

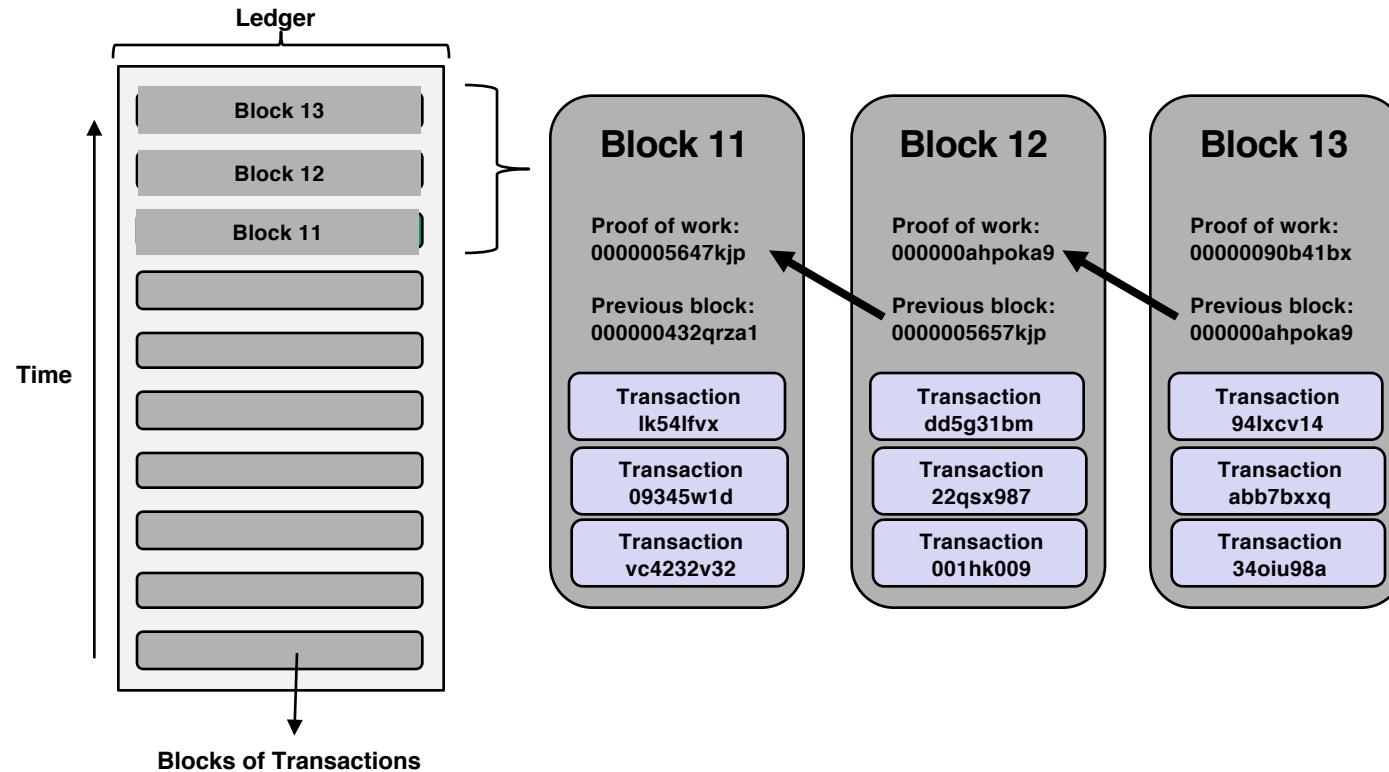
Blockchain Technology Details

Percival Lucena

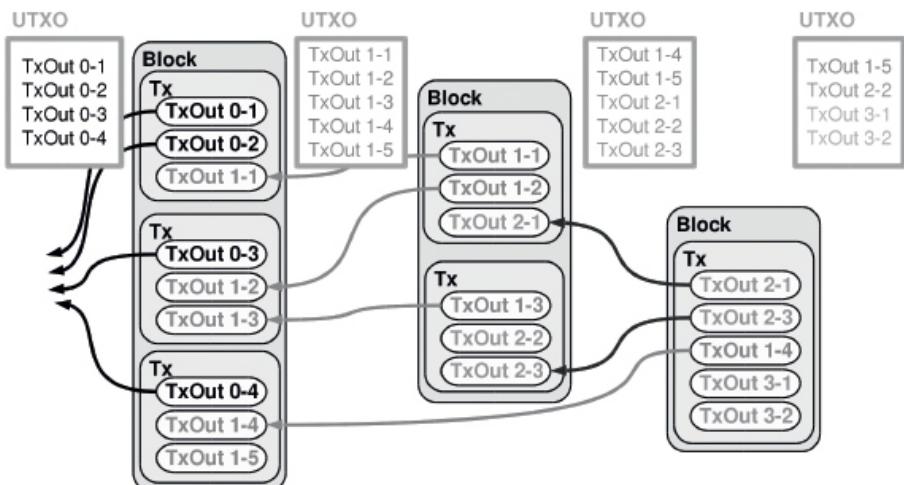
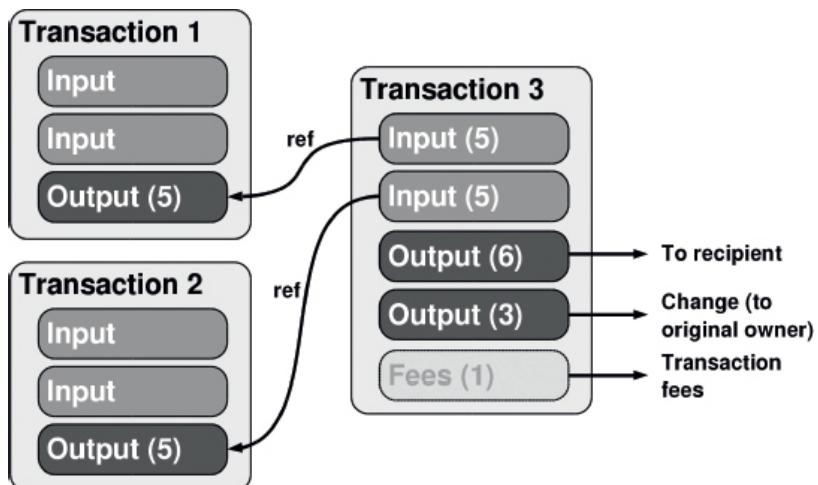
IBM Research

Introduction

Blocks of Transactions



Consensus

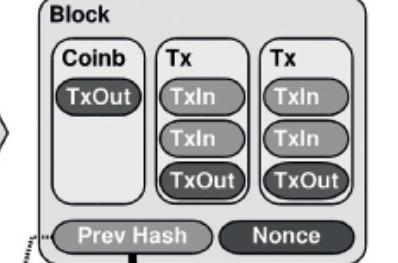


```

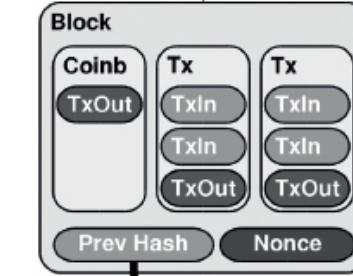
[Percivals-MacBook-Pro:~ percivalslucena$ shasum -a 256 jmeter.log
84e6c4cefd8739c1e7f2b8bb67b573092ba2928bf329b032fc776d4dbdf95005 jmeter.log
[Percivals-MacBook-Pro:~ percivalslucena$
[Percivals-MacBook-Pro:~ percivalslucena$ echo "obase=2; ibase=16; 84E6C4CEFD8739C1E7F2B8BB67B573092BA2928BF329B032FC776D4DBDF95005" | bc
100001100110011000100110011110011000001110011100000111100
011111100101110001001110110011101010111000100100101011011000100100101011111000111101
101000101001001001011111001100010011001100000011001011111000111101
011101101101001101111011111100101010000000101
[Percivals-MacBook-Pro:~ percivalslucena$]

```

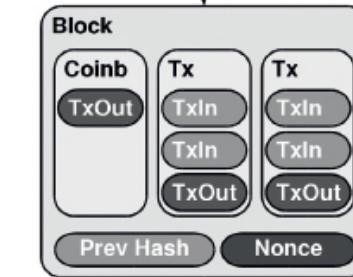
Block #271076 hash:
00000000000000006e11
63e8c741f39bb21c7b06
d2e06726b5630eea8b9
ce5584



~ 10 min



~ 10 min



Height

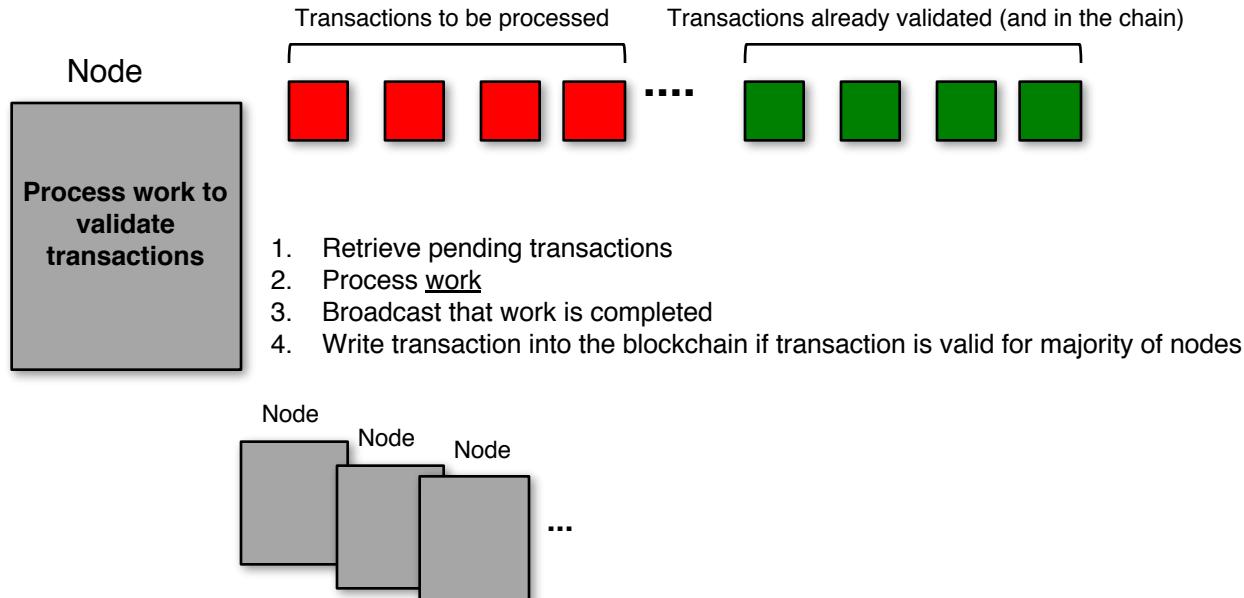
```

[Percivals-MacBook-Pro:~ percivalslucena$ cp jmeter.log teste1; echo "A" >> teste1
[Percivals-MacBook-Pro:~ percivalslucena$ shasum -a 256 teste1
78079dd9cea19c1c75f95247d780e92c5d9d9e541a02b61745759bf833a45c4a teste1
[Percivals-MacBook-Pro:~ percivalslucena$ echo "obase=2; ibase=16; 78079DD9CEA19C1C75F95247D780E92C5D9D9E541A02B61745759BF833A45C4A" | bc
11100000001110011100110011101010000110001110000011100011100
10111110010100100100111101011110000001110010010010110001011101101
0011101100111100101010000011010000001010110110000101110100010101110
10110011011111100000110011101001001011100010010101

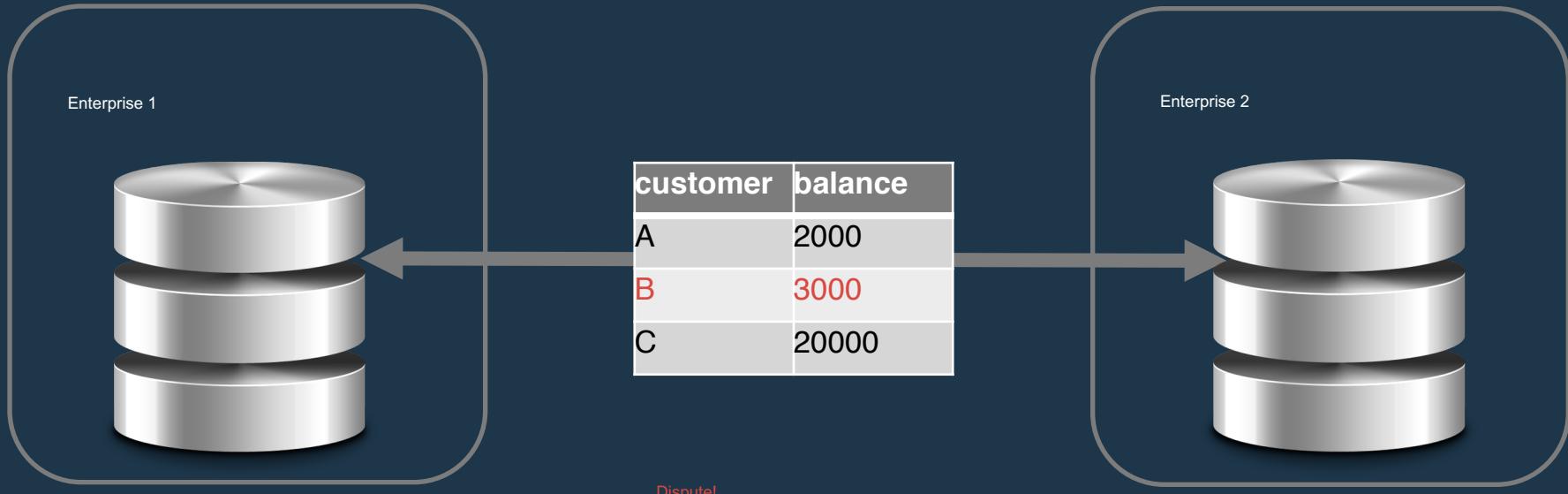
```

Consensus

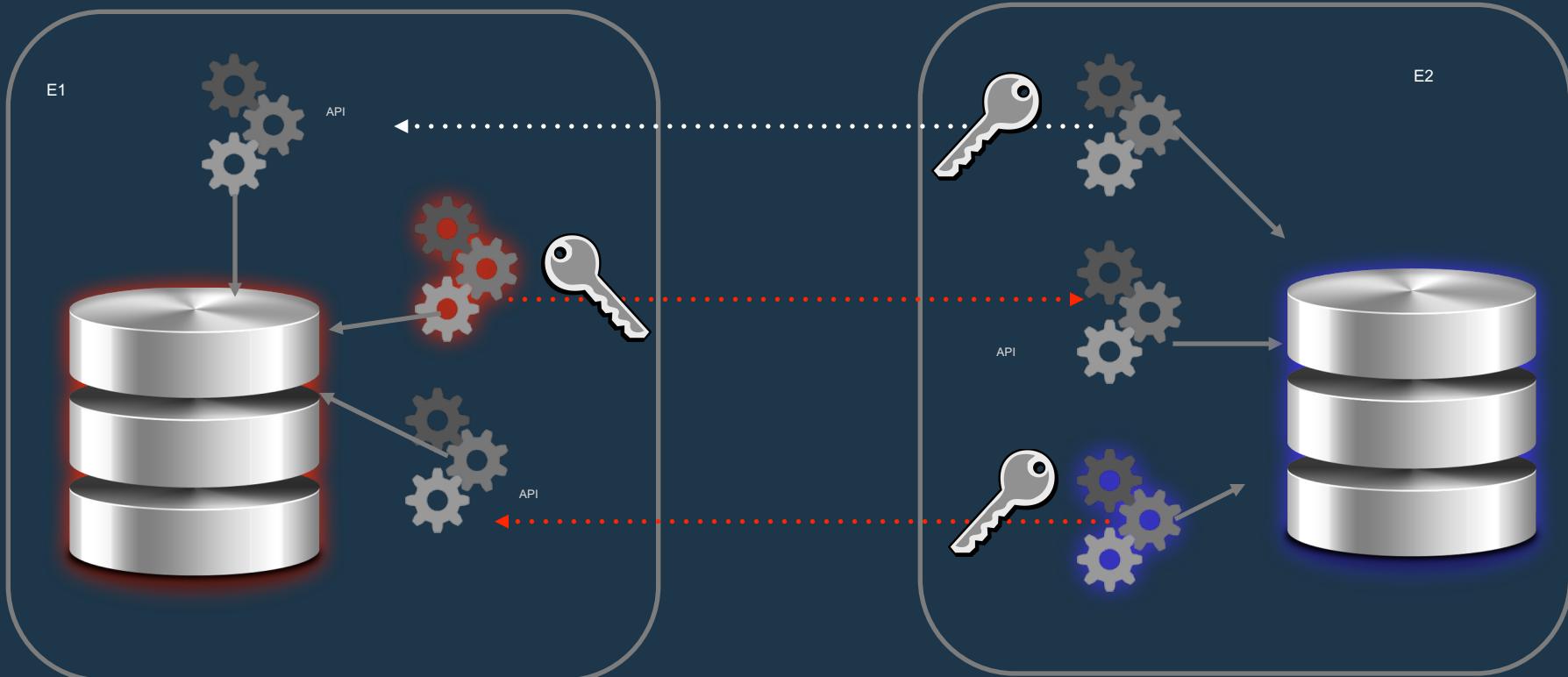
Proof of Work

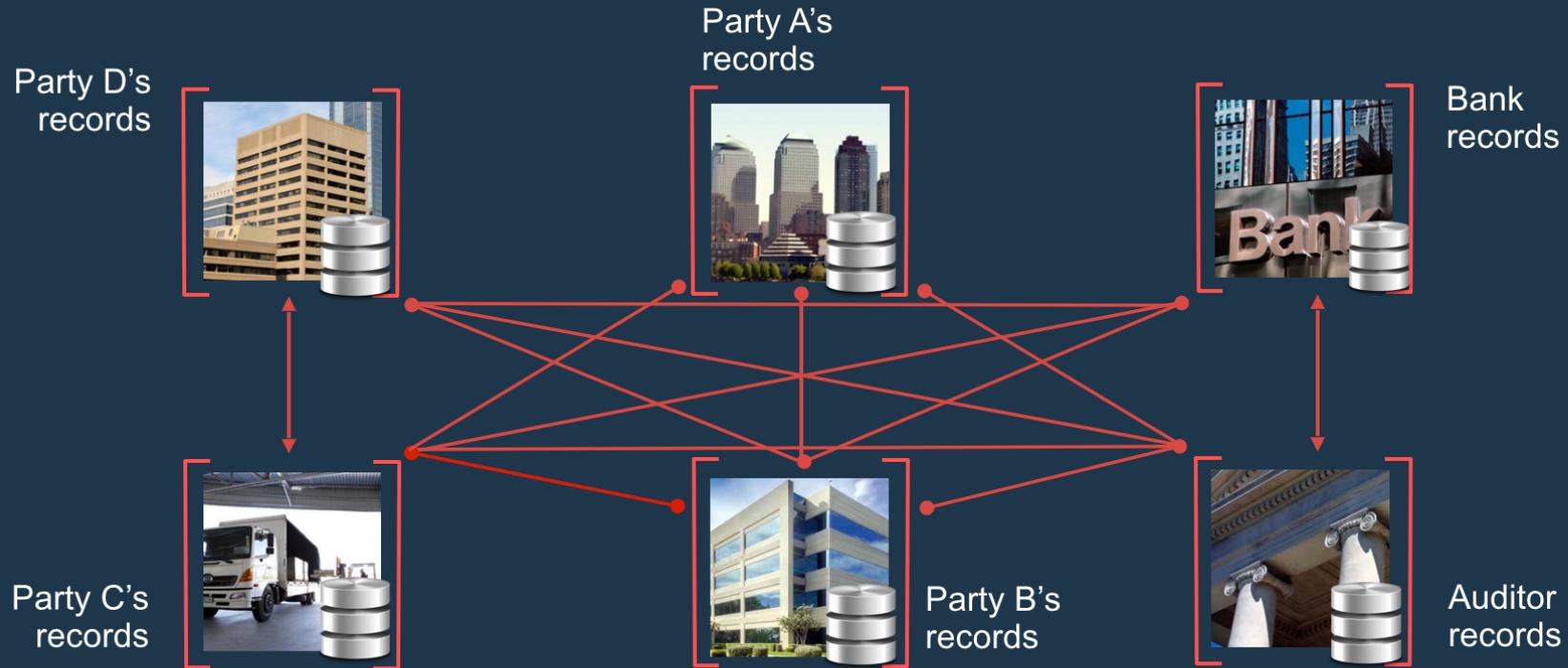


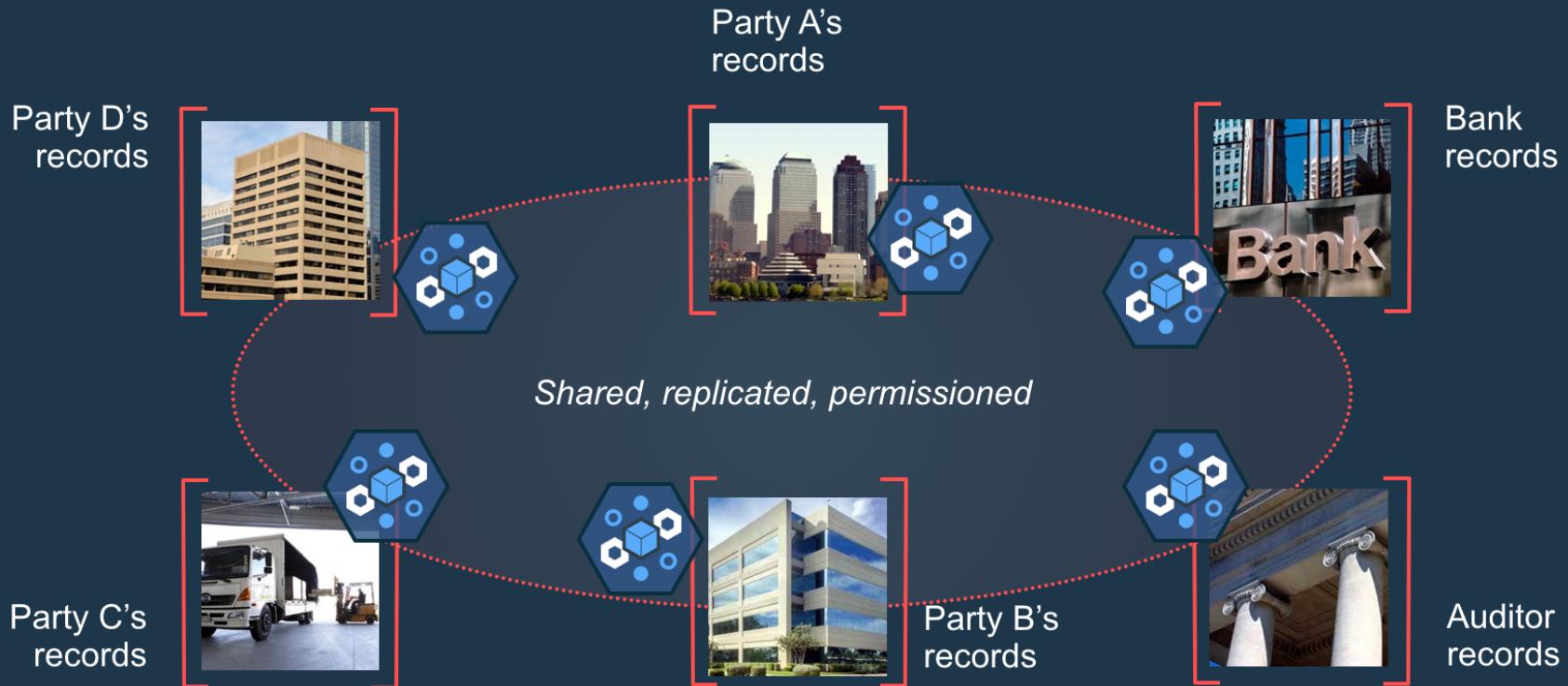
Distributed Databases



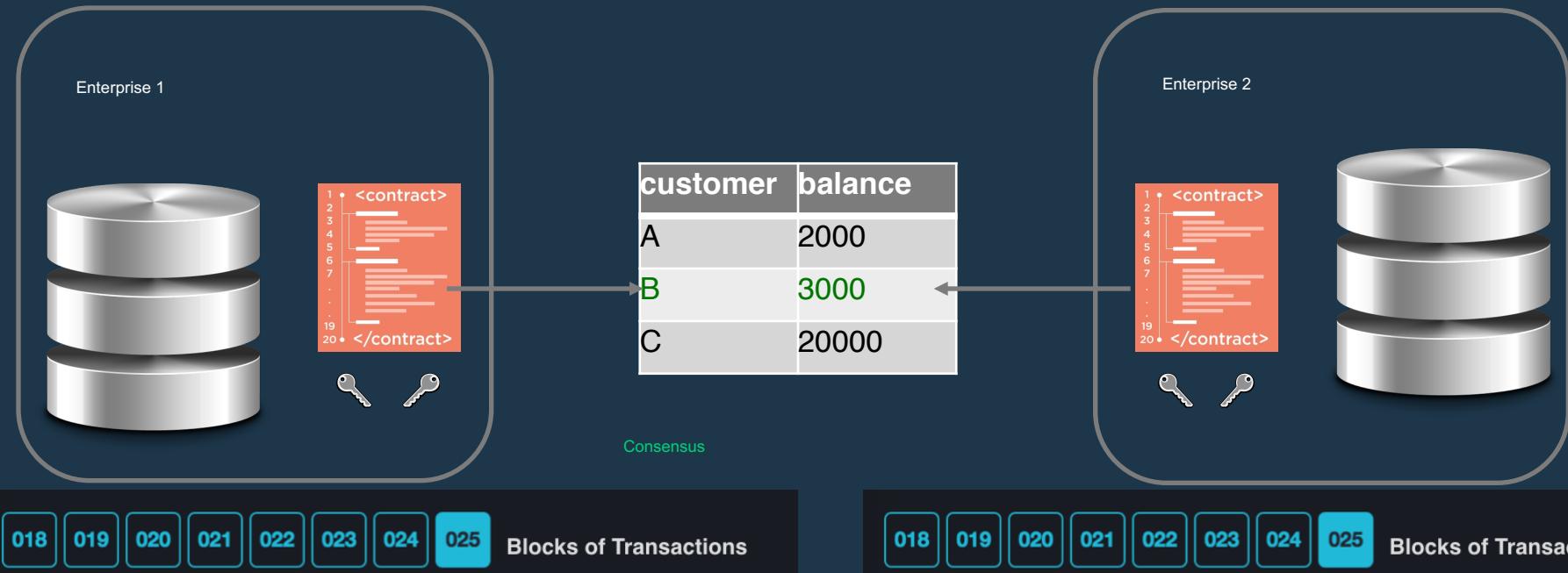
Enterprise Communication







IBM Hyperledger Blockchain:Smart Contracts



The Hyperledger Project

Open Source

Open Standard

Open Governance



The image shows the Hyperledger Project logo, which consists of a grid of logos for various members. The grid is organized into three main sections: PREMIER, GENERAL, and a bottom row. The PREMIER section includes logos for accenture, CME Group, DEUTSCHE BÖRSE GROUP, Digital Asset, DTCC, FUJITSU, HITACHI, IBM, Intel, J.P.Morgan, and R*. The GENERAL section includes logos for ABN AMRO, ANZ, BNY MELLON, Broadridge, Calastone, cisco, cloudsoft, CLS, coinplug, consensys, CREDITS, Cuscal, evue digital labs, Gem, guardtime, itBit, Milligan Partners, NEC, NTT DATA, redhat, Ribbit, Skry, SORAMITSU, STATE STREET, SWIFT, symbiont, tequacreek, THOMSON REUTERS, and VMware.



Brian_Behlendorf
ASF founder
Executive Director

5

PROJECTS

100+

CONTRIBUTORS

95

MEMBERS

6

HACKFESTS

Fabric

R3 corda

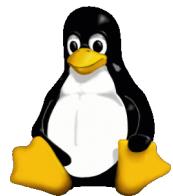
Sawtooth Lake

Hyperledger Offerings



HYPELEDGER
DOCKER

- [docker](#)



LINUX
INSTALL

<https://github.com/hyperledger/fabric>

BLUEMIX
SERVICE



- Managed Service on IBM Cloud
- Your private Blockchain network in 1-click
- Learn with sample applications
- Develop your own Smart Contracts

<https://console.ng.bluemix.net/catalog/services/blockchain/>



<http://hyperledger-fabric.readthedocs.io/en/latest/Setup/Chaincode-setup/#option-2-docker-for-mac-or-windows>

1. git clone <http://github.com/plucena/smartcontracts>
2. cd docker
3. docker pull hyperledger/fabric-peer:latest
4. docker pull hyperledger/fabric-membersvc:latest
5. docker-compose up



DOCKER INSTALL – 0.61 BRANCH

```
Percivals-MacBook-Pro:docker percivalslucena$ docker-compose up
Starting docker_membersrvc_1
Starting docker_vp0_1
Attaching to docker_membersrvc_1, docker_vp0_1
vp0_1    | 02:20:59.454 [logging] LoggingInit -> DEBU 001 Setting default logging level to DEBUG for command 'node'
vp0_1    | 02:20:59.454 [nodeCmd] serve -> INFO 002 Running in chaincode development mode
vp0_1    | 02:20:59.454 [nodeCmd] serve -> INFO 003 Set consensus to NOOPS and user starts chaincode
vp0_1    | 02:20:59.454 [nodeCmd] serve -> INFO 004 Disable loading validity system chaincode
vp0_1    | 02:20:59.454 [peer] func1 -> INFO 005 Auto detected peer address: 172.17.0.3:7051
vp0_1    | 02:20:59.456 [peer] func1 -> INFO 006 Auto detected peer address: 172.17.0.3:7051
vp0_1    | 02:20:59.457 [eventhub_producer] AddEventType -> DEBU 007 registering BLOCK
vp0_1    | 02:20:59.457 [eventhub_producer] AddEventType -> DEBU 008 registering CHAINCODE
vp0_1    | 02:20:59.457 [eventhub_producer] AddEventType -> DEBU 009 registering REJECTION
vp0_1    | 02:20:59.458 [eventhub_producer] AddEventType -> DEBU 00a registering REGISTER
vp0_1    | 02:20:59.458 [nodeCmd] serve -> INFO 00b Security enabled status: true
vp0_1    | 02:20:59.458 [nodeCmd] serve -> INFO 00d Privacy enabled status: false
vp0_1    | 02:20:59.458 [eventhub_producer] start -> INFO 00c event processor started
vp0_1    | 02:20:59.460 [db] open -> DEBU 00e Is db path [/var/hyperledger/production/db] empty [false]
vp0_1    | 02:20:59.675 [nodeCmd] func1 -> DEBU 00f Registering validator with enroll ID: test_vp0
vp0_1    | 02:20:59.675 [crypto] RegisterValidator -> INFO 010 Registering validator [test_vp0] with name [test_vp0]...
vp0_1    | 02:20:59.679 [crypto] Debugf -> DEBU 011 [validator.test_vp0] Data will be stored at [/var/hyperledger/production/crypto/validator/test_vp0]
vp0_1    | 02:20:59.682 [crypto] Debugf -> DEBU 012 [validator.test_vp0] Keystore path [/var/hyperledger/production/crypto/validator/test_vp0/ks] missing [false]: [<clear>]
```

IBM Bluemix Hyperledger Service

https://console.ng.bluemix.net/dashboard/apps/

Docs 265 Trial Days Remaining ▾ Percival Lucena's Account | US South : percival@unasp.net : dev

IBM Bluemix Apps Catalog Support Account

All Services (20) Create Service +

All Categories (1) >

Blockchain Filter

Infrastructure

- Compute
- Storage
- Network
- Security

Apps

- Boilerplates
- Cloud Foundry Apps
- Containers
- OpenWhisk

Services

Application Services

Deliver new web and mobile apps.

Blockchain

Utilize IBM's Blockchain Technology within Bluemix

IBM

IBM Bluemix Hyperledger Service

Docs

265 Trial Days Remaining ▾

Percival Lucena's Account | US South : percival@unasp.net : dev



IBM Bluemix Catalog

Catalog

Support

Account

[View all](#)

Blockchain

Blockchain is a peer-to-peer distributed ledger technology for a new generation of transactional applications that establishes trust, accountability and transparency while streamlining business processes. Think of it as an operating system for interactions, with the potential to vastly reduce the cost and complexity of getting things done. The distributed ledger makes it easier to create

Service name:

Blockchain-pv

Credential name:

Credentials-1

IBM Bluemix Hyperledger Service

Pricing Plans

Monthly prices shown are for country or region: [Brazil](#)

PLAN	FEATURES	PRICING
 Starter Developer plan (beta)	<ul style="list-style-type: none">- 4 peers and a Cert Authority- Deploy and test chaincode- Dashboard with logs, controls, and APIs- Sample apps with source code	Free
<p>Get started using IBM Blockchain! Monitor your network and view health status. Leverage the REST API to deploy and invoke chaincode transactions.</p>		
High Security Business Network plan	<ul style="list-style-type: none">- 4 peers and a Cert Authority on IBM LinuxOne™- All of the capabilities in Starter- Isolated environment on dedicated compute- Optimized performance and high speed network- Advanced Security: HSM, Secure Service Container and more	R\$41,206.80 BRL/MONTHLY

[Terms](#)

Need Help?

[Contact Bluemix Sales](#)

Estimate Monthly Cost

[Cost Calculator](#)

Create

IBM Bluemix Hyperledger Service

IBM Blockchain

Starter Network ID
b20d828038124c06b200c36d49738e7d [Copy](#) Refresh In 02:37/ 3:00 [Tips](#)

Peer	Routes	Discovery	Block Height	Status	Actions
Membership Services	gRPC Copy	-	-	Running	Stop Start
Validating Peer 0	HTTP Copy	4 / 4	9	Running	Stop Start
Validating Peer 1	HTTP Copy	4 / 4	9	Running	Stop Start
Validating Peer 2	HTTP Copy	4 / 4	9	Running	Stop Start
Validating Peer 3	HTTP Copy	4 / 4	9	Running	Stop Start

IBM Bluemix Hyperledger Service



IBM Blockchain

Network

Blockchain

Demo Chaincode

APIs

Logs

Service Status

Support

Starter Network ID

b20d828038124c06b200c36d49738e7d

Copy



Blockchain Overview

[Connected to Validating Peer 0]

7



BLOCKS

0.3



BLOCKS SPEED

1.0



TRANSACTION
ACTIVITY

1



DEPLOYMENTS

1



INVOCATIONS

Block Activity

Time	Block #	Deployments	Invocations	Date	Type	UUID	Chaincode ID	Payload
1 days ago	8	0	0			3aeb97		
1 days ago	7	0	0			93d679		
						68f966f		
						2b093c		

Chaincode

List Operations | Expand Operations

POST /chaincode

Implementation Notes

The `/chaincode` endpoint receives requests to deploy, invoke, and query a target chaincode. This service endpoint implements the JSON RPC 2.0 specification with the payload identifying the desired Chaincode operation within the 'method' field. Pick one body spec from the 3 below that matches your request type.

Response Class (Status 200)

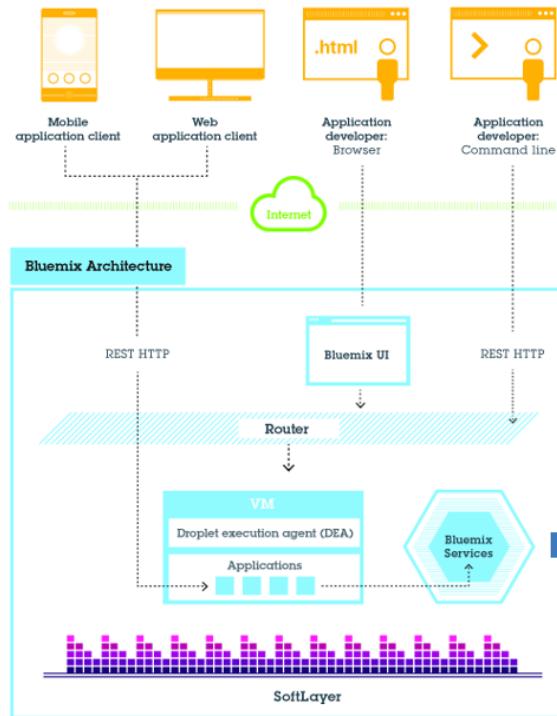
```
{  
  "jsonrpc": "2.0",  
  "result": {  
    "Status": "OK",  
    "Message": "500"  
  },  
  "id": 123  
}
```

Parameters

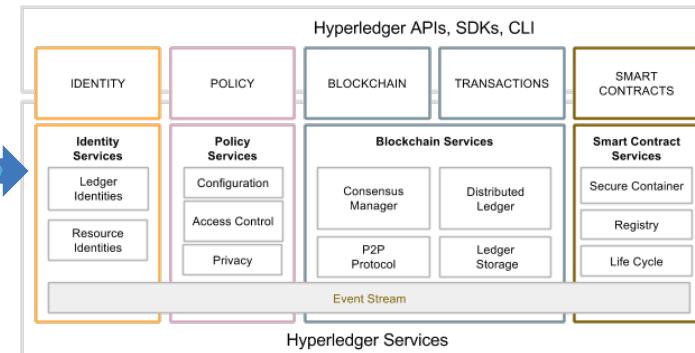
Parameter	Value	Description	Parameter Type	Data Type
QuerySpec	<input type="text"/>	body spec for query	body	<pre>{ "jsonrpc": "2.0", "method": "query", "params": { "type": 1, "chaincodeID": { "name": "mycc" } } }</pre>



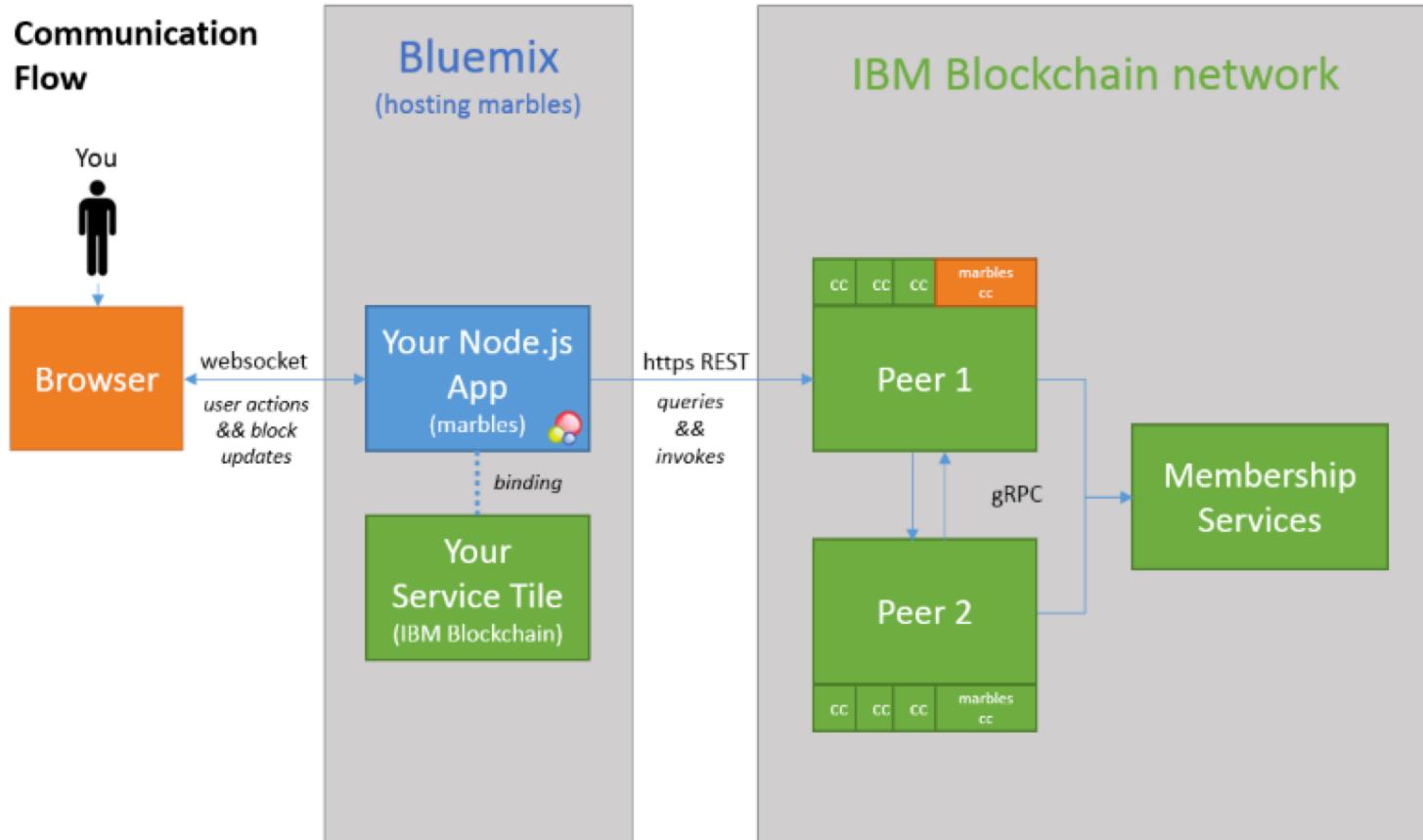
Architecture – Blockchain Platform



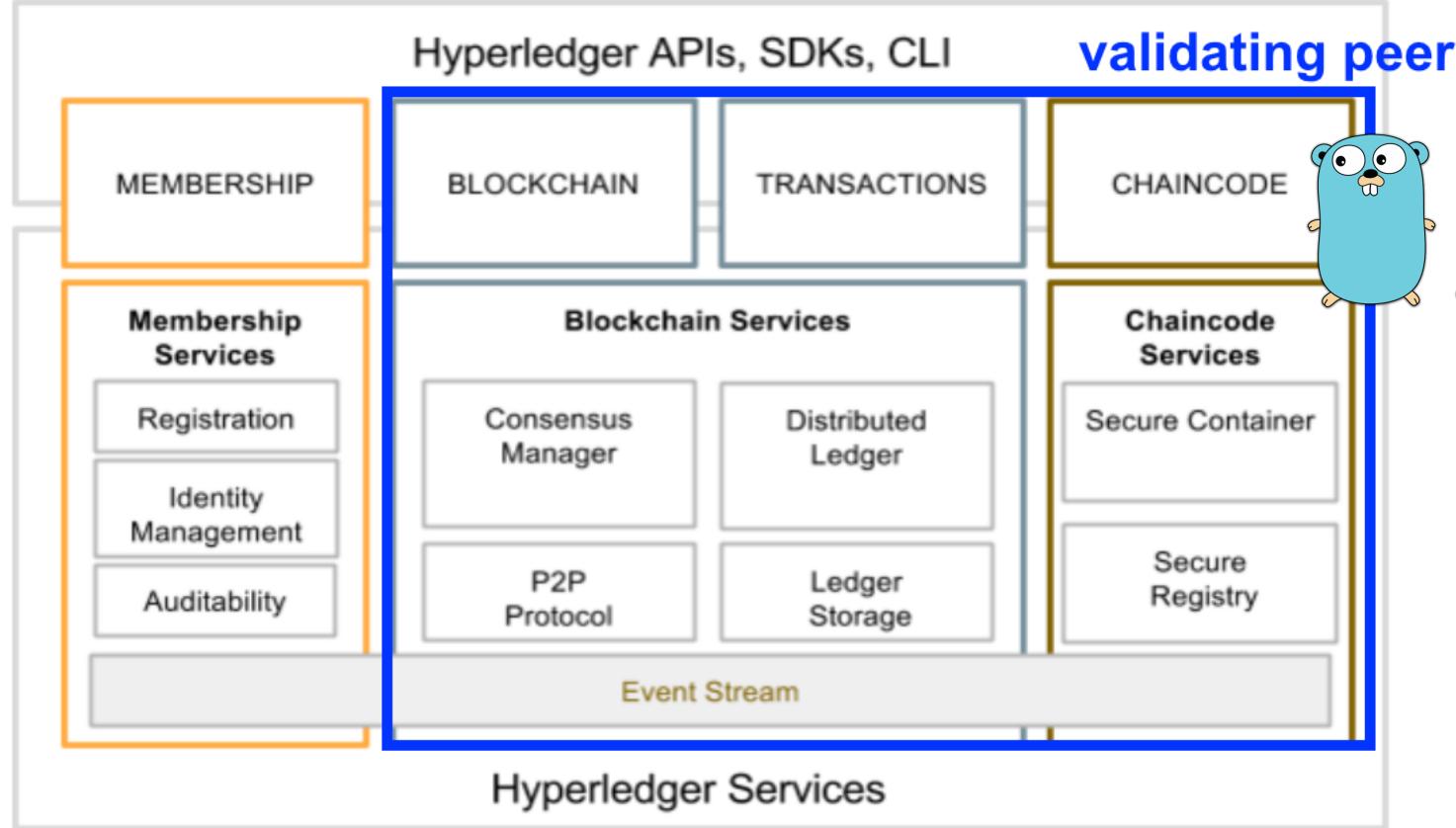
- Run your own application on any runtime available
- API based
- Enterprise Grade Security
- No need to worry about infrastructure and base ledger network
- Same code as publicly available
- IBM Blockchain also available on DockerHub
- IBM blockchain on Dedicated Cloud



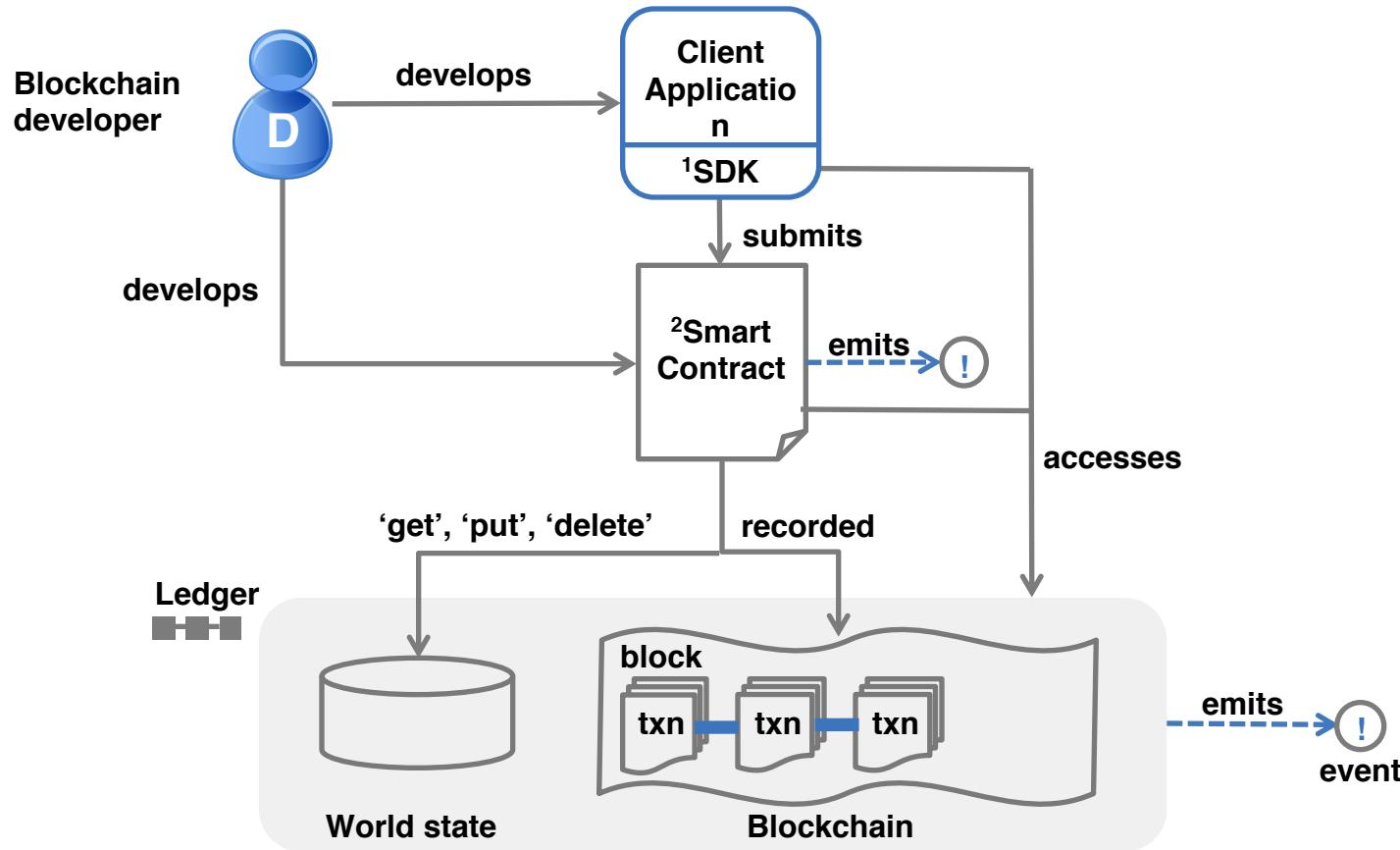
IBM Bluemix Hyperledger Service



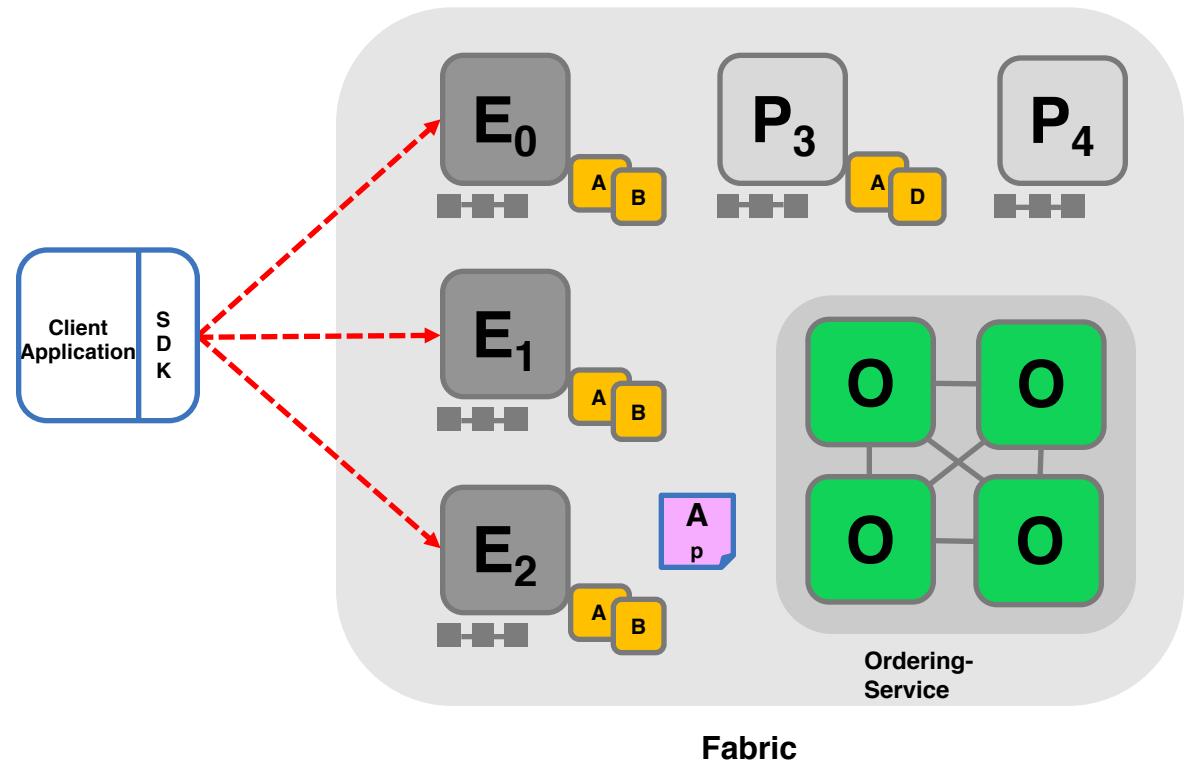
Hyperledger – Architecture 0.6



Hyperledger Blockchain – Single Node View



Smart Contract Execution



Application proposes transaction

Endorsement policy:

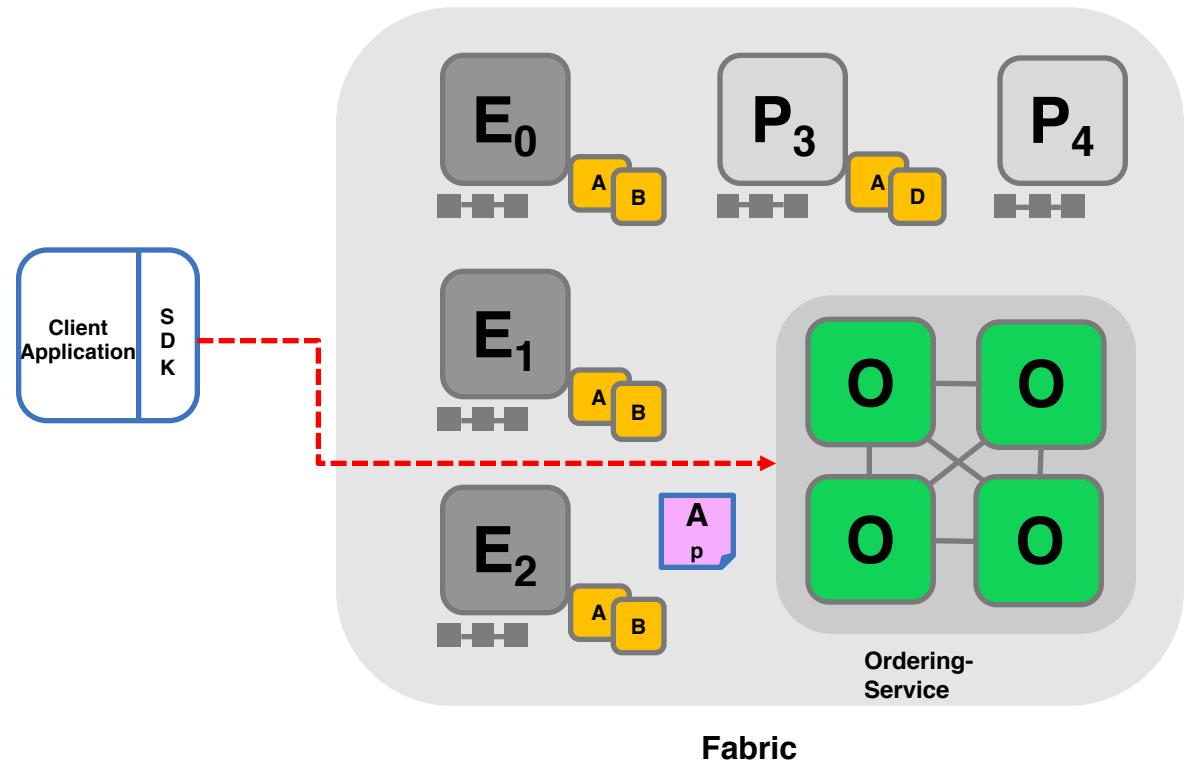
- " E_0 , E_1 and E_2 must sign"
- (P_3 , P_4 are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers $\{E_0, E_1, E_2\}$

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chain code)		Endorsement Policy

Smart Contract Execution



Application submits responses for ordering

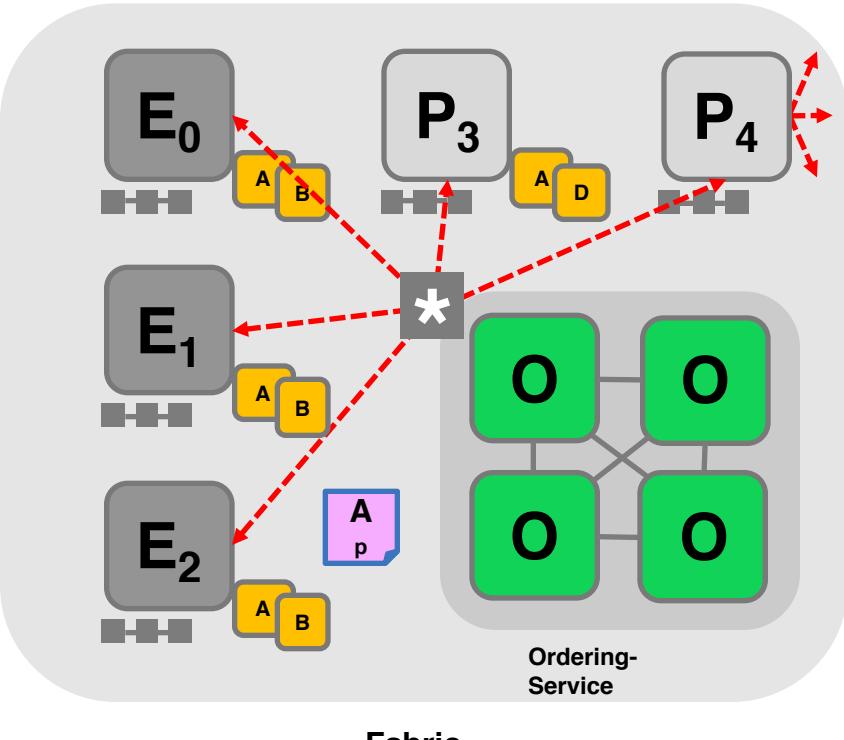
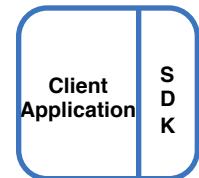
Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chain code)		Endorsement Policy

Smart Contract Execution



Orderer delivers to all committing peers

Ordering service collects transactions into proposed blocks for distribution to committing peers.
Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

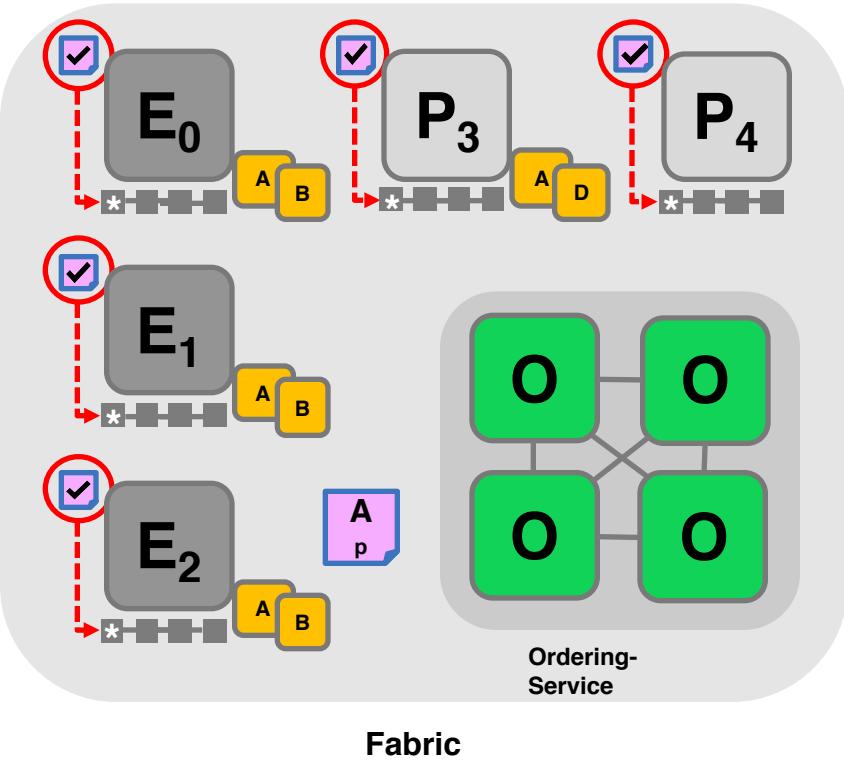
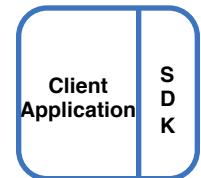
- SOLO (Single node, development)

Key: Kafka (Crash fault tolerance)

- SBFT (Byzantine fault tolerance)

Endorser		Ledger	
Committing Peer		Application	
Ordering Node			
Smart Contract (Chain code)		Endorsement Policy	

Smart Contract Execution



Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

Validated transactions are applied to the world state and retained on the ledger

Invalid transactions are also retained on the ledger but do not update world state

Endorser state	Ledger	Application
Committing Peer	Gray box	Gray box
Ordering Node	Green box	Gray box
Smart Contract (Chain code)	Yellow box	Pink box
Endorsement Policy	Blue dashed line	Gray box

Where do I start? —> SAMPLE APPS

GitHub, Inc. [US] | <https://github.com/IBM-Blockchain>

This organization Search Pull requests Issues Gist ToDo

IBM Blockchain <http://www.ibm.com/blockchain>

Repositories People 0

Search repositories... Type: All Language: All

car-lease-demo
A demonstration using IBM Blockchain to show how the lifecycle of vehicles can be recorded on a blockchain

JavaScript 27 stars 87 forks Updated 5 hours ago

marbles
IBM Blockchain - Marbles Demo

JavaScript 139 stars 210 forks Updated 7 hours ago

Top languages

JavaScript Go Shell

People 0 >

This organization has no public members. You must be a member to see who's a part of this organization.



Where do I start?

← → ⌂ GitHub, Inc. [US] https://github.com/plucena/blockchain-sdk/tree/master/test

This repository Search Pull requests Issues Gist ToDo

Unwatch 1 Star 1 Fork 0

Code Issues 0 Pull requests 0 Boards Reports Projects 0 Wiki

Branch: master **blockchain-sdk / test /** Create new file Upload files Find file History

plucena committed on GitHub Update swagger

Latest commit 5d9536b a minute ago

..

cli	Java examples	a month ago
swagger	Update swagger	a minute ago
testes.js	fix node.js	2 months ago



TESTS - GO SMART CONTRACTS - JAVA SMART CONTRACTS

HOW TO WRITE SMART CONTRACTS



```
func (t *SimpleChaincode) Init(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error)

func (t *SimpleChaincode) Invoke(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error)

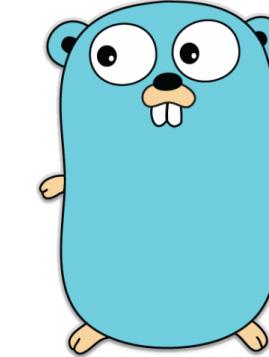
func (t *SimpleChaincode) Query(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error)
```

[**https://github.com/IBM-Blockchain/learn-chaincode**](https://github.com/IBM-Blockchain/learn-chaincode)

```
peer chaincode deploy -n mycc -c '{"Function":"init", "Args": ["a","100", "b", "200"]}'
```

[**https://obc-service-broker-prod.mybluemix.net/v2/swagger#!/Chaincode/chaincodeOp**](https://obc-service-broker-prod.mybluemix.net/v2/swagger#!/Chaincode/chaincodeOp)

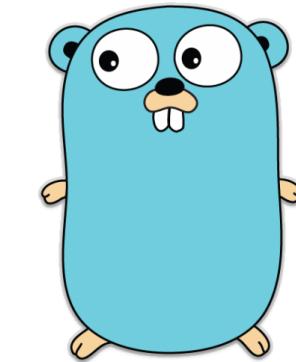
```
17 package main
18
19 import (
20     "errors"
21     "fmt"
22
23     "github.com/hyperledger/fabric/core/chaincode/shim"
24 )
25
26 // SimpleChaincode example simple Chaincode implementation
27 type SimpleChaincode struct {
28 }
29
30 func main() {
31     err := shim.Start(new(SimpleChaincode))
32     if err != nil {
33         fmt.Printf("Error starting Simple chaincode: %s", err)
34     }
35 }
36
37 // Init resets all the things
38 func (t *SimpleChaincode) Init(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
39     if len(args) != 1 {
40         return nil, errors.New("Incorrect number of arguments. Expecting 1")
41     }
42
43     err := stub.PutState("hello_world", []byte(args[0]))
44     if err != nil {
45         return nil, err
46     }
47
48     return nil, nil
49 }
```



```
51 // Invoke is our entry point to invoke a chaincode function
52 func (t *SimpleChaincode) Invoke(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
53     fmt.Println("invoke is running " + function)
54
55     // Handle different functions
56     if function == "init" {
57         return t.Init(stub, "init", args)
58     } else if function == "write" {
59         return t.write(stub, args)
60     }
61     fmt.Println("invoke did not find func: " + function)
62
63     return nil, errors.New("Received unknown function invocation")
64 }
65
66 // Query is our entry point for queries
67 func (t *SimpleChaincode) Query(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
68     fmt.Println("query is running " + function)
69
70     // Handle different functions
71     if function == "read" { //read a variable
72         return t.read(stub, args)
73     }
74     fmt.Println("query did not find func: " + function)
75
76     return nil, errors.New("Received unknown function query")
77 }
78
```



```
79 // write - invoke function to write key/value pair
80 func (t *SimpleChaincode) write(stub *shim.ChaincodeStub, args []string) ([]byte, error) {
81     var key, value string
82     var err error
83     fmt.Println("running write()")
84
85     if len(args) != 2 {
86         return nil, errors.New("Incorrect number of arguments. Expecting 2. name of the key and value to set")
87     }
88
89     key = args[0] //rename for funsies
90     value = args[1]
91     err = stub.PutState(key, []byte(value)) //write the variable into the chaincode state
92     if err != nil {
93         return nil, err
94     }
95     return nil, nil
96 }
97
98 // read - query function to read key/value pair
99 func (t *SimpleChaincode) read(stub *shim.ChaincodeStub, args []string) ([]byte, error) {
100    var key, jsonResp string
101    var err error
102
103    if len(args) != 1 {
104        return nil, errors.New("Incorrect number of arguments. Expecting name of the key to query")
105    }
106
107    key = args[0]
108    valAsbytes, err := stub.GetState(key)
109    if err != nil {
110        jsonResp = "{\"Error\":\"Failed to get state for " + key + "\"}"
111        return nil, errors.New(jsonResp)
112    }
113
114    return valAsbytes, nil
115 }
```





LINKS

GRPC - npm install hfc —save

<https://github.com/hyperledger/fabric/tree/master/sdk/node>

IBM SAMPLE APPS - <https://github.com/IBM-Blockchain>

DeveloperWorks Tutorial - <http://ibm.co/2bOZoMh> -

Java - <https://github.com/xspeedcruiser/java-chaincode-tutorial>