peer chaincode

`peer chaincode`命令允许管理员执行与一个节点上运行有关的链码，例如安装，实例化，调用，包装，查询和升级链码。

语法

`peer chaincode`命令有以下子命令：

- install
- instantiate
- invoke
- list
- package
- query
- signpackage
- upgrade

不同的子命令选项（安装，实例化）牵涉到与一个peer相关的不同链码操作。例如，用`peer chaincode install`子命令选项在节点上安装一个链码，或者用`peer chaincode query`子命令选项为一节点上账本的当前值查询链码。

本主题将描述每个节点链码子命令以及它们的选项。

参数

每个子命令都有一套专门对应各子命令的参数，以及一套涉及到所有`peer chaincode`子命令的全局参数。但并不是所有的子命令都会使用这些参数。比如，`query`子命令就不需要`--orderer`参数。

每个参数都会和与其相关的子命令一起来解析。全局参数包括

* `--cafile <string>`

是通往一文件的路径，该文件包含了用于排序端点的PEM编码受信任证书

* `--certfile <string>`

是通往一文件的路径，该文件包含了用于和orderer端点进行相互TLS通信的PEM编码X509公钥。

* `--keyfile <string>`

是通往一文件的路径，该文件包含了用于和orderer端点进行相互TLS通信的PEM编码私钥

* `-o` or `--orderer <string>`

排序服务端点被指明为`<hostname or IP address>:<port>`

* `--ordererTLSHostnameOverride <string>`

验证与orderer的TLS连接时要用到的主机名覆盖

* `--tls`

当与orderer端点通信时用TLS

* `--transient <string>`

JSON编码中的参数的瞬时映射

peer chaincode install

```
Install a chaincode on a peer. This installs a chaincode deployment spec package (if provided) or

Usage:
  peer chaincode install [flags]

Flags:
      --connectionProfile string       Connection profile that provides the necessary connection
  -c, --ctor string                    Constructor message for the chaincode in JSON format (defa
  -h, --help                           help for install
  -l, --lang string                    Language the chaincode is written in (default "golang")
  -n, --name string                    Name of the chaincode
  -p, --path string                    Path to chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
  -v, --version string                 Version of the chaincode specified in install/instantiate/

Global Flags:
      --cafile string                          Path to file containing PEM-encoded trusted certifica
      --certfile string                        Path to file containing PEM-encoded X509 public key t
      --clientauth                             Use mutual TLS when communicating with the orderer en
      --connTimeout duration                   Timeout for client to connect (default 3s)
      --keyfile string                         Path to file containing PEM-encoded private key to us
  -o, --orderer string                         Ordering service endpoint
      --ordererTLSHostnameOverride string      The hostname override to use when validating the TLS
      --tls                                    Use TLS when communicating with the orderer endpoint
      --transient string                       Transient map of arguments in JSON encoding
```

peer chaincode instantiate

```
Deploy the specified chaincode to the network.

Usage:
  peer chaincode instantiate [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
      --connectionProfile string       Connection profile that provides the necessary connection
  -c, --ctor string                    Constructor message for the chaincode in JSON format (defa
  -E, --escc string                    The name of the endorsement system chaincode to be used fo
  -h, --help                           help for instantiate
  -l, --lang string                    Language the chaincode is written in (default "golang")
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
  -P, --policy string                  The endorsement policy associated to this chaincode
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
```

```
  -v, --version string                 Version of the chaincode specified in install/instantiate/
  -V, --vscc string                    The name of the verification system chaincode to be used f

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode invoke

```
Invoke the specified chaincode. It will try to commit the endorsed transaction to the network.

Usage:
  peer chaincode invoke [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --connectionProfile string       Connection profile that provides the necessary connection
  -c, --ctor string                    Constructor message for the chaincode in JSON format (defa
  -h, --help                           help for invoke
  -I, --isInit                         Is this invocation for init (useful for supporting legacy
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
      --waitForEvent                   Whether to wait for the event from each peer's deliver fil
      --waitForEventTimeout duration   Time to wait for the event from each peer's deliver filter

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode list

```
Get the instantiated chaincodes in the channel if specify channel, or get installed chaincodes on

Usage:
  peer chaincode list [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --connectionProfile string       Connection profile that provides the necessary connection
  -h, --help                           help for list
      --installed                      Get the installed chaincodes on a peer
      --instantiated                   Get the instantiated chaincodes on a channel
      --peerAddresses stringArray      The addresses of the peers to connect to
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode package

```
Package a chaincode and write the package to a file.

Usage:
  peer chaincode package [outputfile] [flags]

Flags:
  -s, --cc-package                 create CC deployment spec for owner endorsements instead of r
  -c, --ctor string                Constructor message for the chaincode in JSON format (default
  -h, --help                       help for package
  -i, --instantiate-policy string  instantiation policy for the chaincode
  -l, --lang string                Language the chaincode is written in (default "golang")
  -n, --name string                Name of the chaincode
  -p, --path string                Path to chaincode
  -S, --sign                       if creating CC deployment spec package for owner endorsements
  -v, --version string             Version of the chaincode specified in install/instantiate/upg

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode query

```
Get endorsed result of chaincode function call and print it. It won't generate transaction.

Usage:
  peer chaincode query [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --connectionProfile string       Connection profile that provides the necessary connection
  -c, --ctor string                    Constructor message for the chaincode in JSON format (defa
  -h, --help                           help for query
  -x, --hex                            If true, output the query value byte array in hexadecimal.
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
  -r, --raw                            If true, output the query value as raw bytes, otherwise fo
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode signpackage

```
Sign the specified chaincode package

Usage:
  peer chaincode signpackage [flags]

Flags:
  -h, --help   help for signpackage
```

```
Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

peer chaincode upgrade

```
Upgrade an existing chaincode with the specified one. The new chaincode will immediately replace

Usage:
  peer chaincode upgrade [flags]

Flags:
  -C, --channelID string                The channel on which this command should be executed
      --collections-config string       The fully qualified path to the collection JSON file inclu
      --connectionProfile string        Connection profile that provides the necessary connection
  -c, --ctor string                     Constructor message for the chaincode in JSON format (defa
  -E, --escc string                     The name of the endorsement system chaincode to be used fo
  -h, --help                            help for upgrade
  -l, --lang string                     Language the chaincode is written in (default "golang")
  -n, --name string                     Name of the chaincode
  -p, --path string                     Path to chaincode
      --peerAddresses stringArray       The addresses of the peers to connect to
  -P, --policy string                   The endorsement policy associated to this chaincode
      --tlsRootCertFiles stringArray    If TLS is enabled, the paths to the TLS root cert files of
  -v, --version string                  Version of the chaincode specified in install/instantiate/
  -V, --vscc string                     The name of the verification system chaincode to be used f

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
      --transient string                    Transient map of arguments in JSON encoding
```

使用示例

peer chaincode instantiate 示例

以下是`peer chaincode instantiate`命令的一些例子，它们在`1.0`版本中`mychannel`通道上将名为`mycc`的链码实例化：

- 用`--tls`和`--cafile`全局变量来对网络中的链码实例化，其中TLS被启用：

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizati
peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile $ORDERER_CA -C mychanne

2018-02-22 16:33:53.324 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default
2018-02-22 16:33:53.324 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default
2018-02-22 16:34:08.698 UTC [main] main -> INFO 003 Exiting.....
```

- 仅用命令专门选项来将网络中的链码实例化，其中TLS未被启用：

```
peer chaincode instantiate -o orderer.example.com:7050 -C mychannel -n mycc -v 1.0 -c '{"Args


2018-02-22 16:34:09.324 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default
2018-02-22 16:34:09.324 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default
2018-02-22 16:34:24.698 UTC [main] main -> INFO 003 Exiting.....
```

peer chaincode invoke 示例

以下是`peer chaincode invoke`命令的一个范例：

- 调用版本`1.0`名为`mycc`的链码，该链码位于`peer0.org1.example.com:7051`和`peer0.org2.example.com:9051`（节点由 –peerAddresses 上的通道`mychannel`中，请求从变量`a`中转移10个单位到变量`b`中：

```
peer chaincode invoke -o orderer.example.com:7050 -C mychannel -n mycc --peerAddresses peer0.

2018-02-22 16:34:27.069 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default
2018-02-22 16:34:27.069 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default
.
.
.
2018-02-22 16:34:27.106 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> DEBU 00a ESCC invoke res
2018-02-22 16:34:27.107 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 00b Chaincode invok
2018-02-22 16:34:27.107 UTC [main] main -> INFO 00c Exiting.....
```

现在你就能看到该调用在日志信息的基础上被成功上传了：

```
2018-02-22 16:34:27.107 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 00b Chaincode invoke s
```

一个成功的回复反映了该交易被成功提交至排序服务。随后，该交易会被添加至区块中，最后，接受通道上每个节点的验证，若不通过则被视为无效。

peer chaincode list 示例

以下是`peer chaincode list`命令的一些例子：

- 用`--installed`flag来对安装在节点上的链码进行列表。

```
peer chaincode list --installed

Get installed chaincodes on peer:
Name: mycc, Version: 1.0, Path: github.com/hyperledger/fabric-samples/chaincode/abstore/go, I
2018-02-22 17:07:13.476 UTC [main] main -> INFO 001 Exiting.....
```

  可以看到该节点安装了一个名为`mycc`的链码，它是版本`1.0`的。
- 用`--instantiated`和`-C`（通道ID）flag一起来对通道上被实例化的链码进行列表。

```
peer chaincode list --instantiated -C mychannel

Get instantiated chaincodes on channel mychannel:
Name: mycc, Version: 1.0, Path: github.com/hyperledger/fabric-samples/chaincode/abstore/go, E
2018-02-22 17:07:42.969 UTC [main] main -> INFO 001 Exiting.....
```

现在你能看到版本`1.0`的链码`mycc`被在`mychannel`通道上实例化了。

peer chaincode package 示例

以下是`peer chaincode package`命令的一个例子，它打包了版本为`1.0`名为`mycc`的链码，生成了链码部署规定，用本地MSP签署了该包装，同时还将其输出为`ccpack.out`：

```
peer chaincode package ccpack.out -n mycc -p github.com/hyperledger/fabric-samples/chaincode/a
.
.
.
2018-02-22 17:27:01.404 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default e
2018-02-22 17:27:01.405 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default v
.
.
.
2018-02-22 17:27:01.879 UTC [chaincodeCmd] chaincodePackage -> DEBU 011 Packaged chaincode int
2018-02-22 17:27:01.879 UTC [main] main -> INFO 012 Exiting.....
```

peer chaincode query 示例

以下是`peer chaincode query`命令的示例，该命令在节点账本上查询版本`1.0`名为`mycc`的链码，查询变量`a`的值：

- 从输出中可看出变量`a`在查询时有一个值是90。

```
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'

2018-02-22 16:34:30.816 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default
2018-02-22 16:34:30.816 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default
Query Result: 90
```

peer chaincode signpackage 示例

以下是`peer chaincode signpackage`命令的示例，它接受了现存的签名包，还创建了一个新的签名包，上面有本地MSP追加的一个签名。

```
peer chaincode signpackage ccwith1sig.pak ccwith2sig.pak
Wrote signed package to ccwith2sig.pak successfully
2018-02-24 19:32:47.189 EST [main] main -> INFO 002 Exiting.....
```

peer chaincode upgrade 示例

以下是`peer chaincode upgrade`命令的示例，它在通道`mychannel`上将版本`1.1`名为`mycc`的链码升级成版本`1.2`，其中包含了一个新的变量`c`：

- 启用TLS，用`--tls`和`--cafile`global flags来升级网络中的链码：

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizati
peer chaincode upgrade -o orderer.example.com:7050 --tls --cafile $ORDERER_CA -C mychannel -n
.
.
.
```

```
2018-02-22 18:26:31.433 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default
2018-02-22 18:26:31.434 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default
2018-02-22 18:26:31.435 UTC [chaincodeCmd] getChaincodeSpec -> DEBU 005 java chaincode enable
2018-02-22 18:26:31.435 UTC [chaincodeCmd] upgrade -> DEBU 006 Get upgrade proposal for chain
.
.
.
2018-02-22 18:26:46.687 UTC [chaincodeCmd] upgrade -> DEBU 009 endorse upgrade proposal, get
.
.
.
2018-02-22 18:26:46.693 UTC [chaincodeCmd] upgrade -> DEBU 00c Get Signed envelope
2018-02-22 18:26:46.693 UTC [chaincodeCmd] chaincodeUpgrade -> DEBU 00d Send signed envelope
2018-02-22 18:26:46.908 UTC [main] main -> INFO 00e Exiting.....
```

- 不启用TLS，仅用命令专门选项来升级网络中的链码：

```
peer chaincode upgrade -o orderer.example.com:7050 -C mychannel -n mycc -v 1.2 -c '{"Args":["
.
.
.
2018-02-22 18:28:31.433 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default
2018-02-22 18:28:31.434 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 004 Using default
2018-02-22 18:28:31.435 UTC [chaincodeCmd] getChaincodeSpec -> DEBU 005 java chaincode enable
2018-02-22 18:28:31.435 UTC [chaincodeCmd] upgrade -> DEBU 006 Get upgrade proposal for chain
.
.
.
2018-02-22 18:28:46.687 UTC [chaincodeCmd] upgrade -> DEBU 009 endorse upgrade proposal, get
.
.
.
2018-02-22 18:28:46.693 UTC [chaincodeCmd] upgrade -> DEBU 00c Get Signed envelope
2018-02-22 18:28:46.693 UTC [chaincodeCmd] chaincodeUpgrade -> DEBU 00d Send signed envelope
2018-02-22 18:28:46.908 UTC [main] main -> INFO 00e Exiting.....
```