IBM

Open by design™

IBM开源技术微讲堂

Hyperledger Fabric v1.4 LTS

第七讲 《应用开发指南》

每周四晚8点直播

课程安排

03/14 区块链赋能产业价值和商业模式
03/21 Hyperledger 项目概览 社区介绍
03/28 Fabric 1.4 LTS 功能介绍 架构概览
04/04 Peer 解析
04/11 Orderer 解析
04/18 MSP 与 CA
04/25 应用开发指南
05/09 部署实践

欢迎关注微信公众号
“IBM开源技术”
获取更多资讯

公众号中发送**“replay”**
获取往期视频地址

欢迎加入社区

社区WIKI https://wiki.hyperledger.org/

中文频道: 中国工作组 https://wiki.hyperledger.org/display/TWGC/Technical+Working+Group+China

加入方法:

如果您愿意**持续**为中国工作组社区做贡献，形式包括推动会议和计划，帮助入门开发者，组织活动推广超级账本技术，贡献源码，翻译文档，我们欢迎您在mail list中踊跃报名

RocketChat聊天室: https://chat.hyperledger.org/channel/twg-china

自我介绍

David Liu刘宇翔

- CTO@Mediconcen
- Technical Ambassador@Hyperledger
- fabric(sdk-node) contributor
- Hyperledger HK meetup organizer

Outline

开发周期

技术栈

Chaincode LifeCycle

DevOps LifeCycle

引用，鸣谢：04/04 李春玲： 《Peer 解析》

开发周期

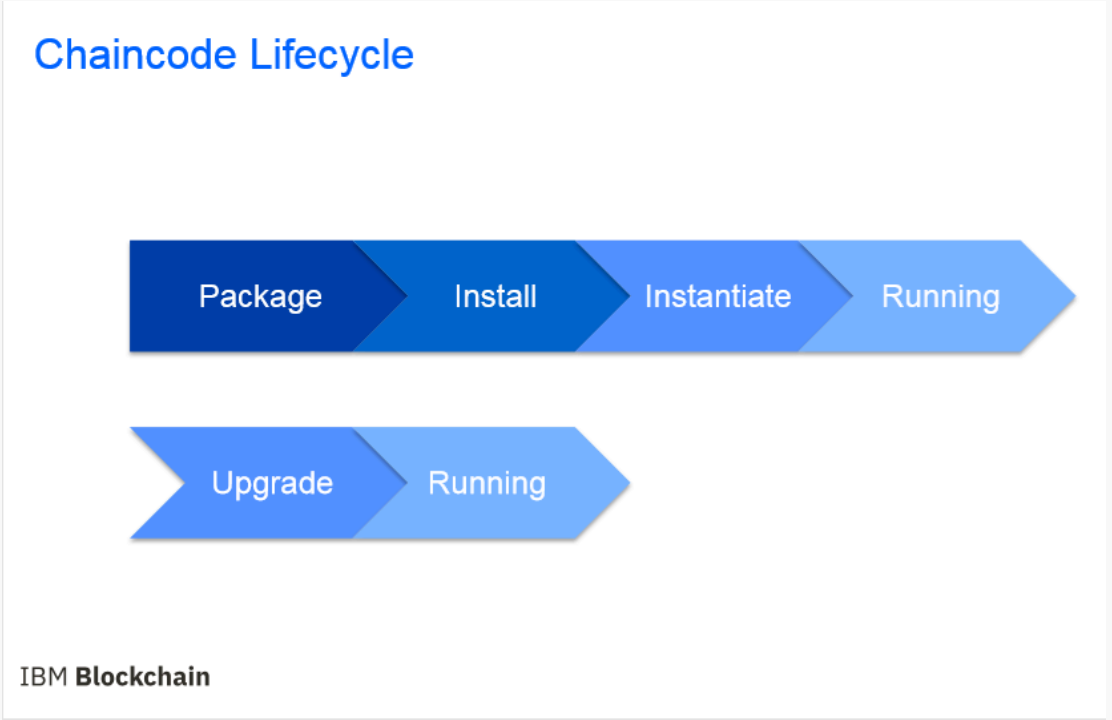
节点部分

1. Generate crypto material[CA]
2. Generate genesis block file
3. Start orderer and peer

通道部分

1. Generate channel bootstrap file
2. Create channel
3. Peer join channel

4. Chaincode lifeCycle



技术栈

- 经典软件工程管理
  - 依赖管理 govendor/dep, npm/yarn, gradle/maven, pip
  - 异常处理 defer，async/await
  - 测试流水线 Smoke, Unit/Mock, SI tests
- 精通PKI密码体系
  - ECDSA
  - X509
  - HSM和pkcs11

技术栈（chaincode开发）

- golang/Nodejs/java

技术栈（devOps）

- Certified Hyperledger Fabric Administrator: CLI in Unix/Linux
- Java/Golang/Nodejs/Python
- Docker
- GRPC

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Chaincode API

Minimum Golang chaincode

```
//failed to invoke chaincode name:"lscc" , error: API error (400): OCI runtime create failed:
container_linux.go:348: starting container process caused "exec: \"chaincode\": executable file not found in
$PATH": unknown
package main
type StressChaincode struct{}
//called when initialize, upgrade
func (t *StressChaincode) Init(stub shim.ChaincodeStubInterface) peer.Response {
                return shim.Success(nil)
}
//called when query, phase-1 of invoke
func (t *StressChaincode) Invoke(stub shim.ChaincodeStubInterface) peer.Response {
                return shim.Success(nil)
}
func main() {
                err:=shim.Start(new(StressChaincode))
}
```

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Minimum nodejs chaincode

Chaincode.js

```
const {Shim} = require('fabric-shim');
class Chaincode {
  constructor() {}
  async Init(stub) {
     return Shim.success();
  }
  async Invoke(stub) {
     return Shim.success();
  }
}

Shim.start(new Chaincode());
```

Package.json

```
{
 "scripts": {
   "start": "node chaincode.js"
 },
 "dependencies": {
   "fabric-shim": "^1.4.1",
   "fabric-shim-crypto": "^1.4.1"
 }
}
```

shim.ChaincodeStubInterface

Params

```
GetArgs() [][]byte

GetFunctionAndParameters() (string, []string)//returns 1st argument as function, ”fcn”, rest as params
```

Access world states

```
GetState(key string)

PutState(key string, value []byte)

      DelState(key string)
```

Witness the immutable

```
GetHistoryForKey(key string) Iterator<KeyModification>

GetHistoryForKey returns a history of key across time.

GetHistoryForKey requires peer configuration core.ledger.history.enableHistoryDatabase=true.

type KeyModification struct {
  TxId      string
  Value     []byte
  Timestamp TimeLong  // timestamp provided by the client in the proposal header.
  IsDelete  bool
}
```

List for NoSQL: key range

```
GetStateByRange(startKey, endKey string) iterator<KV>

    This is also used to get all keys(exclude compositeKey)
    Size limit:100
type KV struct {
  Namespace    string   // same as chaincode ID
  Key          string
  Value        []byte
}
```

Composite key

Composite key: joining of key with unprintable char,

```
CreateCompositeKey(objectType string, attributes []string) (string, error)

SplitCompositeKey(compositeKey string) (string, []string, error)
```

Usage: extends data namespace, prevent key overlap

CouchDB or default levelDB

prerequisite: config in peer

```
'CORE_LEDGER_STATE_STATEDATABASE=CouchDB',
      `CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=${container_name}:5984`,
      `CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=${user}`,
      `CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=${password}`
```

Pros and Cons

- Rich query: `GetQueryResult(query string)`
  - 利用couchdb的语法按查询规则做值检索
- Performance impact from network
- Couchdb only stored world states
- 需要预置索引集合couchdb metadata: index.json
- security leakage and operation complexity

func Init

1. 不可查询性set payload in peer.response in vain [TODO]
2. Chaincode更新时会被执行Chaincode upgrade will not reset ledger, unless...
3. privateData未就绪

   stub.putPrivateData==>Error: collection config not define for namespace [collectionName]
   See also in https://github.com/hyperledger/fabric/commit/8a705b75070b7a7021ec6f897a80898abf6a1e45

开发模式

Fabric结点运行模式

- 一般模式
  - Chaincode运行在docker容器中
  - 开发调试过程非常繁杂
    - 部署 -> 调试 -> 修改 -> 创建docker镜像 -> 部署 -> ...
- 开发模式：--peer-chaincodedev
  - Chaincode运行在本地
  - 开发调试相对容易
    - 部署 -> 调试 -> 修改 -> 部署 -> ...

外部配合:
orderer(dev genesis), TLS

构造容器的45秒

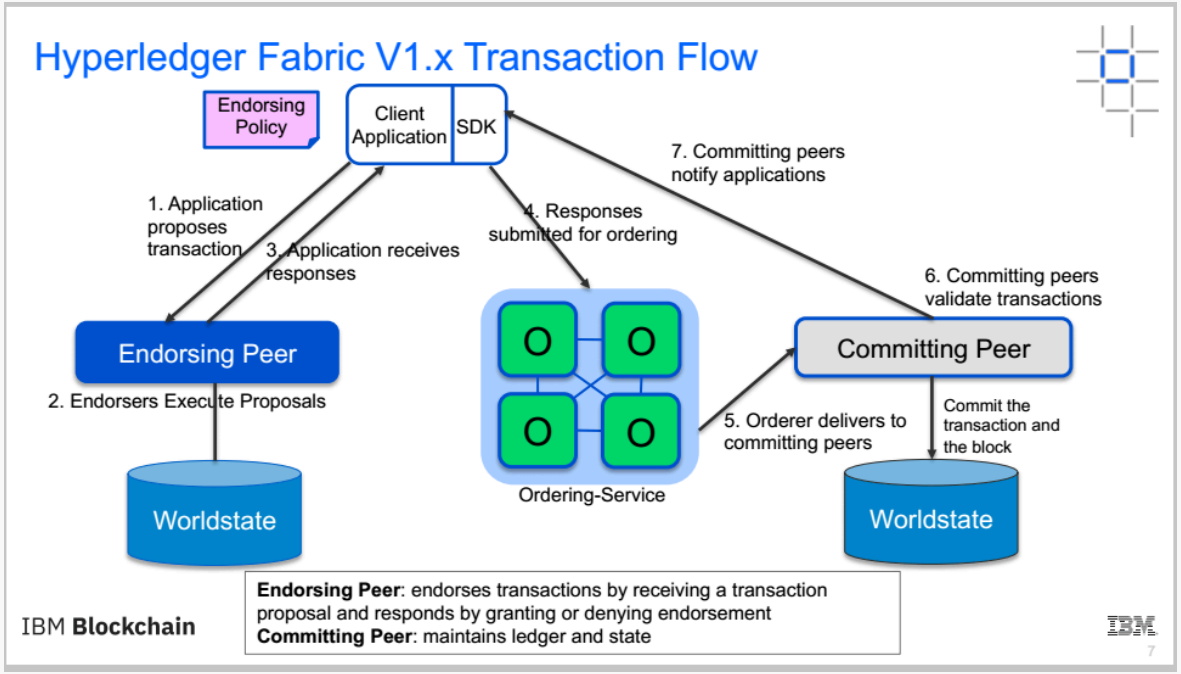
用MockStub代替开发模式
- couchdb

用chaincode upgrade代替部署

不支持docker image "fabric-peer"

交易流程

交易流程图


Hyperledger Fabric V1.x Transaction Flow
Endorsing Policy
Client Application
SDK
1. Application proposes transaction
2. Endorsers Execute Proposals
3. Application receives responses
4. Responses submitted for ordering
5. Orderer delivers to committing peers
6. Committing peers validate transactions
7. Committing peers notify applications
Endorsing Peer
Committing Peer
Ordering-Service
Worldstate
Worldstate
Commit the transaction and the block
IBM Blockchain
Endorsing Peer: endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement
Committing Peer: maintains ledger and state
IBM

Invoke vs Query

Invoke 2-phase commit:

1. phase-1: sendTransactionProposal
    a. build chaincode container “dev-*” if not exist
        Tricky: unexpected long time to wait for first invoke
    b. Exec Invoke func in your chaincode
2. phase-2: sendTransaction
3. Optional: wait for eventHub response

Query: phase-1 only

HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

复合API

Cross chaincode invoke

`InvokeChaincode(calledChaincode, args, channel) peer.Response`

- InvokeChaincode doesn't create another new transaction
- If the calledChaincode is on the same channel, it simply adds the calledChaincode ReadWriteSet to the calling transaction.
- If the calledChaincode is on a different channel, only the Response is returned to the calling chaincode, not affecting this ReadWriteSet. Act like a “Query”, PutState will not have any effect

Context invariant

-[CID,transient Map,TxID,timestamp]

Basis of CID

CID: Client Identity Chaincode Library

```
GetCreator() ([]byte, error)
```

```
// ClientIdentityImpl implements the ClientIdentity interface
type clientIdentityImpl struct {
  stub  ChaincodeStubInterface
  mspID string
  cert  *x509.Certificate
  attrs *attrmgr.Attributes
}
//交易签名 ！== 背书
```

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

GetTransient

GetTransient returns the `ChaincodeProposalPayload.Transient` field.

It is a map that contains data (e.g. cryptographic material) that might be used to implement some form of application-level confidentiality. The contents of this field, as prescribed by `ChaincodeProposalPayload`, are supposed to always be omitted from the transaction and excluded from the ledger.

例子：fabric-shim-crypto

PrivateData

Prerequisite

- manually set anchor peers

- predefined collection: {permissioned orgs,requiredPeerCount,maxPeerCount}

- only `OR` is allowed in collection policy ("2-of" forbidden)

Key level endorsement

GetStateValidationParameter(key string) ([]byte, error)

SetStateValidationParameter(key string, ep []byte) error

KeyLevelEndorsement

只支持“与”关系

覆盖合约级endorsement policy(EP)

If a key is modified and a key-level endorsement policy is present, the key-level endorsement policy overrides the chaincode-level endorsement policy.


HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS
DevOps

开发周期

节点部分

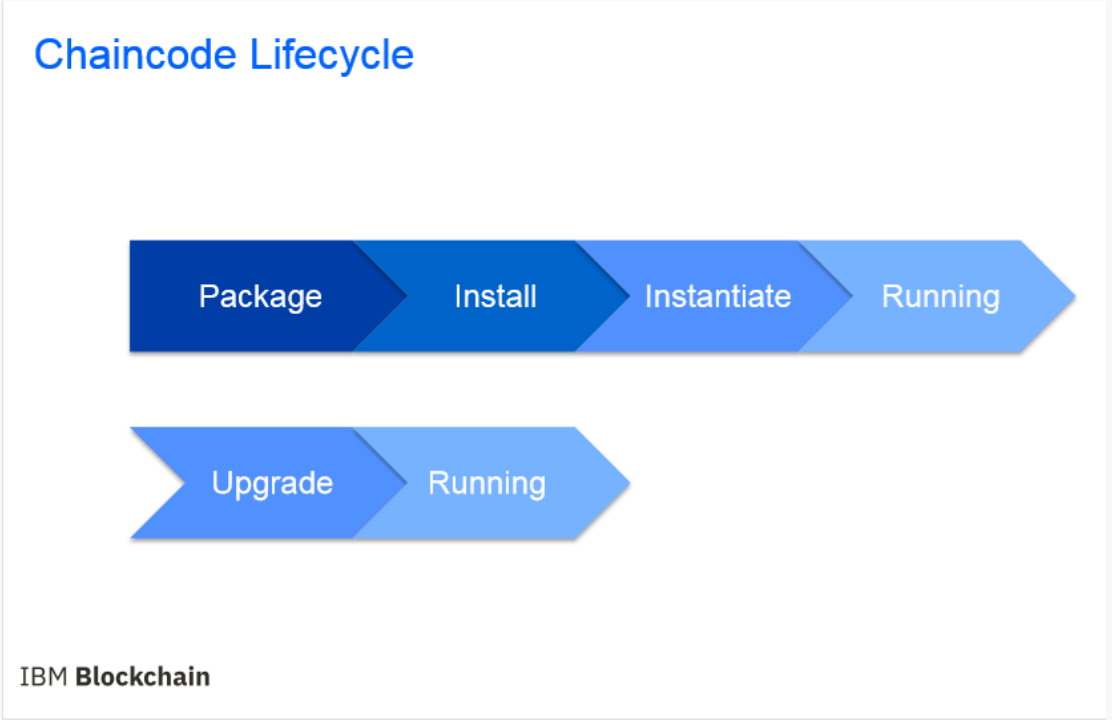
1. Generate crypto material[CA]
2. Generate genesis block file
3. Start orderer and peer

通道部分

1. Generate channel bootstrap file
2. Create channel
3. Peer join channel

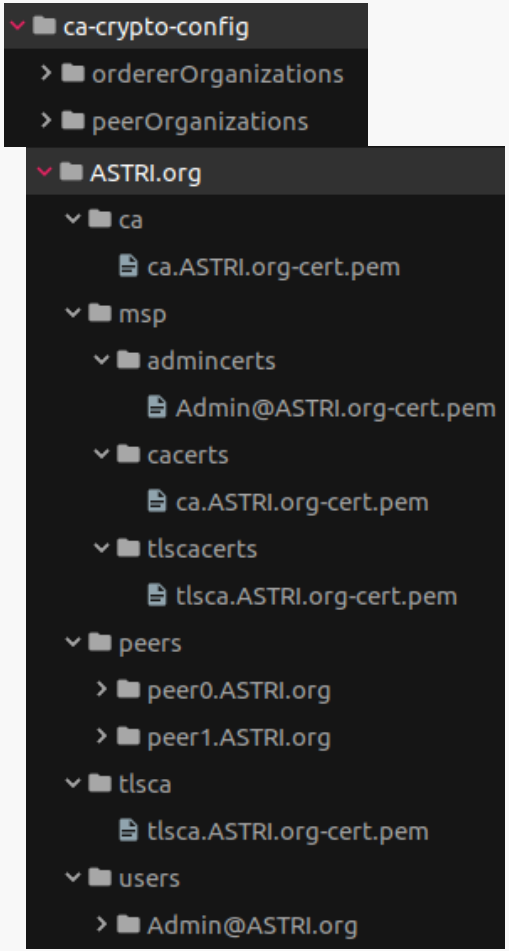
Chaincode部分

1. Chaincode lifeCycle



节点部分 Generate crypto material[CA]

1. Cryptogen
2. Fabric ca based
   a. Restful http server “fabric-ca-server” (Binary, docker image ‘fabric-ca’)
   b. “fabric-ca-client” (Binary, helper in sdks)
      i. $ npm fabric-ca-client



Restful http server “fabric-ca-server”

Fabric-ca-server

- CA Service
  - Register, revoke
  - Enroll, reenroll, generate CRL,
- Identity Service
- Certificate Service
- Affiliation Service

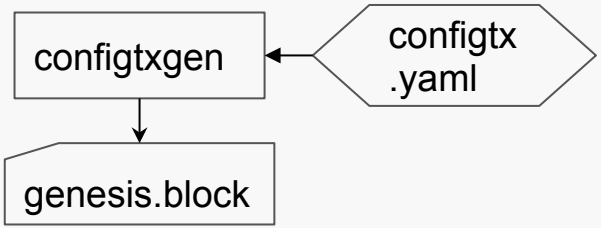
节点部分 Generate genesis block file

Binary: configtxgen

channelName=testchainid

configtxgen -outputBlock $outputFile -profile $profile -channelID $channelName

Genesis block即为系统通道的起始配置



节点部分 Start orderer and peer

Orderer最小配置

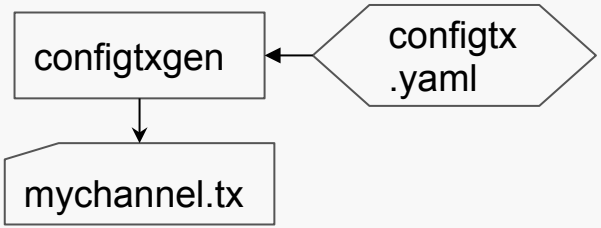
```
'ORDERER_GENERAL_LISTENADDRESS=0.0.0.0', // used to self identify

`ORDERER_GENERAL_TLS_ENABLED= {!!tls}`,

'ORDERER_GENERAL_GENESISMETHOD=file',

`ORDERER_GENERAL_GENESISFILE= {                    }/ {BLOCK_FILE}`, -----

`ORDERER_GENERAL_LOCALMSPID= {id}`, -----

`ORDERER_GENERAL_LOCALMSPDIR= {configPath}`, -----
```

Peer最小配置

通道部分 Generate channel bootstrap file

Binary: configtxgen

configtxgen -outputCreateChannelTx $outputFile -profile $profile -channelID $channelName



通道部分 Create channel

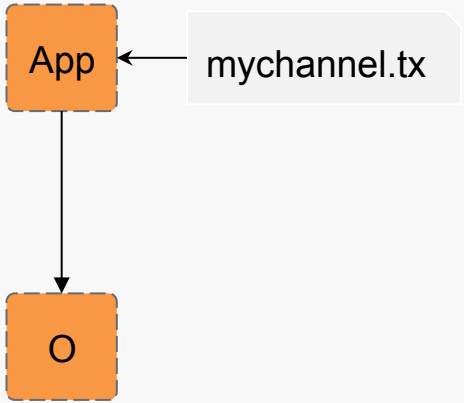
Client.js#createChannel(request)

{

return this._createOrUpdateChannel

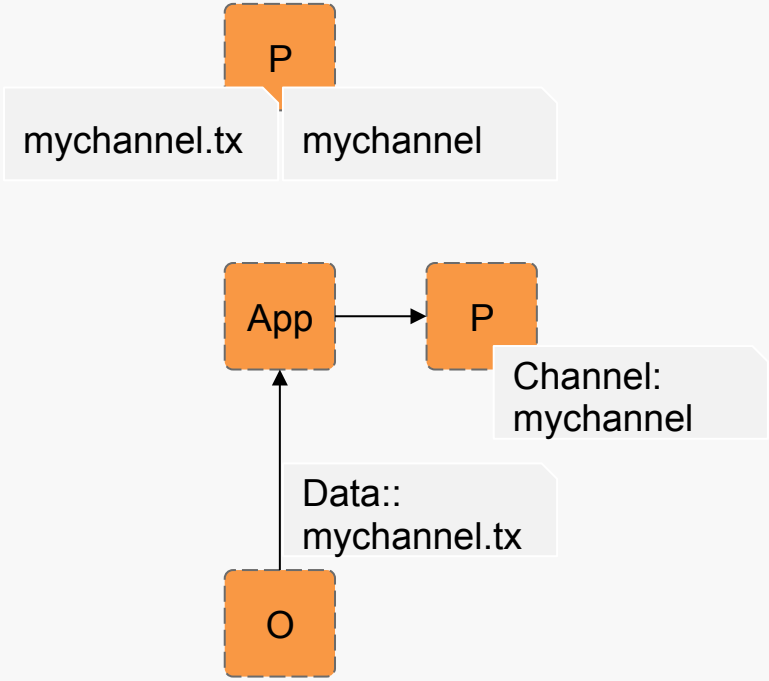
(request, request && request.envelope);

}



通道部分 Peer join channel

1. $ peer channel join
2. Channel.js#joinChannel



Chaincode lifeCycle

install chaincode

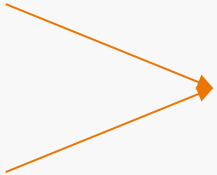
1. Commands: peer chaincode
    a. $ peer chaincode package -s -S -i "AND('OrgA.admin')" ... ccpack.out
    The “-s -S” option creates a package that can be signed by multiple owners.Otherwise, the process will create a SignedCDS that includes only the instantiation policy in addition to the CDS.
    a. $ peer chaincode signpackage ccpack.out signedccpack.out
1. Client.js#installChaincode[SignedCDS] default instantiate policy

Params: chaincode name/id, peer, path, version, [instantiate policy] [metadataPath]

Chaincode lifeCycle

Instantiate chaincode -> sendInstantiateProposal

Upgrade chaincode -> sendUpgradeProposal



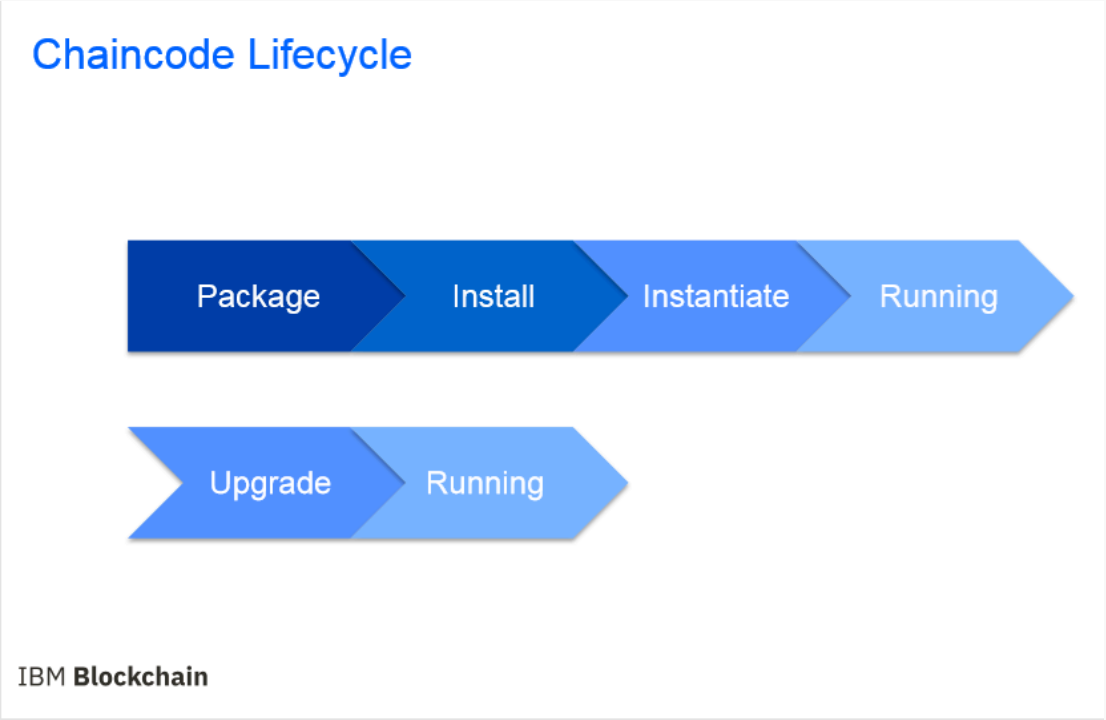
channel._sendChaincodeProposal

Chaincode lifeCycle

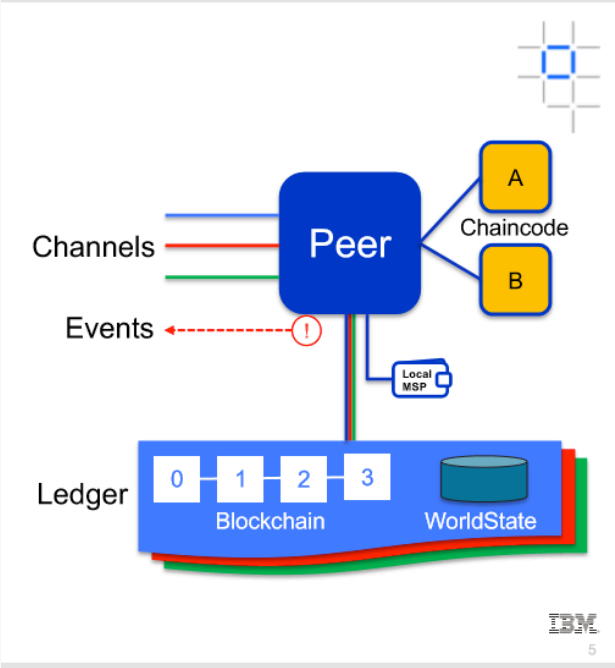
upgrade chaincode

用途：
更新endorsement policy， collection policy
更新合约内容，重新执行func Init

[package ->] install -> upgrade
以另一个chaincode version



Peer


Channels
Events
Peer
A
Chaincode
B
Local MSP
Ledger
0
1
2
3
Blockchain
WorldState
IBM
5

课程安排

03/14 区块链赋能产业价值和商业模式
03/21 Hyperledger 项目概览 社区介绍
03/28 Fabric 1.4 LTS 功能介绍 架构概览
04/04 Peer 解析
04/11 Orderer 解析
04/18 MSP 与 CA
04/25 应用开发指南
05/09 部署实践

欢迎关注微信公众号
“IBM开源技术”
获取更多资讯

公众号中发送**“replay”**
获取往期视频地址