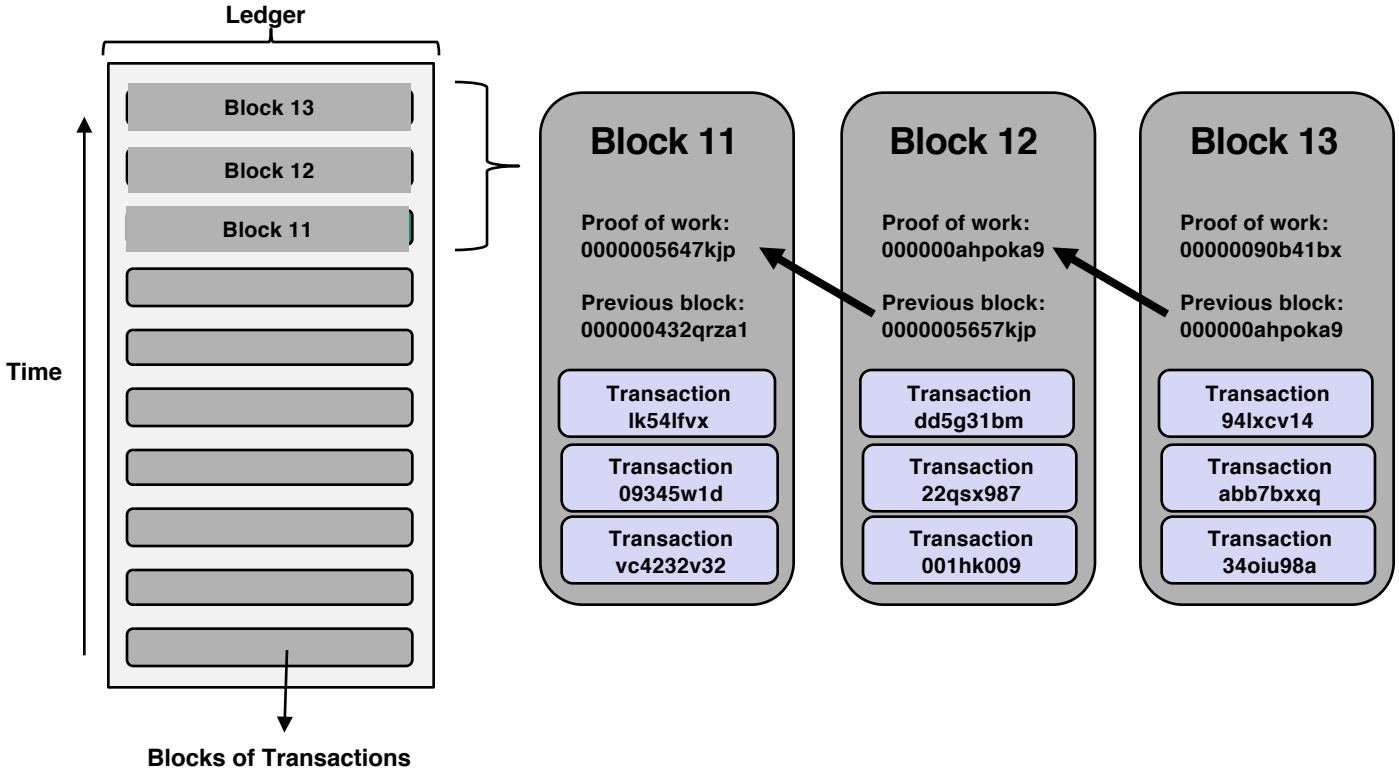
Blockchain Technology Details

Percival Lucena

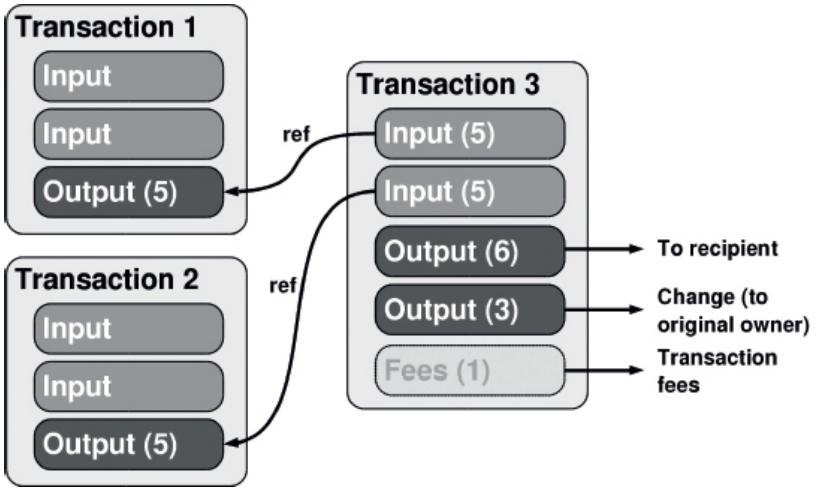
IBM Research

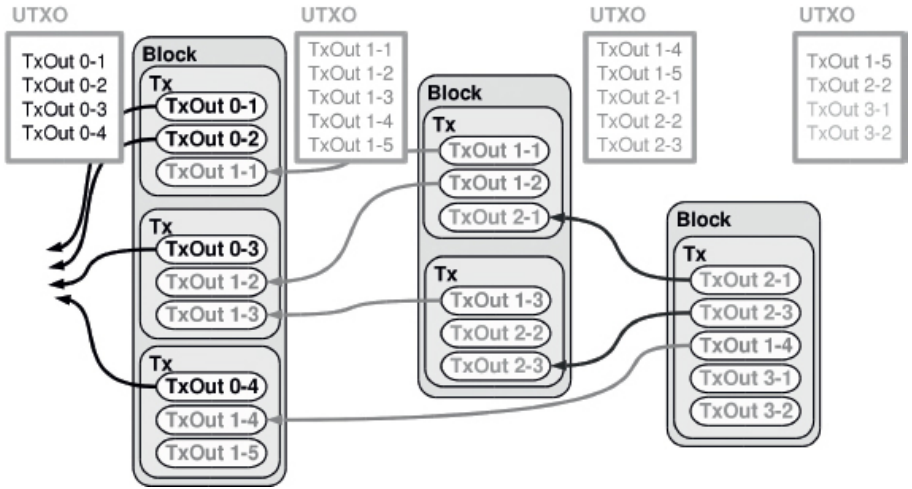
Introduction

Blocks of Transactions



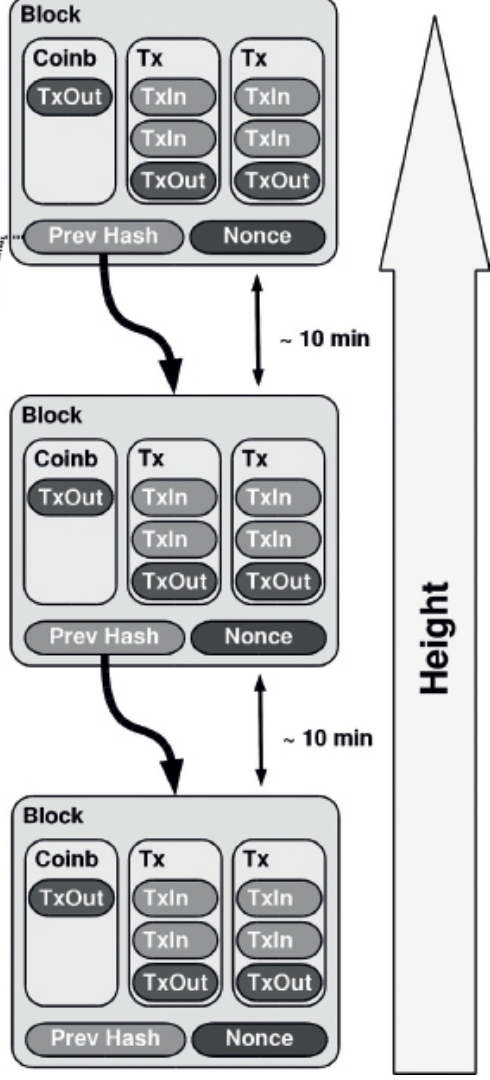
Consensus





```
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
10000100111001101100010011001110111111101100001110011100111000001110\
01111111001010111000101110111011001111011010101110011000010010010101 1\
10100010100100101000101111110011001010011011000000110010111111000111\
01110110110101001101101111011111100101010000000101
```

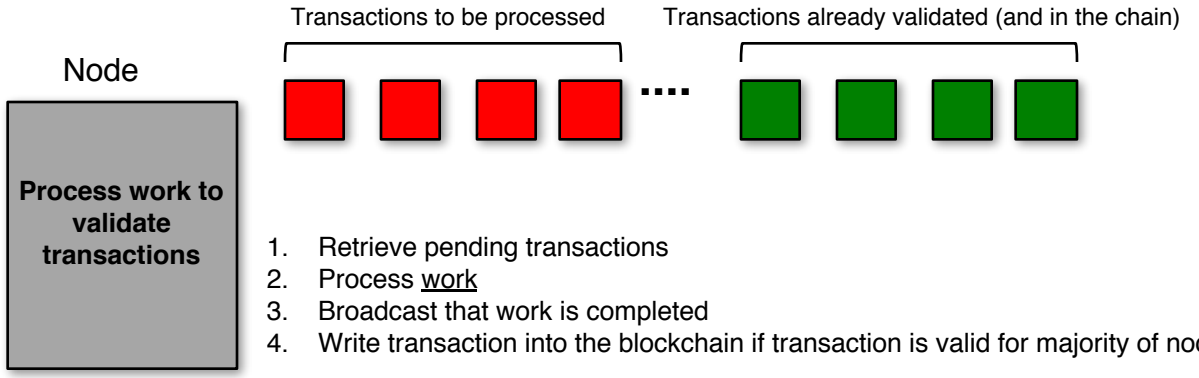
Block #271076 hash:
000000000000000 6e11
63e8c741f39bb21c7b06
d2e06726b5630eea8b9
ce5584



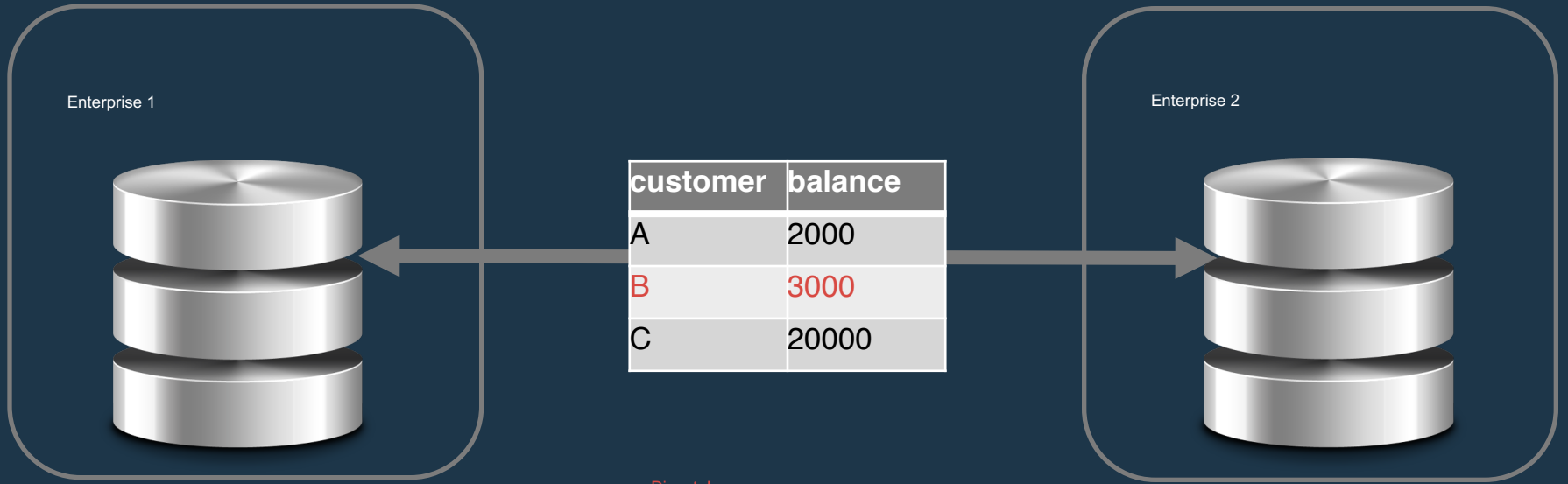

```
Percivals-MacBook-Pro:~ percivalslucena$ cp jmeter.log teste1; echo "A" >> teste1
Percivals-MacBook-Pro:~ percivalslucena$ shasum -a 256 teste1
78079dd9cea19c1c75f95247d780e92c5d9d9e541a02b61745759bf833a45c4a  teste1
Percivals-MacBook-Pro:~ percivalslucena$ echo "obase=2; ibase=16; 78079DD9CEA19C1C75F95247D780E92C5D9D9E541A02B61745759BF833A45C4A" | bc
11110000000011110011101110110011100111010100001100111000001110001110\
10111111001010100100100011111010111100000011101001001011000101110 11\
00111011001111001010100000110100000001010110110000101110100010101110\
10110011011111110000011001110100100010111001001010
```

Consensus

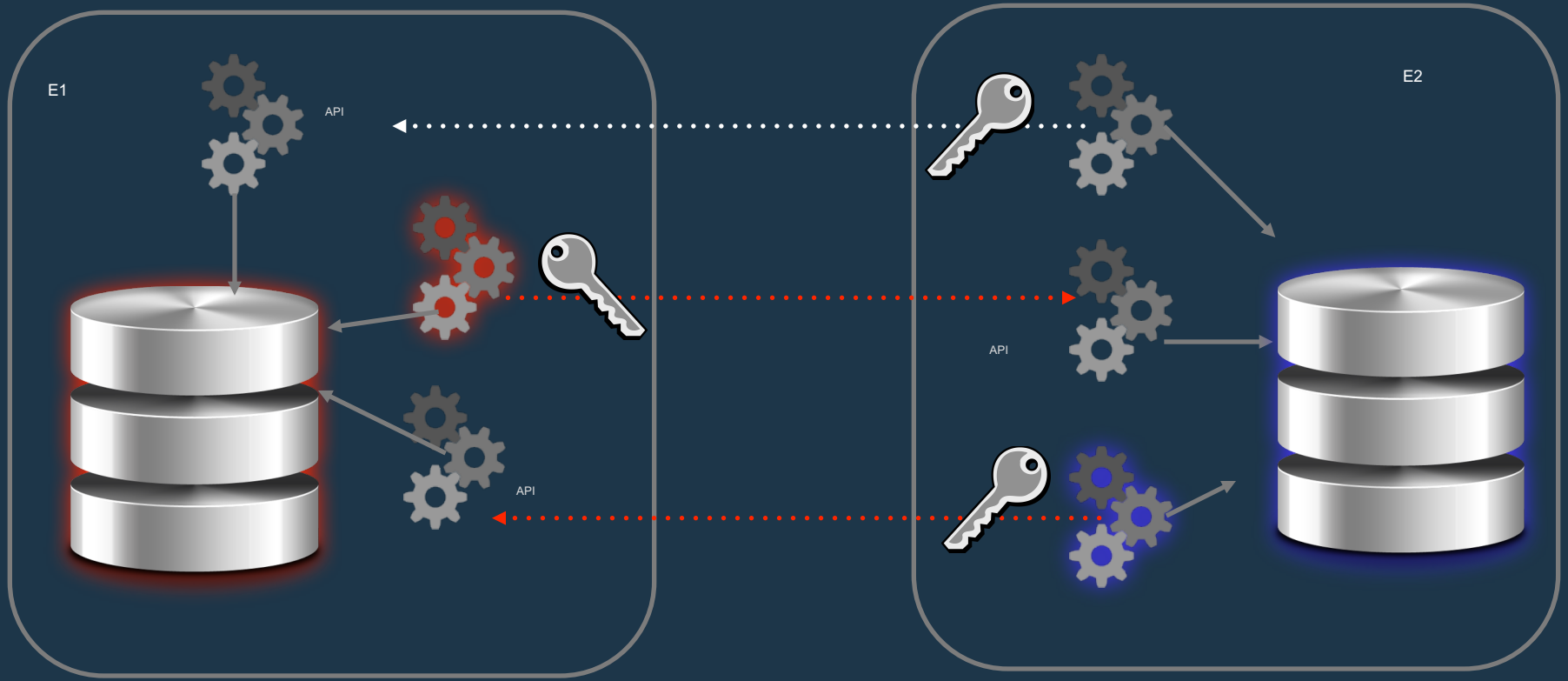
Proof of Work



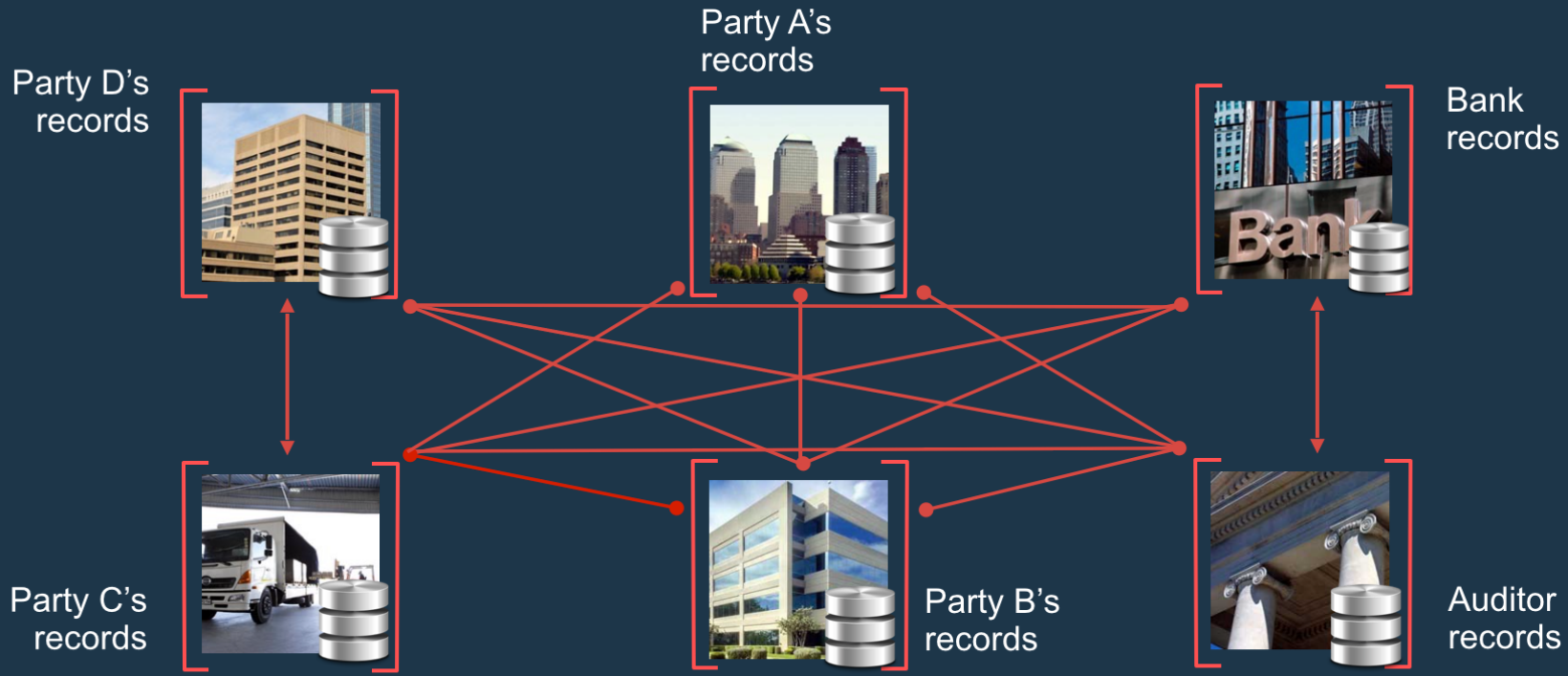
Distributed Databases



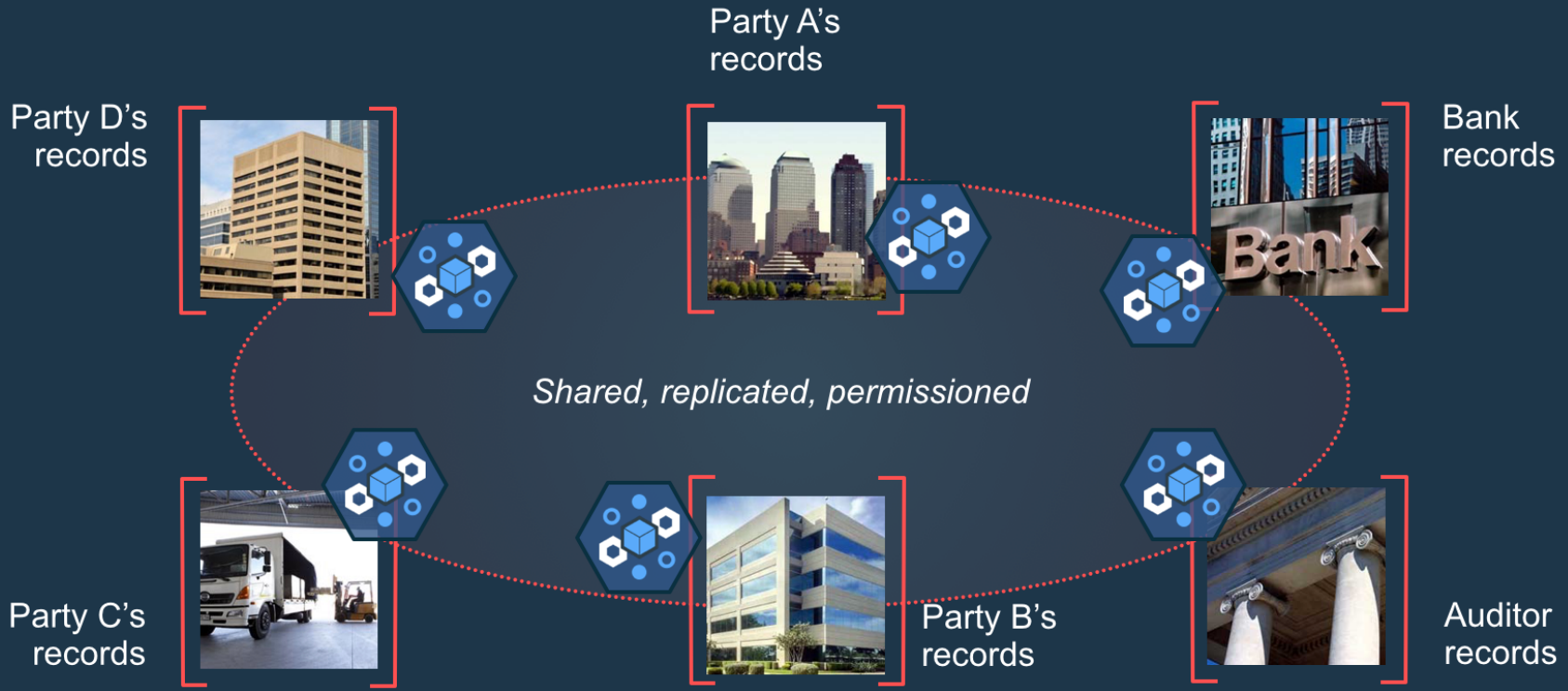
Enterprise Communication



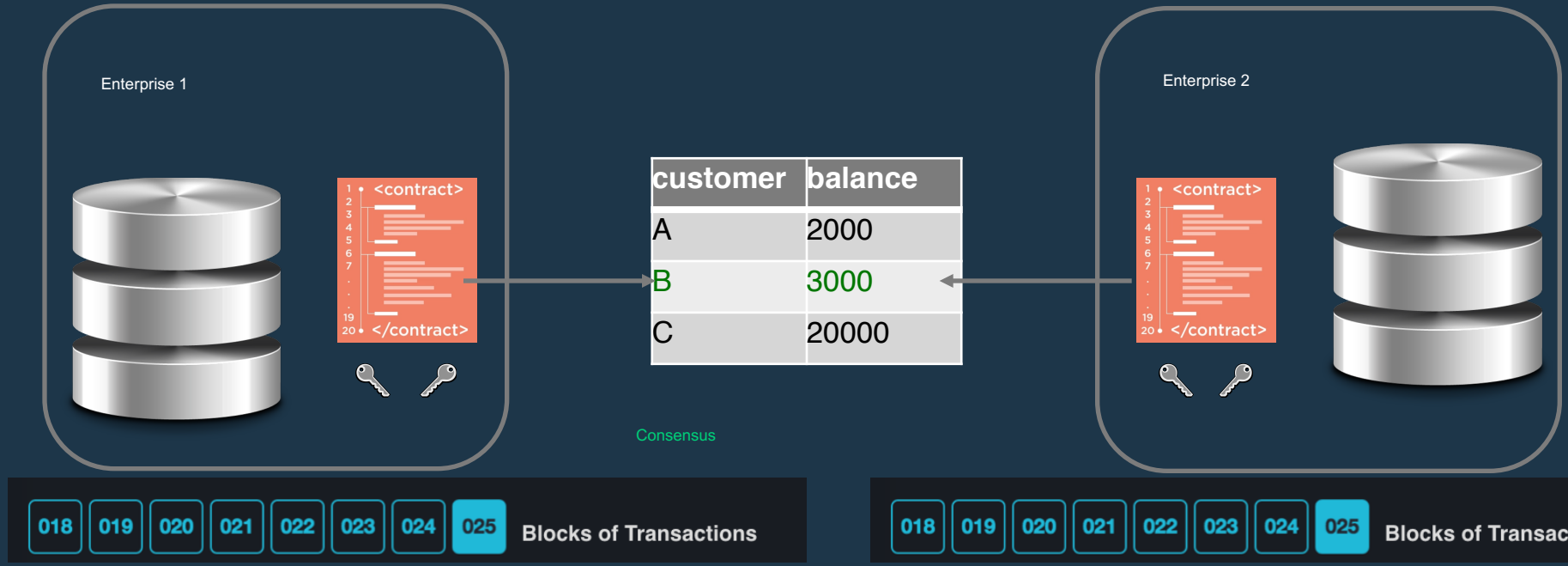
Reconciliation!



IBM Hyperledger Blockchain



IBM Hyperledger Blockchain:Smart Contracts



The Hyperledger Project

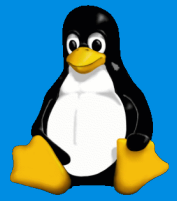
Open Source Open Standard Open Governance

PREMIER

accenture · CME Group · Deutsche Börse Group · Digital Asset · DTCC · Fujitsu · Hitachi Inspire the Next · IBM · intel · J.P.Morgan · R3

GENERAL

ABN AMRO · ANZ · BLOCKCHAIN · Blockstream · bloq · BNY Mellon · Broadridge · Calastone · Cisco · cloudsoft · CLS · coinplug · Consensys · CREDITS · Cuscal · eVUE · guardtime · itBit · Milligan Partners · NEC · NTT DATA · redhat · Ribbit · Skry · Soramitsu · State Street · SWIFT · symbiont · tequa creek · Thomson Reuters · vmware



Brian_Behlendorf
ASF founder
Executive Director

Fabric

R3 corda

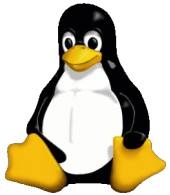
Sawtooth Lake

5	100+	95	6
PROJECTS	CONTRIBUTORS	MEMBERS	HACKFESTS

Hyperledger Offerings

HYPELEDGER DOCKER

- docker



LINUX INSTALL

https://github.com/hyperledger/fabric

BLUEMIX SERVICE

Blockchain

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

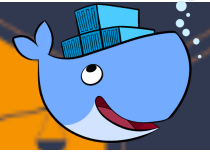
https://console.ng.bluemix.net/catalog/services/blockchain/

Docker

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

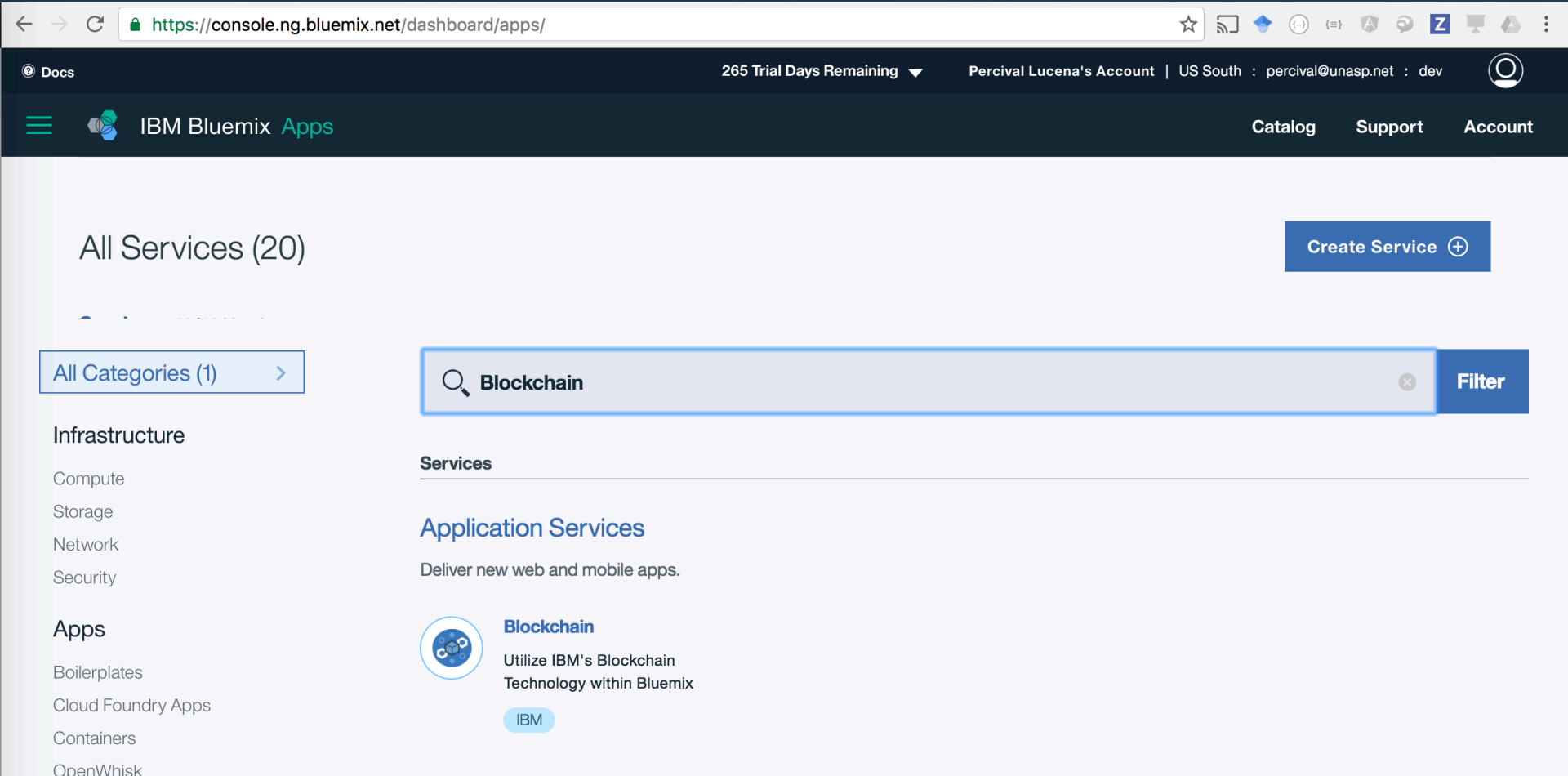
1. git clone http://github.com/plucena/smartcontracts
2. cd docker
3. docker pull hyperledger/fabric-peer:latest
4. docker pull hyperledger/fabric-membersrvc:latest
5. docker-compose up

DOCKER INSTALL – 0.61 BRANCH

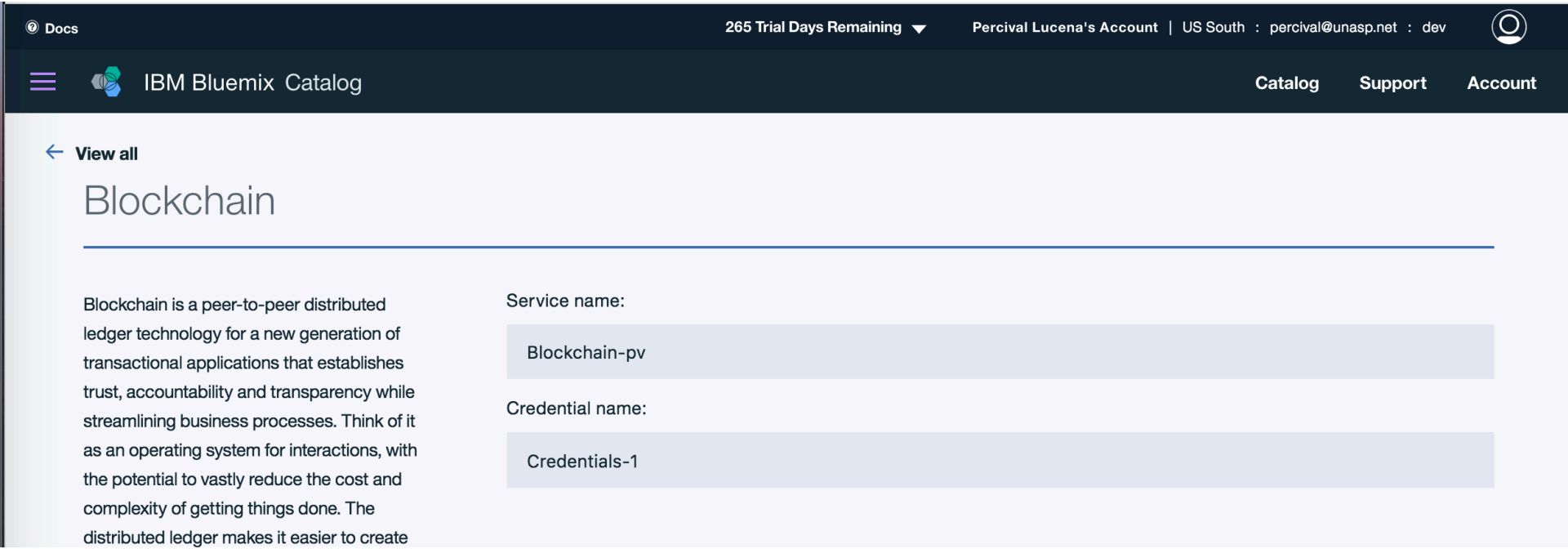


```
Percivals-MacBook-Pro:docker percivalslucena$ docker-compose up
Starting docker_membersrvc_1
Starting docker_vp0_1
Attaching to docker_membersrvc_1, docker_vp0_1
vp0_1         | 02:20:59.454 [logging] LoggingInit -> DEBU 001 Setting default logging level to DEBUG for command 'node'
vp0_1         | 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]: [<clea
>]
```

IBM **Bluemix** Hyperledger Service



IBM **Bluemix** Hyperledger Service



IBM **Bluemix** Hyperledger Service

Pricing Plans

Monthly prices shown are for country or region: Brazil

	PLAN	FEATURES	PRICING
✓	**Starter Developer plan (beta)**	**- 4 peers and a Cert Authority** - Deploy and test chaincode - Dashboard with logs, controls, and APIs - Sample apps with source code	Free
		Get started using IBM Blockchain! Monitor your network and view health status. Leverage the REST API to deploy and invoke chaincode transactions.	
	High Security Business Network plan	**- 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

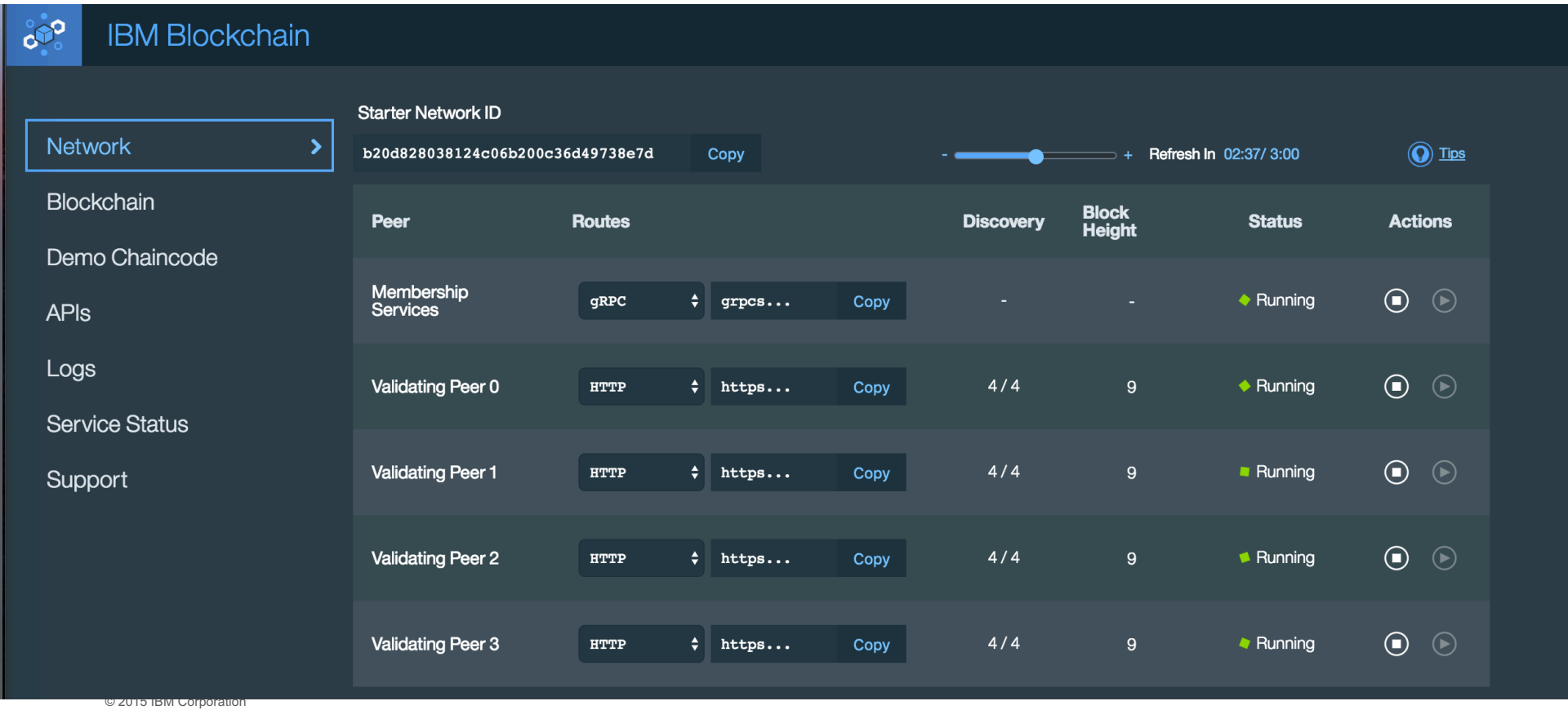
Terms

Need Help?
Contact Bluemix Sales

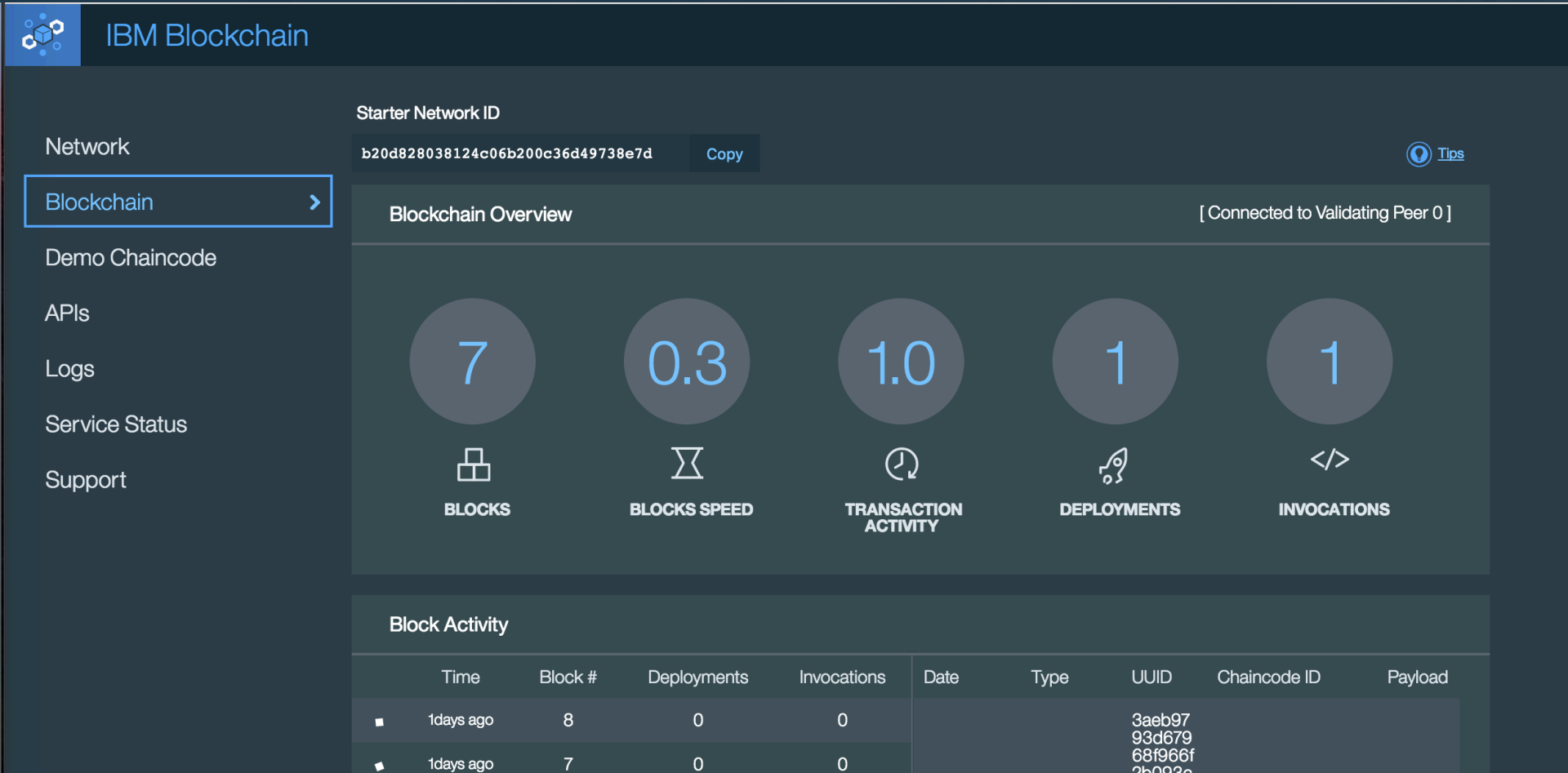
Estimate Monthly Cost
Cost Calculator

Create

IBM **Bluemix** Hyperledger Service



IBM **Bluemix** Hyperledger Service



Blockchain

List Operations | Expand Operations

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)

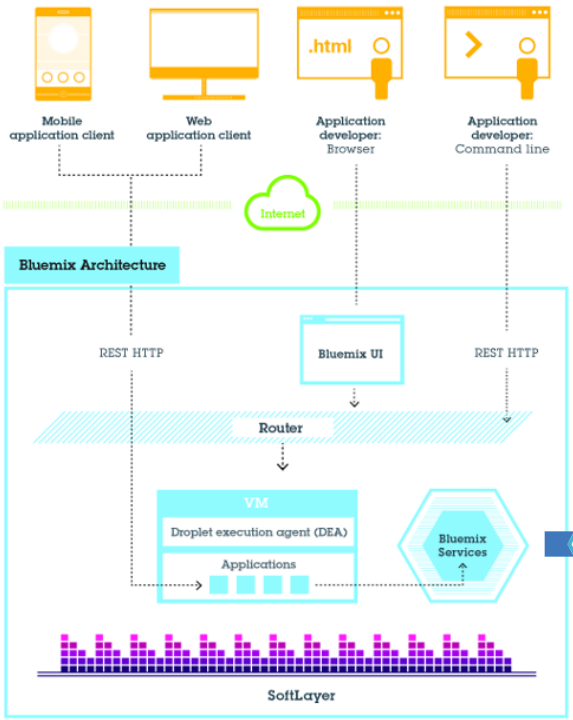
```
{
  "jsonrpc": "2.0",
  "result": {
    "Status": "OK",
    "Message": "500"
  },
  "id": 123
}
```

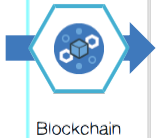
Parameters

Parameter	Value	Description	Parameter Type	Data Type
QuerySpec		*body spec for query*	body	{ "jsonrpc": "2.0", "method": "query", "params": { "type": 1, "chaincodeID": {

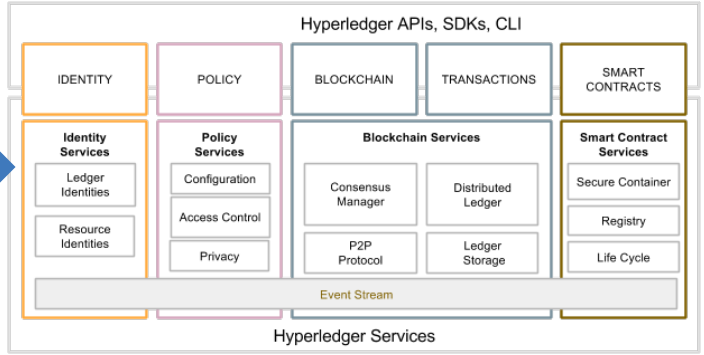
Architecture – Blockchain Platform

Technology Leadership Council Brazil

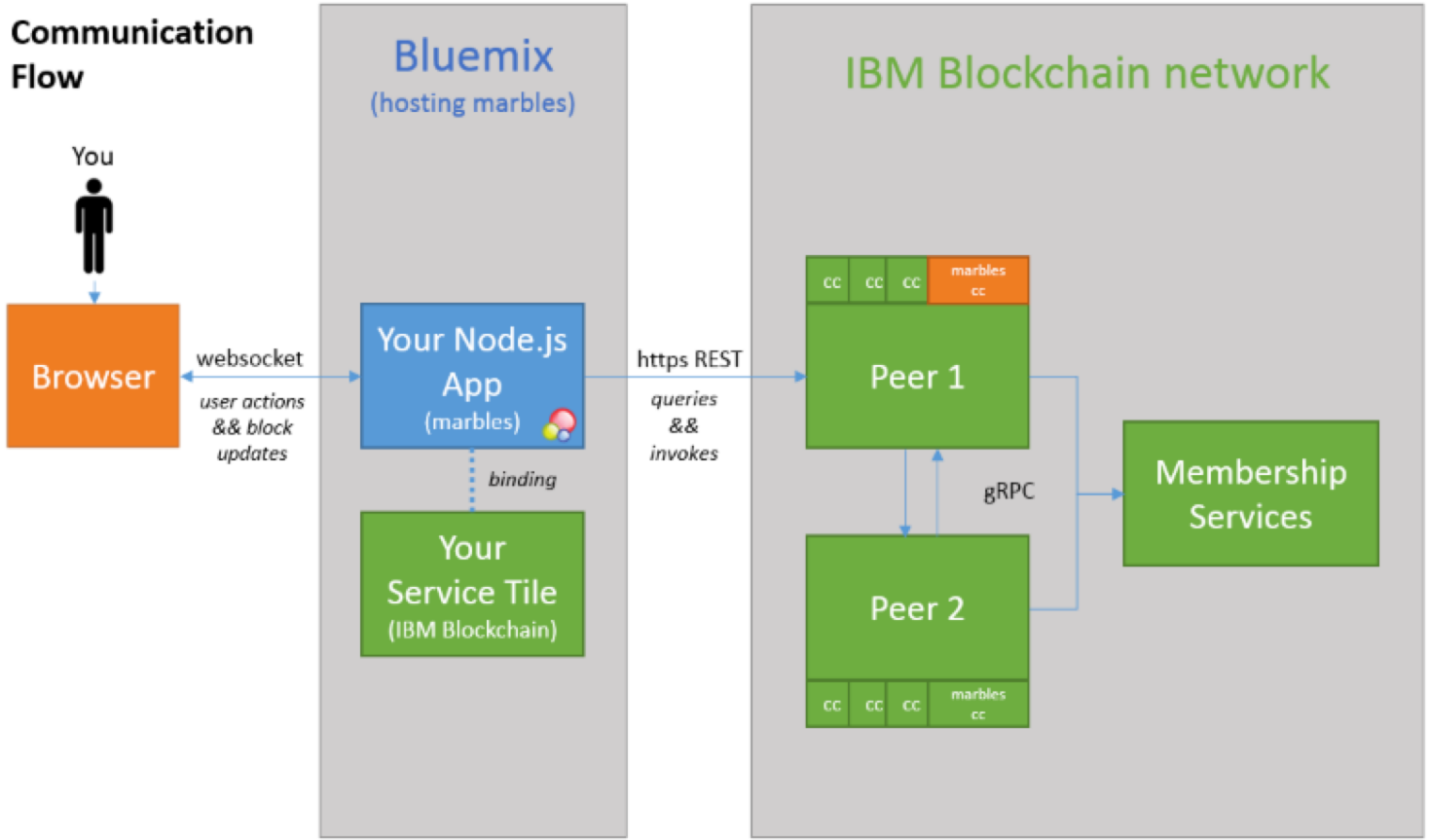




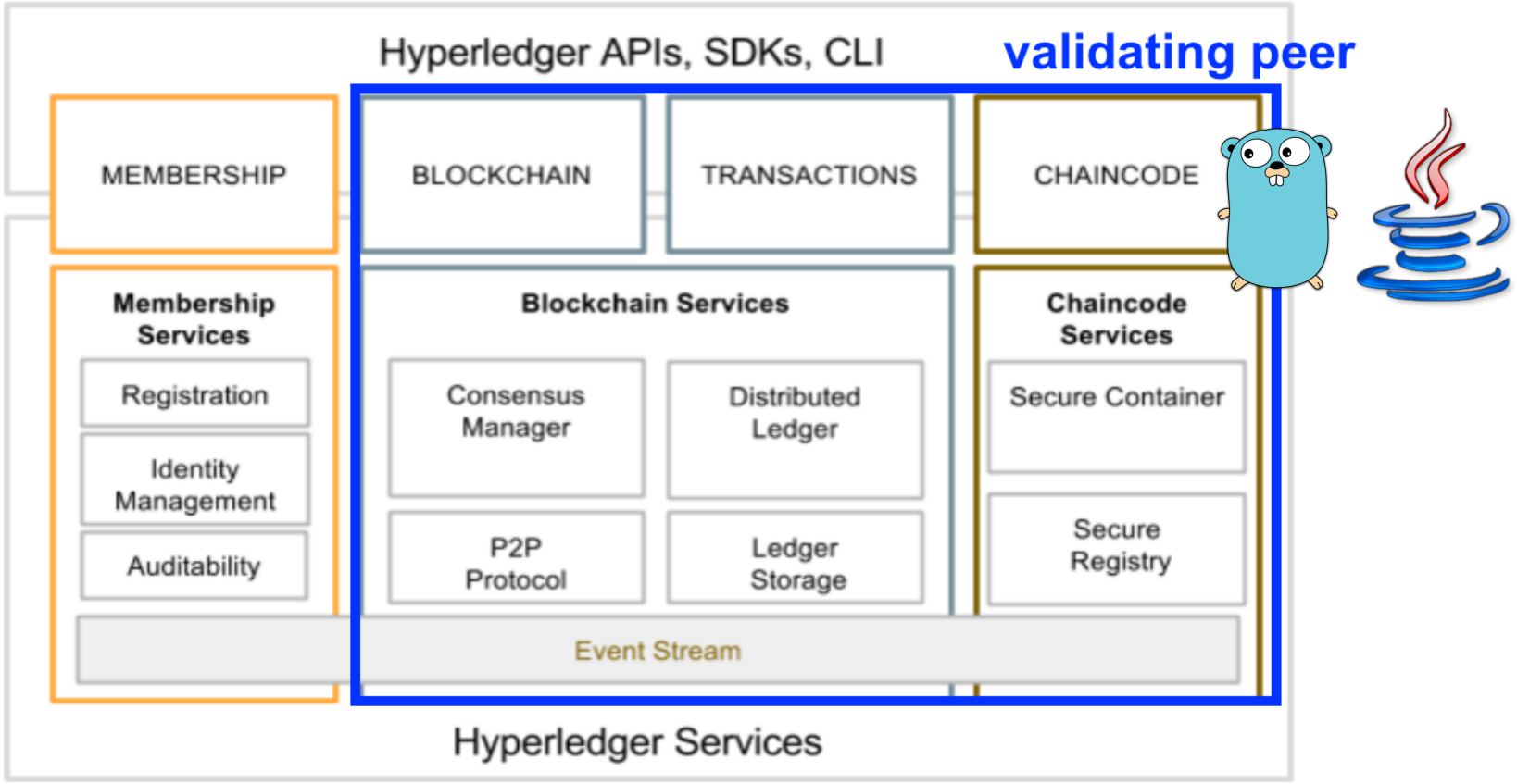
- Run your own application on any runtime available
- API based
- Enterprise Grade Security
- No need to worry about infrastructure and base ledger network
- Same code as publicly available
- IBM Blockchain also available on DockerHub
- IBM blockchain on Dedicated Cloud



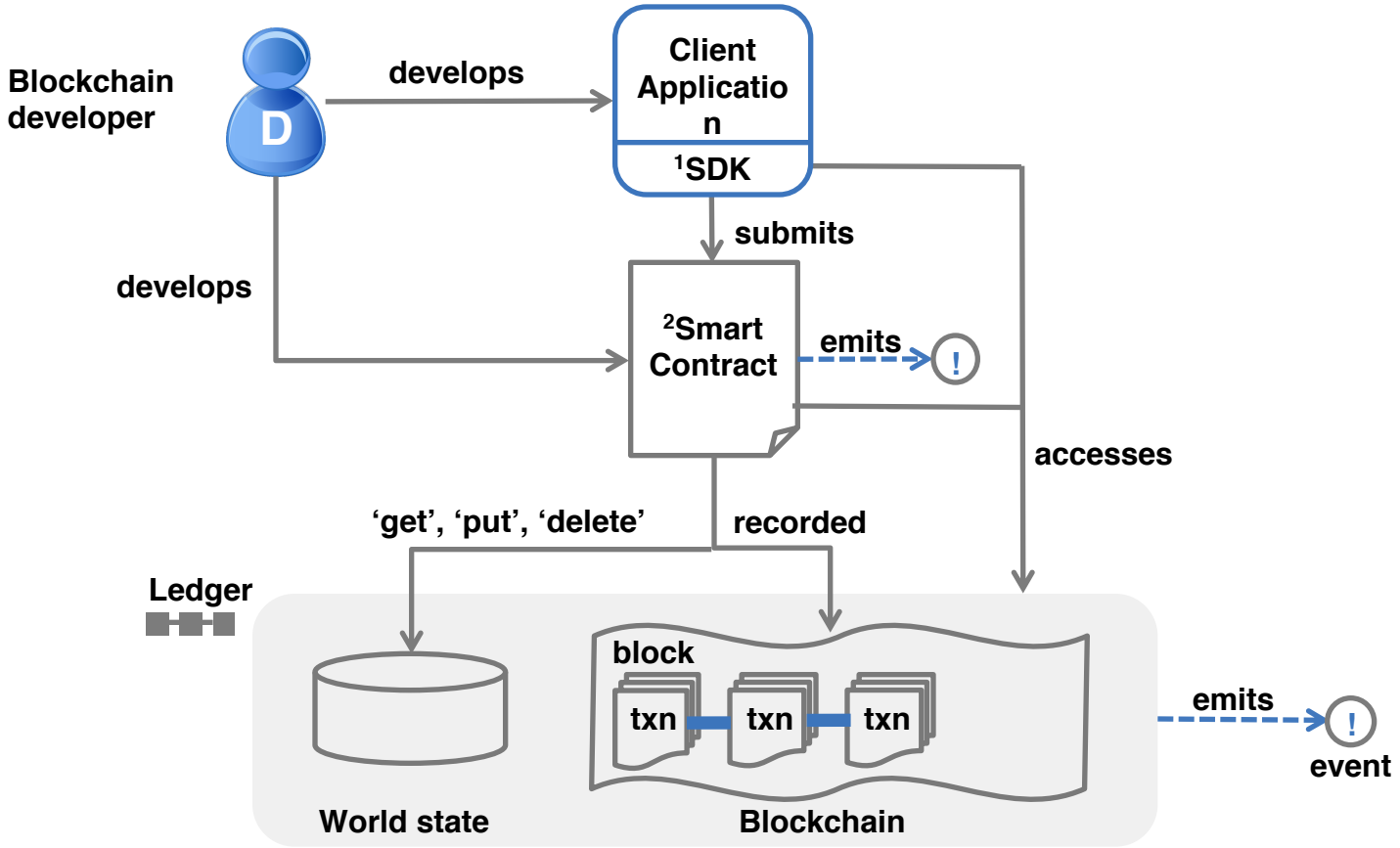
IBM **Bluemix** Hyperledger Service



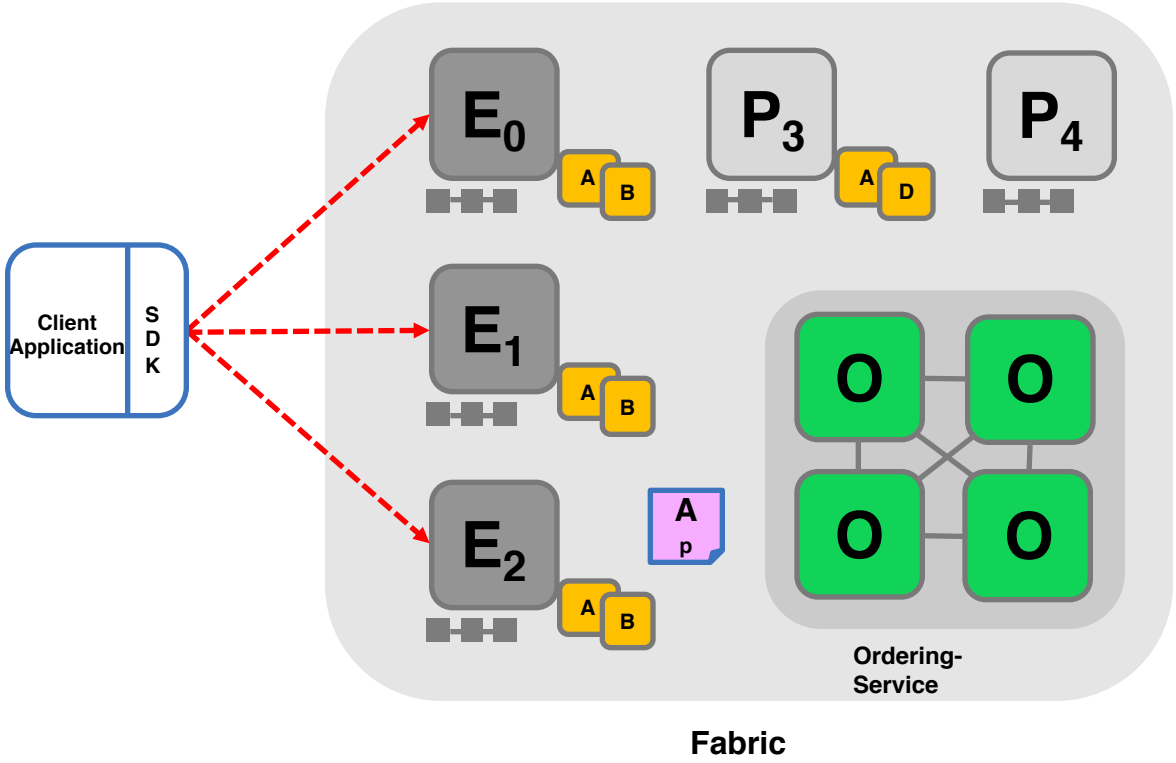
Hyperledger – Architecture 0.6



Hyperledger Blockchain – Single Node View



Smart Contract Execution

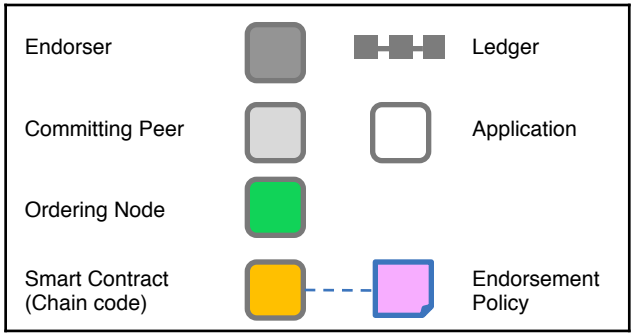


Application proposes transaction

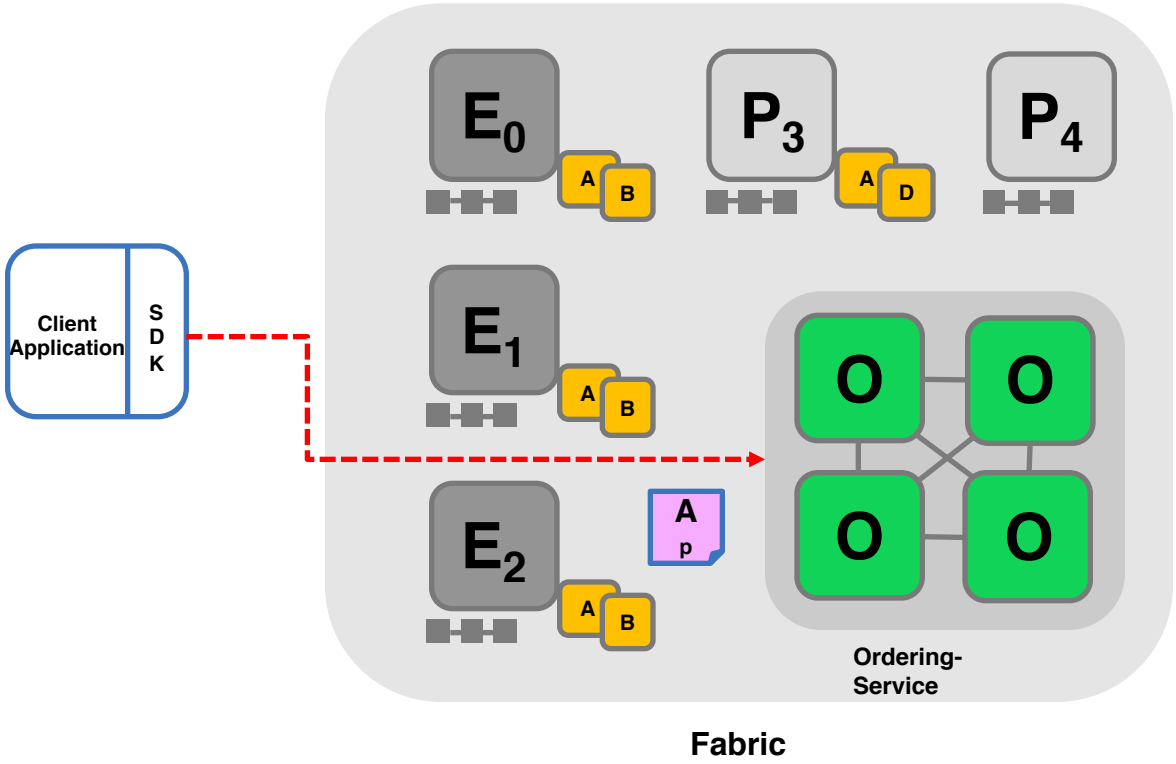
Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

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

Key:



Smart Contract Execution

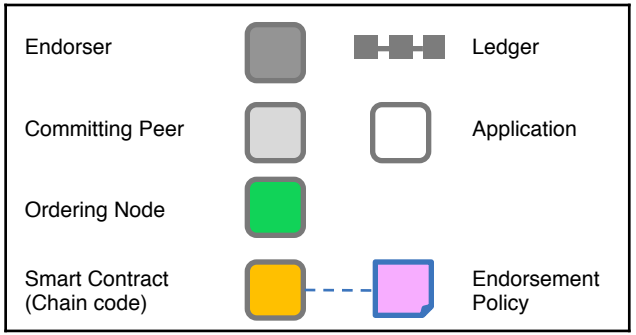


Application submits responses for ordering

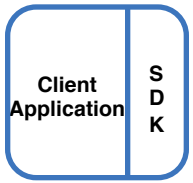
Application submits responses as a transaction to be ordered.

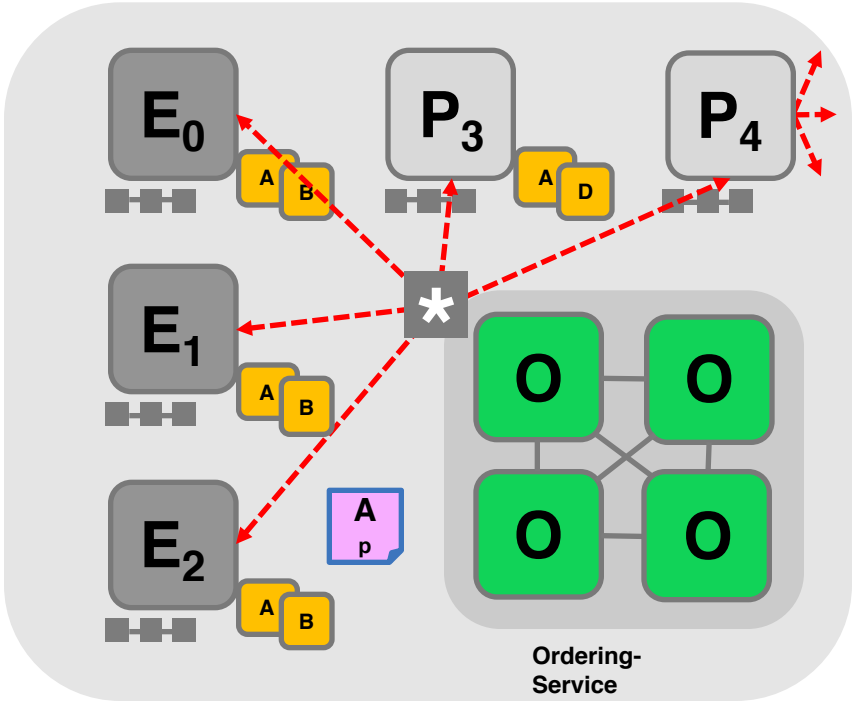
Ordering happens across the fabric in parallel with transactions submitted by other applications

Key:



Smart Contract Execution





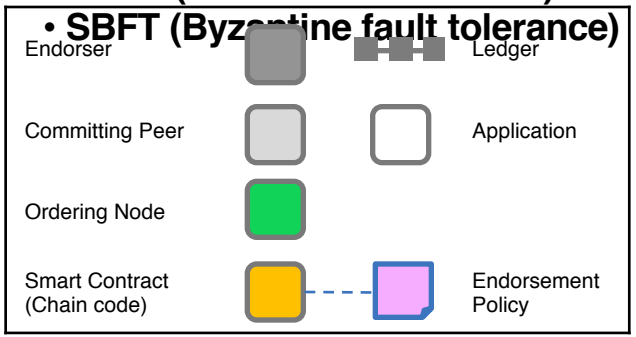
Fabric

Orderer delivers to all committing peers

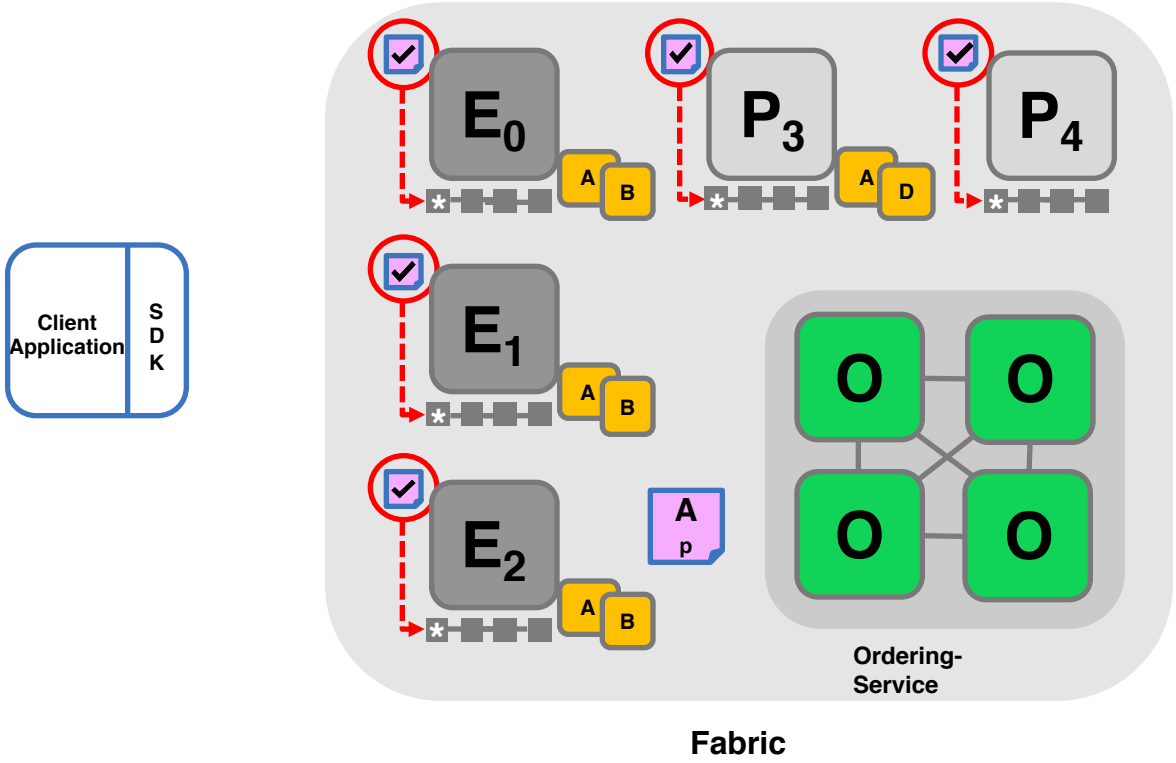
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

- **SOLO (Single node, development)**
- **Kafka (Crash fault tolerance)**
- **SBFT (Byzantine fault tolerance)**



Smart Contract Execution

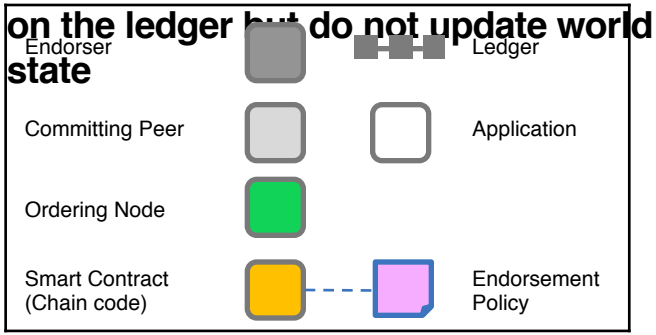


Committing peers validate transactions

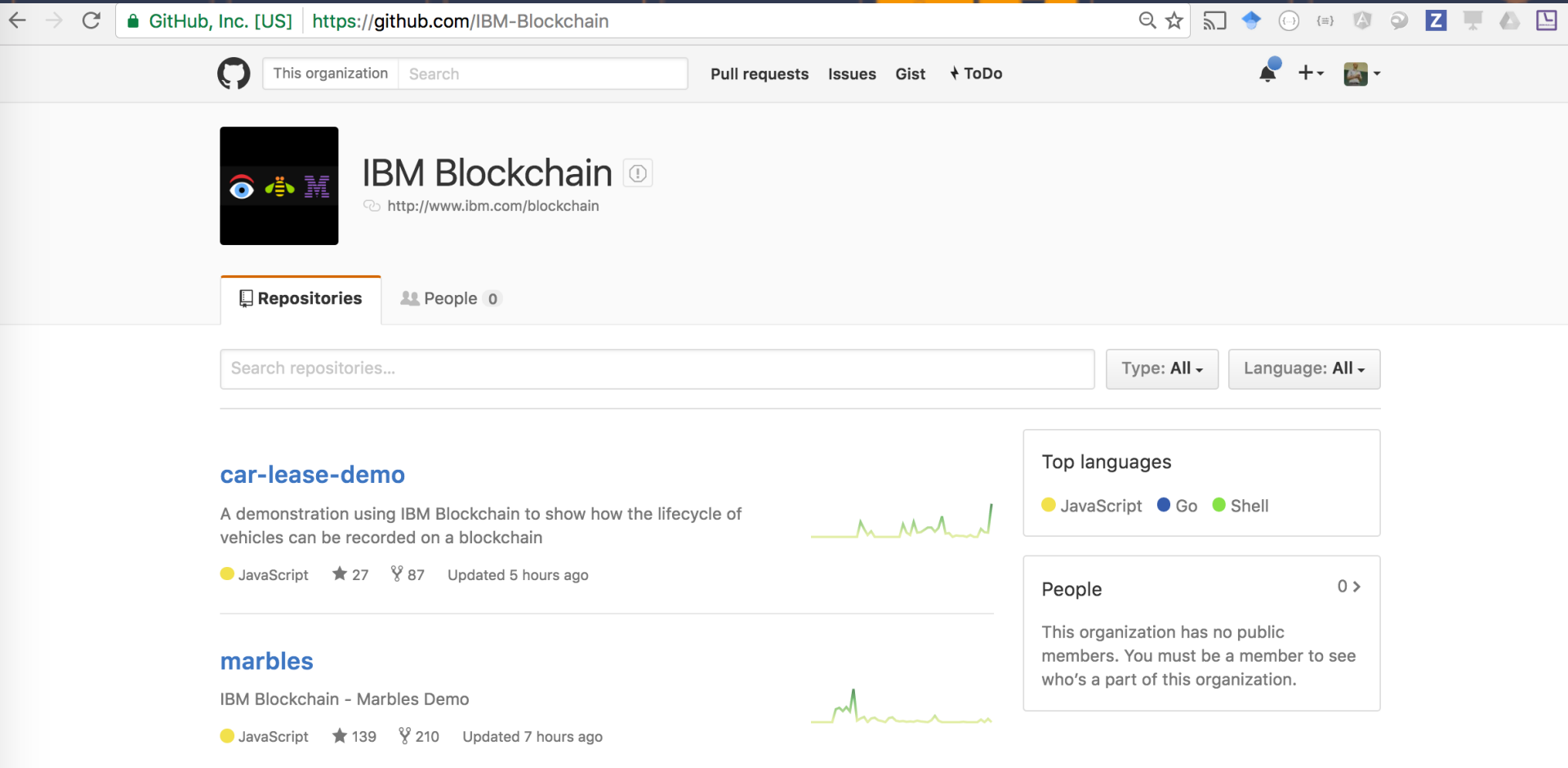
Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

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

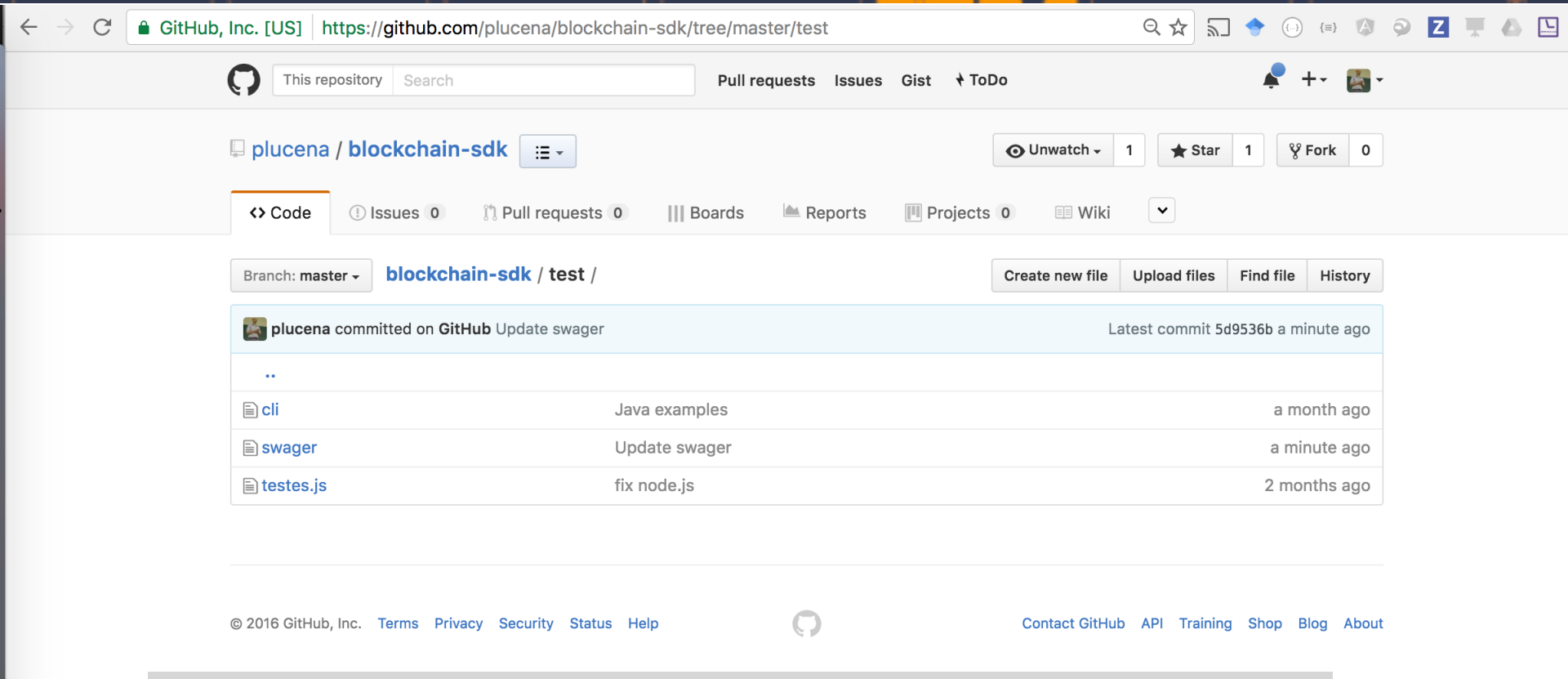
Invalid transactions are also retained on the ledger but do not update world state



Where do I start? —-> SAMPLE APPS



Where do I start?



TESTS - GO SMART CONTRACTS - JAVA SMART CONTRACTS

HOW TO WRITE SMART CONTRACTS

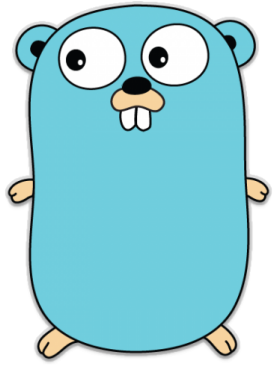
```
func (t *SimpleChaincode) Init(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error)

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

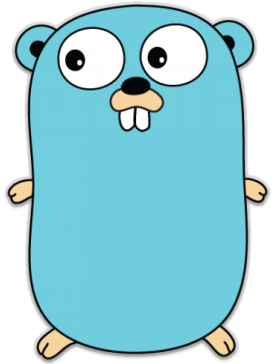
```
peer chaincode deploy -n mycc -c '{"Function":"init", "Args": ["a","100", "b", "200"]}'
```

https://obc-service-broker-prod.mybluemix.net/v2/swagger#!/Chaincode/chaincodeOp

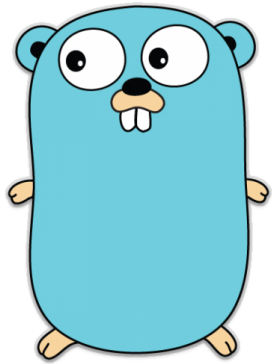
```go
package main

import (
	"errors"
	"fmt"

	"github.com/hyperledger/fabric/core/chaincode/shim"
)

// SimpleChaincode example simple Chaincode implementation
type SimpleChaincode struct {
}

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

// Init resets all the things
func (t *SimpleChaincode) Init(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
	if len(args) != 1 {
		return nil, errors.New("Incorrect number of arguments. Expecting 1")
	}

	err := stub.PutState("hello_world", []byte(args[0]))
	if err != nil {
		return nil, err
	}

	return nil, nil
}
```



```go
// Invoke isur entry point to invoke a chaincode function
func (t *SimpleChaincode) Invoke(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
        fmt.Println("invoke is running " + function)

        // Handle different functions
        if function == "init" {
                return t.Init(stub, "init", args)
        } else if function == "write" {
                return t.write(stub, args)
        }
        fmt.Println("invoke did not find func: " + function)

        return nil, errors.New("Received unknown function invocation")
}

// Query is our entry point for queries
func (t *SimpleChaincode) Query(stub *shim.ChaincodeStub, function string, args []string) ([]byte, error) {
        fmt.Println("query is running " + function)

        // Handle different functions
        if function == "read" { //read a variable
                return t.read(stub, args)
        }
        fmt.Println("query did not find func: " + function)

        return nil, errors.New("Received unknown function query")
}
```



```go
// write - invoke function to write key/value pair
func (t *SimpleChaincode) write(stub *shim.ChaincodeStub, args []string) ([]byte, error) {
	var key, value string
	var err error
	fmt.Println("running write()")

	if len(args) != 2 {
		return nil, errors.New("Incorrect number of arguments. Expecting 2. name of the key and value to set")
	}

	key = args[0] //rename for funsies
	value = args[1]
	err = stub.PutState(key, []byte(value)) //write the variable into the chaincode state
	if err != nil {
		return nil, err
	}
	return nil, nil
}

// read - query function to read key/value pair
func (t *SimpleChaincode) read(stub *shim.ChaincodeStub, args []string) ([]byte, error) {
	var key, jsonResp string
	var err error

	if len(args) != 1 {
		return nil, errors.New("Incorrect number of arguments. Expecting name of the key to query")
	}

	key = args[0]
	valAsbytes, err := stub.GetState(key)
	if err != nil {
		jsonResp = "{\"Error\":\"Failed to get state for " + key + "\"}"
		return nil, errors.New(jsonResp)
	}

	return valAsbytes, nil
}
```



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