
ELEMENTAL CONCEPT

Hyperledger Fabric Workshop

Chaincode and Application Development

Paul Sitoh

10-11 November 2018

Agenda

- Prerequisite
- Go chaincode programming
- Install and instantiate chaincode
- Fabric Node SDK
- Hyperledger Explorer
- Fabric Development Kit -- Roadmap
- Q & A

Prerequisite

Chaincode and Application Development

Prerequisite

- Install Git
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- Install Docker
  - https://tuleap-documentation.readthedocs.io/en/latest/developer-guide/quick-start/install-docker.html
- Install Go
  - https://golang.org/doc/install#install
- Install Fabric Developer Kit
  - https://github.com/aladdinid/fabric-devkit

Go chaincode

Chaincode and Application Development

Go chaincode

The smallest unit of chaincode

```
package main

import (
    "fmt"

    "github.com/hyperledger/fabric/core/chaincode/shim"
    pb "github.com/hyperledger/fabric/protos/peer"
)

type SimpleChaincode struct{}

func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    return shim.Success([]byte("Init called"))
}

func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    return shim.Success([]byte("Invoke called"))
}

func main() {
    err := shim.Start(new(SimpleChaincode))
    if err != nil {
        fmt.Printf("Error: %s", err)
    }
}
```

Go chaincode

ChaincodeStubInterface

- GetFunctionAndParameters() (string, []string)
- GetState(key string) ([]byte, error)
- PutState(key string, value []byte) error
- https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim#ChaincodeStubInterface

Go chaincode

```go
// SimpleChaincode representing a class of chaincode
type SimpleChaincode struct{}

// Init to initiate the SimpleChaincode class
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    logger.Info("Hello Init")
    fcn, _ := stub.GetFunctionAndParameters()
    if fcn == "init" {
        return initialise(stub)
    }

    return shim.Error("Fail to initialise state")
}

// Invoke a method specified in the SimpleChaincode class
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    logger.Info("Hello Invoke")
    fcn, _ := stub.GetFunctionAndParameters()

    if fcn == "pay" {
        return pay(stub)
    }

    if fcn == "query" {
        return query(stub)
    }

    return shim.Success([]byte("Invoke"))
}
```

```go
// Transaction
func initialise(stub shim.ChaincodeStubInterface) pb.Response {

    args := stub.GetArgs()

    name1 := string(args[1])
    amount1 := args[2]

    logger.Infof("Name1: %s Amount1: %s", name1, string(amount1))

    err := stub.PutState(name1, amount1)
    if err != nil {
        return shim.Error(fmt.Sprintf("Failed to store state %v", err))
    }

    name2 := string(args[3])
    amount2 := args[4]

    logger.Infof("Name2: %s Amount2: %s", name2, string(amount2))

    err = stub.PutState(name2, amount2)
    if err != nil {
        return shim.Error(fmt.Sprintf("Failed to store state %v", err))
    }

    return shim.Success([]byte("Initialisation completed"))
}
```

Go chaincode

Packaging for chaincode deployment

```
$GOPATH/
    src/
        github.com/user/another-repo/
            support/
                superduper-support.go
            superduper-algo.go
        github.com/user/repo/
            mychaincode/
                chaincode.go
                util/
                    mymaths.go
                vendor/
                    github.com/user/another-repo
                        support/
                            superduper-support.go
                        superduper-algo.go
                    vendor.json
```

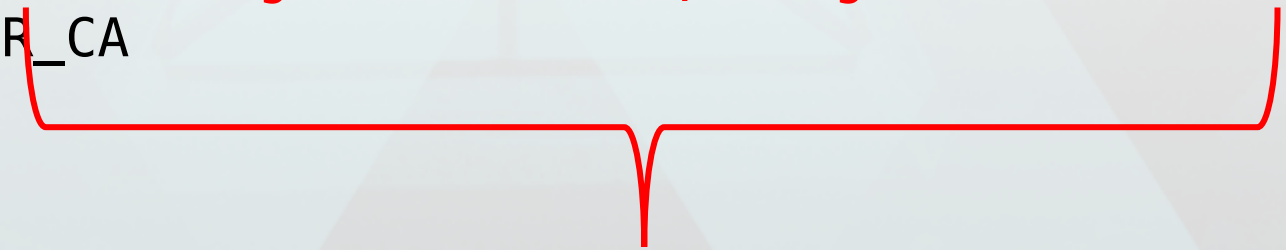
Install and instantiate chaincode

Chaincode and Application Development

Install and instantiate chaincode

```
peer chaincode install -n $CHAINCODEID -v $CHAINCODE_VERSION -l
$CHAINCODE_LANG -p $path_to_chaincode --tls --cafile $ORDERER_CA
```

```
peer chaincode instantiate -o $ORDERER -C $CHANNELNAME -n
$CHAINCODEID -l $CHAINCODE_LANG -v $CHAINCODE_VERSION -c
$constructor -P "OR ('Org1MSP.member', 'Org2MSP.member')" --tls
--cafile $ORDERER_CA
```

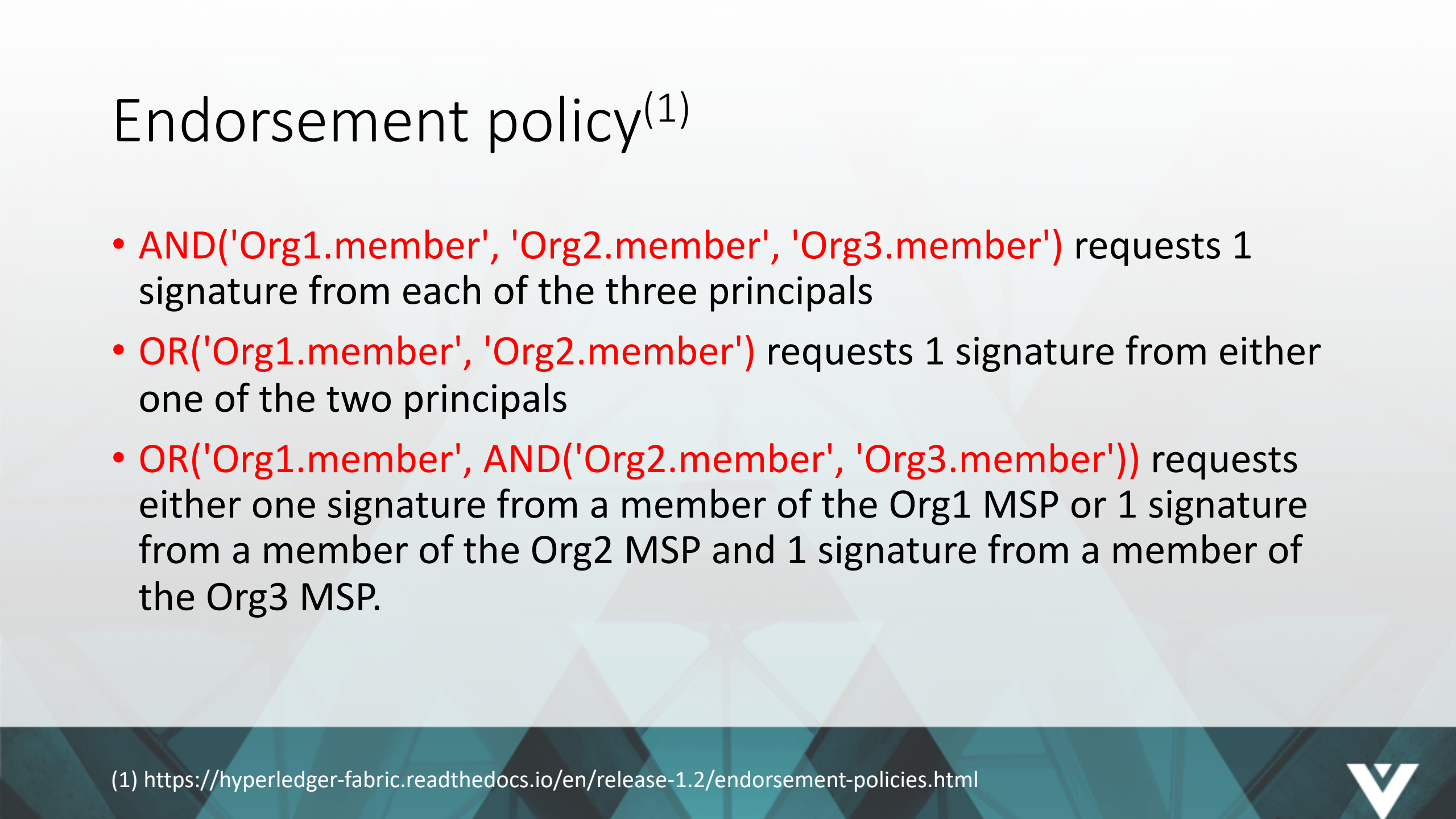


Endorsement Policy

Endorsement policy(1)

- AND('Org1.member', 'Org2.member', 'Org3.member') requests 1 signature from each of the three principals
- OR('Org1.member', 'Org2.member') requests 1 signature from either one of the two principals
- OR('Org1.member', AND('Org2.member', 'Org3.member')) requests either one signature from a member of the Org1 MSP or 1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP.

(1) https://hyperledger-fabric.readthedocs.io/en/release-1.2/endorsement-policies.html



Endorsement policy(1)

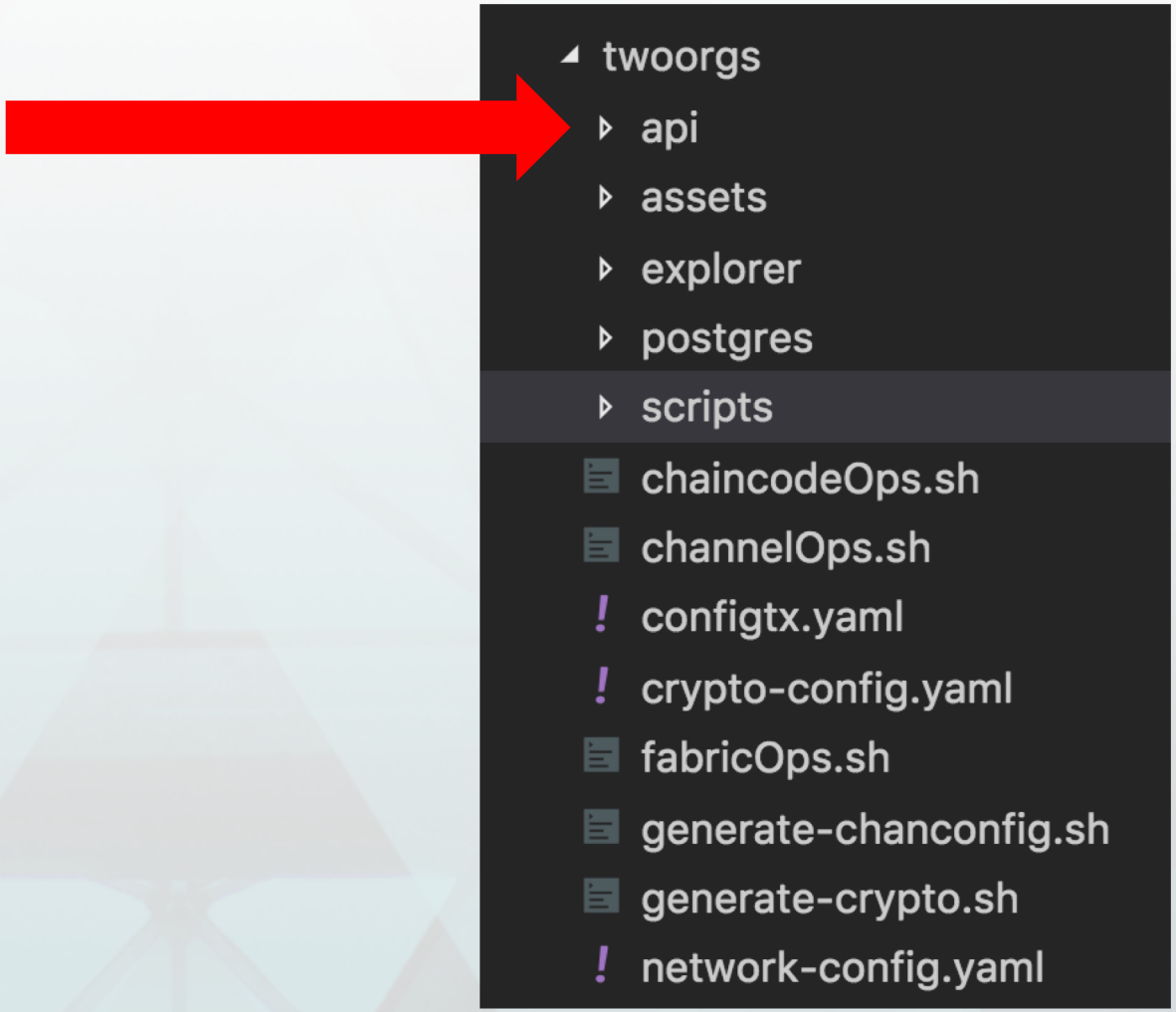
- If not specified at instantiation time, the endorsement policy defaults to “any member of the organizations in the channel”. For example, a channel with “Org1” and “Org2” would have a default endorsement policy of “OR(‘Org1.member’, ‘Org2.member’)”

(1) https://hyperledger-fabric.readthedocs.io/en/release-1.2/endorsement-policies.html

Fabric Node SDK

Chaincode and Application Development

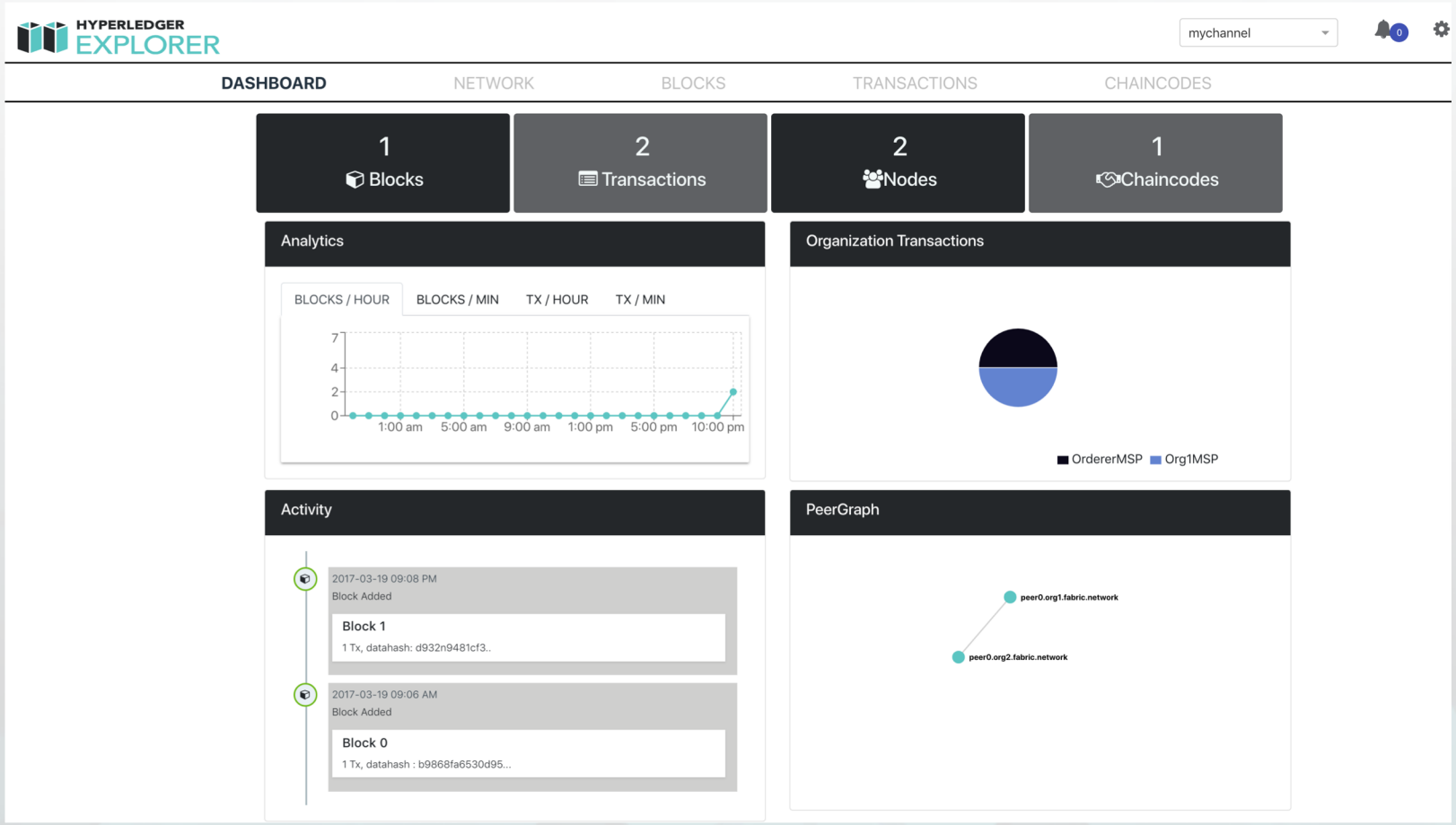
Fabric node SDK



Hyperledger Explorer

Chaincode and Application Development

Explorer in action



Fabric Development Kit – Roadmap

Chaincode and Application Development

Fabric Development Kit – Roadmap

- Add more reference implementations
- Fix the cli so it can generate skeleton code for end-to-end operations for local development
- Enable cli to deploy to any cloud for development
- Start a community to facilitate sharing of knowledge on multiple blockchain platforms – tentatively known as `Konsensia` - in a neutral forum

Q & A

Chaincode and Application Development