVATE_KEY。同理，替换CA2_PRIVATE_KEY。

执行generateChannelArtifacts函数，使用configtxgen工具基于configtx.yaml创

建节点和通道配置文件，包括订购者通道的创建块。，使用通道配置来配置事务文件channel.tx，使用锚节点配置来更新事务文件Org1MSPanchors.tx和org2MSPanchors.tx

订购者系统通道

的创建块。

执行configtxgen命令创建订购者创建块文件genesis.block

新应用通道的配置事务文件

执行configtxgen命令。，创建应用通道的配置事务文件channel.tx，然后执行对等通道创建读取该文件。

锚节点配置更新事务文件

执行configtxgen命令。，创建锚节点配置更新事务文件Org1MSPanchors.tx和Org2MSPanchors.tx，然后执行对等通道更新进行更新。

返回network_setup.sh脚本，判断CouchDB标志是否启用(默认关闭)，用docker-Compose工具执行docker-Compose-cil.yaml文件。,starttoconstructthenetwork

andcontinuetoanalyzethenetwork_settings.shscript.Thedefaultwriting_fileisdocker-compose-cli.yamlfile.当设置了kafka共识或raft共识时，使用相应的yaml文件。

订购者节点继承docker-compose-base.yaml中的orderer.example.com配置属性，当订购者节点容器启动时，执行下面的命令

。

docker-compose-cil.yaml文件[XY002][XY001]继续追踪base/docker-compose-base.yaml，找到orderer.example.com服务，挂载目录并公开端口7050。

订购者配置仍然被引入到peer-base.yaml

中的订购者基础服务中。

四个对等节点继承docker-compose-base.yaml中对应容器名的配置属性，当对等节点容器启动时，执行以下命令

。

docker-compose-cil.yaml在文件对等节点配置。

docker-compose-base.YAML在文件节点对等配置[XY002](XY001)介绍同行基地服务

docker-compose-cil.YAML在文件关于客户端配置。orderer节点、peer节点和CLI容器启动后，实际调用的是script.sh脚本，在CLI容器中执行，CLI容器实际上是一个用户客户端，只是一个命令行客户端。，在容器中运行。默认情况下，CLI的身份是admin.org1，连接到peer0.org1节点，执行script.sh脚本

script.sh脚本，依次执行默认测试流程。，包括新建应用通道、添加节点、更新锚节点、安装链码、实例化链码、调用链码、查询链码等操作。

Fabric要求创建、加入和更新渠道的权限必须具有渠道组织的管理员状态。。调用setGlobals设置全局环境变量，CLI客户端可以灵活切换指定容器的管理员角色，可以直接连接操作指定的Peer节点，先切换到Peer0/Org1节点。

接下来以Org1的管理员身份执行peer指令，将通道配置文件Channel.tx发送到Orderer节点，创建mychananel的应用通道。如果创建成功，则返回一个创建块。，存储在对等节点的文件系统中，包含channel.tx指定的通道配置信息，

遍历所有节点，调用setGlobals切换指定节点，执行对等指令。，将Org1组织中包含的Peer0/Org1和Peer1/Org1节点添加到mychananel应用通道中，并将Genesis块的mychananel.block设置为命令行参数。Org2组织类似。

joinchannelwithrytry函数中的setGlobals是设置CLI容器的环境变量的函数。例如，setGlobals12将CLI的标识设置为admin.org22。，连接peer1.org2节点。

使用对等通道加入命令让节点加入通道，$CHANNEL_NAME.block是通道成功创建时返回的块。，当上面的org1.peer0创建通道时，该块保留在CLI容器中，因此可以直接使用。节点成功加入通道后，它将创建一个以CHANNEL_NAME.block

开始的链

一个组织只能有一个锚节点，该锚节点只有在节点加入通道后才能更新。连续两次调用updateAnchorPeers更新两个组织的锚节点配置，用Org1管理标识更新Peer0/Org1的配置。，并指定锚节点配置更新文件Org1MSPanchors.txOrg2的组织方式相同。

连续调用installChaincode两次，分别在Peer0/Org1和Peer2/Org2中安装chaincode_example02链码，并将链码命名为&quotmycc&quot对于1.0版。如果成功安装了链代码，则在指定的安装目录/var/hyperledger/production/chaincodes中存在名为.version的链代码文件。

链代码的实例化必须在安装链代码的节点上进行。同一渠道所有节点上相同的实例化数据在渠道账簿中共享，用户只需在任一对等节点上成功进行一次链码实例化。经过排序和打包后，实例化的数据将被广播到其他节点，通道中的所有合法节点都可以访问链码的实例化数据。

实例化将链码添加到通道中，启动目标节点的容器，初始化与链码相关的初始值。这里的初始值是[&quot；a&quot，&quot100&quot，&quotb&quot，&quot200&quot].&quot实例化&quot会产生一个链代码的容器，比如dev-peer0-org1.example.com-mycc-1.0。。实例化过程需要指定背书策略，定义为AND(&#039；Org1msp。同行&#039；'Org2msp。同行&#039；)通过设置-P参数。，表示任何交易必须有org1和org2节点的共同背书。

在对等体0/Org1上调用chaincodeQuery函数chaincodeQuery和链代码调用函数chaincodequery。。查看A的余额，从账户A转10元到账户B.

在Peer3/Org2上安装链码，查询A的余额，查看余额是否为90元。如果是，则转移成功。

generatefiletemplate

$cryptogenshowtemplatecrypto-config.yaml

byusingcryptotoolsprovidedbytheconstruction.

修改并添加组织和订购者节点。

根据crypto-config.yaml文件生成证书文件：

$cryptogengenerateconfig=crypto-config。YAML

查看生成的证书文件夹结构：

您需要从fabric

的源代码案例中复制configtx.yaml文件

CPGOPATH/src/github。com/hyperledger/fabric-samples/first-network/configtx。YAML

修改configtx.yaml文件。

在修改之前，创建一个文件夹来保存要创建的创建块文件

。

将创建块文件和通道的命令写入脚本！Generate.sh

脚本文件和配置文件的目录结构：

执行generate.sh文件以生成创建块文件和通道。实际上只有一个组织，不需要生成锚节点更新文件.

$

配置docker-撰写文件：

启动容器，检查启动后容器的运行情况

$docker-composeup-d

$docker-composePS

。

这里创建两个脚本文件来管理docker容器

clear_docker.sh文件：

restart.sh文件：

。

创建配置文件时，有两个文件可供参考.

修改的sdk配置文件：

创建一个模型对象，给它赋值，开始初始化sdk

。

使用pkg/fabsdk/fabsdk.go中的New()方法进行实例化

在创建请求之前，您需要使用gopackager。NewCCPackage方法来生成资源。CCPackage对象，传递两个参数，一个是链代码的路径(相对于项目的路径)，一个是GOPATH的路径。

使用pkg/client/resmgmt/resmgmt.go文件

中的方法安装链代码。

在创建请求之前，您需要生成一个类型为*cb的对象。SignaturePolicyEnvelope，只需使用文件third_party/github.com/hyperledger/fabric/common/causedsl/causedsl_builder.go中的方法即可，这里提供了几种方法，你可以使用任何一种。这里，我们使用SignedByAnyMember方法。

实例化链代码

使用pkg/client/channel/chclient.go中的Execute()方法写入数据：

rsp，错误：=模型渠道客户端。执行(请求)

写入之前，要创建请求：

tempArgs是要传递给链代码的参数，可以封装，不受参数个数的限制

。

使用pkg/client/channel/chclient.go中的Query()方法查询数据：在查询之前，您还需要创建一个请求。

项目中链码的路径应该是项目名称/chaincode文件夹

例如：

driverfabricdemo/chaincode

。

不要省略项目名，写：chaincode

错误原因：cert的两个字段。URIs和第三方物流。URIs没有定义。

进入tpl对象，/usr/local/go/src/crypto/x509/x509.go是一个结构，找不到URIs字段

。

将go版本从1.9.3升级到1.11.3，然后输入文件/usr/local/go/src/crypto/x509/x509.go再次检查结构内容：

执行sdk的EXECUTE()方法时出错。

方法不存在，一般是因为链代码的Invoke方法中的方法名与EXECUTE()方法传入的方法名不同。

但是可以肯定的是，链代码的Invoke方法中的方法名和项目中的Excute()方法时传入的方法名是一模一样的！但是它&#039；很奇怪，为什么会出现这种错误？使用dockerrmi删除dev-peerx.travle.xq.com的图像，然后再次运行它。

在创建实例化链代码请求时总是提示

。

str(type*cb)cannotbeused.SignaturePolicyEnvelope)asthetype*common.SignaturePolicyEnvelopeless...(？F1)Checkinformation:reportcompoundwordswithincompatibletypesandvalues.

It&#039；很明显是同一类型，但它&#039；it’它总是错的。应该是IDE&#039

的问题。删除供应商文件夹后，不会有提示。只需使用供应商来初始化和添加外部项目！

出现这个错误，一般是配置文件有问题，需要仔细检查

。

Fabric是超级图书联盟推出的核心区块链框架，适用于复杂企业内部和之间构建联盟链。根据超级账本联盟的目标，Fabric构建为支持可插拔组件的模块化基础联盟链框架。

与以太坊的Quorum不同，Fabric从一开始就只考虑了企业间的应用。其独特的渠道概念，将不同子网的企业按照不同的业务目的连接起来，每个子网对应一个渠道。每个频道都有自己独立的区块链。Quorum显然只有一个公网(所有企业节点都加入进来)，企业之间的私有业务通过私有管理器完成。

了解渠道的最简单方法是将其与消息服务提供的主题进行比较。其实Fabic本来就是根据卡夫卡？基于分布式消息服务。

？在光纤网络中。，一个企业可以有一个或多个节点加入整个联盟链；企业可以加入一个或多个通道(子网)；一个节点可以加入一个或多个信道。每个通道构成一个

子网。所以Fabric是一种由子网组成的网络。

那么Fabric是如何实现智能合约的执行，完成业务上行(在区块链记录交易结果)的呢？

与其他画幅不同。Fabric将整个流程分为三个阶段：

业务背书阶段：客户发送的背书节点&#039；的请求完成业务计算(但不更新状态)并完成背书；将背书结果返回给客户。

业务排序阶段：客户端通过通道将背书结果发送给订购方，在此对交易进行排序，打包成块，最终分发到与通道相连的所有节点。

业务验证写入账本阶段：通过八卦网，渠道的所有节点都会收到新的区块，节点会对区块内的每一笔交易进行验证，以确定其是否有效：有效的话，会遵循新的世界状态。，无效将被标记为&quot无效&quot，世界状态不会更新，但整个区块会完整添加到账本中(包括无效交易)。

根据上面的描述，Fabric节点实际上可以分为？、常用节点和订单节点：

？Peer，普通节点，完成背书(仅包括合同执行)和验证。

订购者，对节点进行排序并完成排序。

具有订购者节点的结构网络可以描述如下：

每个通道定义属于该通道的所有节点。，但并不是所有节点都必须连接到订购者节点(私有数据或事务可以通过gossip协议通信在节点之间传播)。

？在区块链，共识是区块链的基础。与公共链不同联盟链的共识要求所有加入账簿的交易都是确定的、最终的，即不能有分叉，区块之间的顺序是确定的，并且只有一条链。在织物中，这一客观要求是通过排序来实现的。，所有的交易都会提交到orderer节点得到某个订单，最后打包成块进入账本。Fabric从1.4.1开始支持基于Raft的排序服务。可以认为共识是基于Raft达成的。

基于RAFT的排序服务与早期的Kafka相比，分布更好，配置更简单，是联盟链中常用的达成共识算法。默认情况下，仲裁使用RAFT作为共识层。简单地说RAFT是领导者和追随者的典范。所有加入RAFT网络的节点在任何时候都有一个领导者。只有这个领导有权决定交易顺序，打包成块。其他节点只能作为跟随者提交事务和同

步块。

基于FAFT网络，每个企业可以有一个或多个节点参与订购。。Frabric中企业之间的网络连接可以改成以下形式：

？区块链的用户在以太网中称为EOA(账户外)，EOA的载体是钱包。。让&#039；让我们按照这个概念来看看Fabric是如何实现用户和启动事务的。结构中的EOA是由ca中心颁发的证书(x.509)。，一个证书代表一个身份(这个和以太坊还是很不一样的，以太坊里的一个EOA其实就是一个哈希地址)EOA可以参与的渠道和授权运营是MSP(会员服务？提供商)(如下图)。

注意：证书是加密验证身份的常见做法；证书包含个人信息、公钥和颁发此证书的CA的签名。。验证者只需要有这个CA的证书(包括CA的公钥)就可以验证签名是否正确，证书的内容是否被篡改。简单来说，通过CA和证书我们可以得到一个可验证的身份和信任链。

？如上图，钱包一般是作为面料中EOA的载体。，一个钱包可以包含多个身份(x.509证书)。身份通过CA提供的信任链验证正确性。

？在验证身份之后，Fabric通过区块链网络中的MSP来解决身份是否代表组织成员以及它在组织中的角色。例如，channel首先验证当前用户标识是否是有效标识。，然后通过MSP来检查企业及其所拥有的角色，最后确定用户是否有权限执行该操作。

可以说Fabric的访问控制是通过MSP完成的。需要在每个需要访问控制的地方定义一个MSP。。例如，每个通道定义一个MSP，它指定通道范围内的资源的访问权限。MSP在Fabric中是一个模糊的概念，也是它提供企业间安全访问的基础。

如前所述，Fabric将业务处理和上网分为三个部分，分别是背书、整理、验证和添加到账本。

背书是Fabric执行智能合同的阶段。在以太坊，智能合约在EVM执行。，支持多种语言。在Fabric中，智能合约被称为链码：链码可以被理解为智能合约的容器，它可以包含一个或多个智能合约。，不适用于EVM，chaincode在JVM或NodeJS中执行。

客户端应用程序通过智能合约访问账簿，每个可访问的智能合约安装在客户端可访问的节点上。，并在通道中定义。(只有合同的节点叫背书节点，没有合同的节点叫未承诺节点，提交节点只维护账簿)

客户端应用提交交易请求，请求到达背书节点。背书节点将首先验证客户&#039；签名以确保客户&#039；的身份有权执行此交易，然后执行交易中提到的智能合约并生成背书响应(或trans-proposal)。。该认可响应通常包含世界状态的读集合和写集合，以及该事务上的节点的签名。这和以太坊联盟链的主要区别在于，背书阶段只是模拟交易，并不真正更新交易结果。。真正的更新事务在第三阶段完成。最后，背书节点将生成的背书响应范慧发送给客户端，智能合同部分的执行结束。

通常，一笔交易的执行需要多方签字。因此，客户端需要将一笔交易发送到多个背书节点，这些背书节点的选择需要满足背书策略的要求。

下图是一个包含客户、背书节点、提交节点的网络示意图。

根据Fabric官方参考文件，客户交易的正果过程可以用下图来描述。

如上图，从1到3是背书阶段，4是整理阶段，4.1、4、2、5是验证提交阶段。参考Frabic&#039s节点概念。你可以进一步了解交易明细的概念。

整体来看，Fabric更侧重于企业。通过以上，你可以对面料的基本构成和概念有一个大致的了解。面料本身并不神秘。，正在使用现有的企业间技术。为了更好的理解，建议参考分布式消息系统和企业安全基础设施(CA相关)的支持。相比以太网联盟链的实现，Fabric&#039的子网概念更适用于复杂的企业间应用程序。但由于其复杂的安全考虑，运营成本非常高。此外，织物&#039；使用证书作为用户身份有很大的局限性。在新的2.0中，织物将在这里发生变化。

下一篇未来我们会看到锯齿吗？由国际米兰提供的区块链框架。联盟链

区块链(1)知道以太坊联盟链

区块链(2)知道Quotum

。

区块链的AllianceChain(3)了解Fabric的AllianceChain

区块链(4)了解锯齿

看完了吗？相信你现在对使用chaincode已经有了基本的了解。！也可以收集页面获取更多使用柴火取暖时需要启动的知识！区块链，虚拟货币，我们是认真的！