Chaincode

- Go / Node.JS로 작성
- 체인코드는 장부 원장 상태를 읽고 업데이트하도록 작성된 프로그램 (스마트 계약서)
  - 모든 비즈니스 로직은 체인 코드 안에 있음
- 체인코드는 피어에서 실행되고 트랜잭션을 생성하는 '스마트 계약'
  - 사용자가 Hyperledger Fabric 네트워크의 공유 원장에서 트랜잭션을 생성하고 자산의 World State를 업데이트 가능
- 체인 코드는 프로그래밍 가능한 코드이며 Go로 작성되고 채널에서 인스턴스화
  - 개발자는 체인 코드를 사용하여 비즈니스 계약, 자산 정의를 개발하고 집합 적으로 분산 된 응용 프로그램을 관리
  - 체인코드는 응용 프로그램에 의해 호출 된 트랜잭션을 통해 원장 상태를 관리
  - 자산은 특정 체인 코드에 의해 생성되고 업데이트되며 다른 체인 코드로 액세스 할 수 없음
- 응용 프로그램은 체인 코드를 통해 블록 체인 원장과 상호 작용
  - 체인 코드는 트랜잭션을 보증하고 채널에서 인스턴스화하는 모든 피어에 설치 해야 함

Chaincode

- Hyperledger Fabric에서 Smart Contract를 개발하는 방법
  1) 개별 계약을 독립 실행 형 체인 코드로 코드화
  2) 체인 코드를 사용하여 하나 또는 여러 유형의 비즈니스 계약 수명주기를 관리하는 분산 응용 프로그램 생성
     - 최종 사용자가 이러한 응용 프로그램 내에서 계약 인스턴스를 인스턴스화 하여 사용 (O)

- Chaincode Key APIs
  - ChaincodeStub
    : 기본 원장과 상호 작용하여 자산을 쿼리, 업데이트 및 삭제할 수 있는 기능을 제공
  - ChaincodeStubInterface

Chaincode

- **func (stub *ChaincodeStub) GetState(key string) ([]byte, error)**
  - 원장에서 지정된 키 값을 반환
  - GetState는 장부에 커밋되지 않은 Write set에서 데이터를 읽지 않는다.
    - 커밋 되지 않은 PutState에 의해 수정 된 데이터를 고려하지 않음
    - 키가 상태 데이터베이스에 없으면 (nil, nil)이 반환

- **func (stub *ChaincodeStub) PutState(key string, value []byte) error**
  - 지정된 키와 값을 트랜잭션 쓰기 세트에 데이터 쓰기 제안으로 설정
  - PutState는 트랜잭션이 확인되고 성공적으로 커밋 될 때까지 원장에 영향을 미치지 않는다.

- **func (stub *ChaincodeStub) DelState(key string) error**
  - 트랜잭션 제안서의 쓰기 세트에서 삭제할 지정된 키를 기록
  - 트랜잭션이 확인되고 성공적으로 커밋되면 키와 값이 장부에서 삭제

Chaincode

- **Chaincode 프로그램 작성 시 2가지 메소드 구형**
  - **Init**
    - **체인 코드가 인스턴스화 또는 업그레이드 트랜잭션을 수신 할 때 호출**
    - **응용 프로그램 상태를 초기화**
  - **Invoke**
    - **invoke 트랜잭션이 수신되어 모든 트랜잭션 제안을 처리 할 때 호출**

1) **개발자는 체인 코드 내에 Init 메서드와 Invoke 메서드를 모두 구현**
2) **체인 코드는 피어 체인 코드 설치 명령을 사용하여 설치하고 체인 코드를 호출하기 전에 피어 체인 코드 인스턴스화 명령을 사용하여 인스턴스화**
3) **다음 피어 체인 코드 호출 또는 피어 체인 코드 쿼리 명령을 사용하여 트랜잭션을 생성 가능**

Sample Chaincode - Dependencies

```
package main
import (
    “fmt”
    “github.com/hyperledger/fabric/core/chaincode/shim”
    “github.com/hyperledger/fabric/protos/peer”
)
```

- fmt
  - 디버깅 / 로깅을위한 Println 포함
- github.com/hyperledger/fabric/core/chaincode/shim
  - Chaincode Key API 섹션에 설명 된대로 원장과 상호 작용해야 할 chaincode 인터페이스 및 chaincode 스텁에 대한 정의가 포함
- github.com/hyperledger/fabric/protos/peer
  - 피어 protobuf 패키지를 포함

Sample Chaincode - Struct

```
type SampleChaincode struct {

}
```

- 오브젝트 / 클래스의 정의
  - SampleChaincode는 자산을 관리하기위한 간단한 체인 코드를 구현

Sample Chaincode – Init Method

```
func (t *SampleChaincode) Init(stub shim.ChainCodeStubInterface) peer.Response {
        // Get the args from the transaction proposal
        args := stub.GetStringArgs()
        if len(args) != 2 {
            return shim.Error("Incorrect arguments. Expecting a key and a value")
        }
        // We store the key and the value on the ledger
        err := stub.PutState(args[0], []byte(args[1]))
        if err != nil {
            return shim.Error(fmt.Sprintf("Failed to create asset: %s", args[0]))
        }
        return shim.Success(nil)
}
```

- Init 구현은 두 개의 매개 변수를 입력으로 받아들이고 stub.PutState 함수를 사용하여 원장에 키 / 값 쌍을 작성하도록 제안
- stub.GetStringArgs는 키 / 값 쌍이 될 것으로 예상되는 인수의 유효성을 검색하고 검사
  - 두 개의 인수가 지정되었는지 확인
  - 그렇지 않은 경우 Init 메소드의 오류를 반환하여 문제가 발생했음을 나타냄
- stub.PutState 함수는 장부에 초기 상태를 저장하는 데 사용
  - 첫 번째 인수를 키로 지정하고 두 번째 인수를 해당 키의 값으로 지정하여 함수를 호출
  - 오류가 반환되지 않으면 Init 메서드에서 성공을 반환

Sample Chaincode – Invoke Method

```
func (t *SampleChaincode) Invoke(stub shim.ChaincodeStubInterface) peer.Response {
        // Extract the function and args from the transaction proposal
        fn, args := stub.GetFunctionAndParameters()
        var result string
        var err error
        if fn == "set" {
            result, err = set(stub, args)
        } else { // assume 'get' even if fn is nil
            result, err = get(stub, args)
        }
       //Failed to get function and/or arguments from transaction proposal
        if err != nil {
            return shim.Error(err.Error())
        }
        // Return the result as success payload
        return shim.Success([]byte(result))
    }
```

- 클라이언트가 호출 할 수 있는 두 가지 기본 동작은 get 및 set
  - get 메소드: 기존 자산의 값을 쿼리하고 반환하는 데 사용
  - set 메소드는 새 자산을 작성하거나 기존 자산의 값을 갱신하는 데 사용
  - GetFunctionandParameters를 호출하여 함수 이름과 매개 변수 변수를 분리
    - 각 트랜잭션은 set 또는 get 중 하나

Sample Chaincode – Invoke Method

```
func set(stub shim.ChaincodeStubInterface, args []string) (string, error) {
        if len(args) != 2 {
            return "", fmt.Errorf("Incorrect arguments. Expecting a key and a value")
        }
        err := stub.PutState(args[0], []byte(args[1]))
        if err != nil {
            return "", fmt.Errorf("Failed to set asset: %s", args[0])
        }
        return args[1], nil
}
```

- set 메소드는, 지정된 값으로 키에 의해 식별되는 자산을 작성 또는 변경
- set 메소드는 지정된 키 / 값 쌍을 포함하도록 World State를 수정
  - 키가 존재하면 PutState 메서드를 사용하여 키가 새 값으로 ***대체***
  - 그렇지 않으면 새 자산이 지정된 값으로 ***작성***

Sample Chaincode – Invoke Method

```
func get(stub shim.ChaincodeStubInterface, args []string) (string, error) {
        if len(args) != 1 {
            return "", fmt.Errorf("Incorrect arguments. Expecting a key")
        }
        value, err := stub.GetState(args[0])
        if err != nil {
            return "", fmt.Errorf("Failed to get asset: %s with error: %s", args[0], err)
        }
        if value == nil {
            return "", fmt.Errorf("Asset not found: %s", args[0])
        }
        return string(value), nil
    }
```

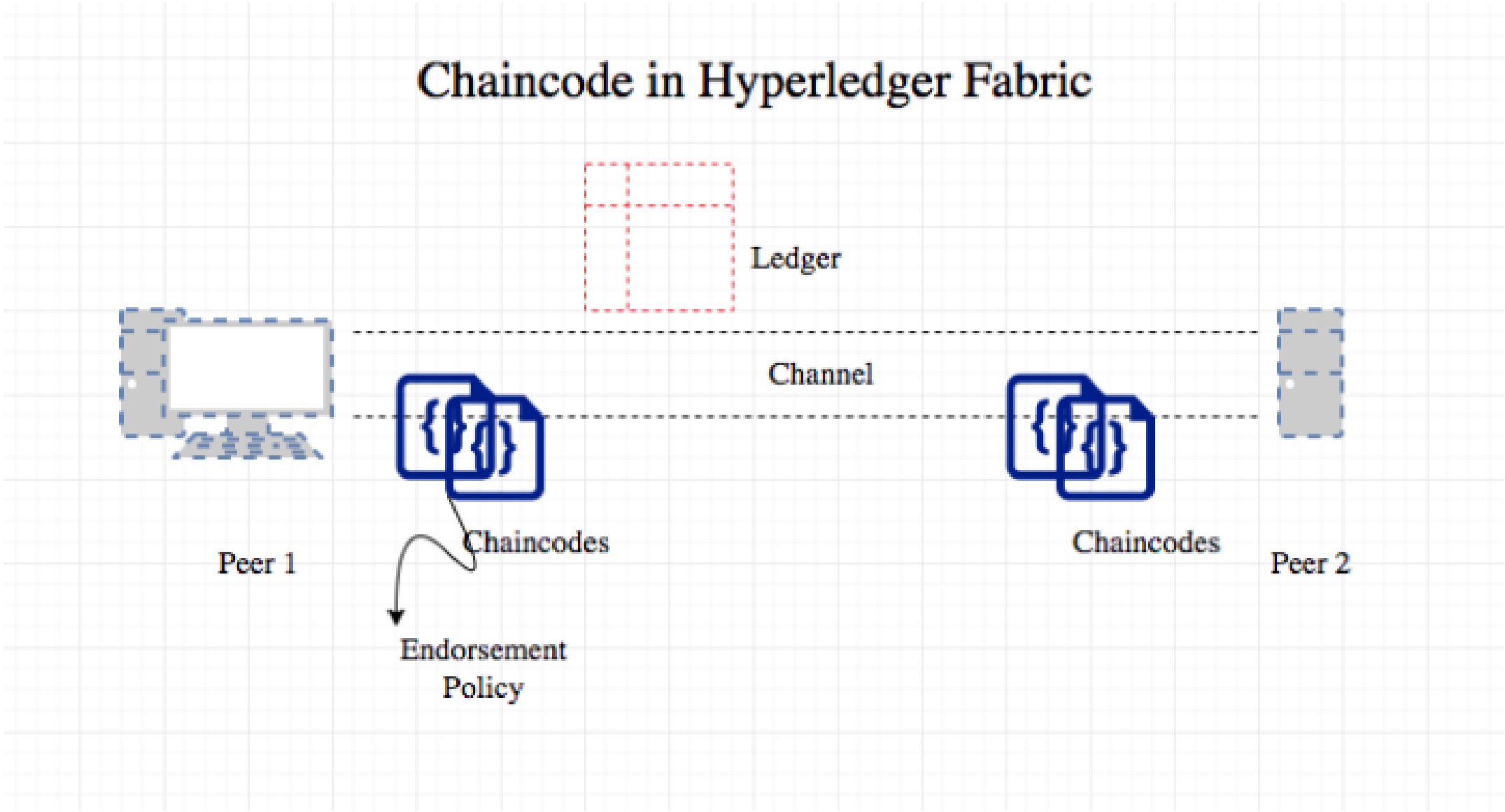
- get 메소드는, 지정된 키의 값을 취득
- 응용 프로그램이 단일 키를 전달하지 않으면 오류가 리턴
- 그렇지 않으면 GetState 메서드를 사용하여 지정된 키의 World State를 쿼리
  - 키가 장부 (및 월드 상태)에 아직 추가되지 않은 경우 오류가 반환
  - 그렇지 않은 경우, 지정된 키에 대해 설정된 값이 메소드에서 리턴

Sample Chaincode – Main Method

```
func main() {
        err := shim.Start(new(SampleChaincode))
        if err != nil {
            fmt.Println("Could not start SampleChaincode")
        } else {
            fmt.Println("SampleChaincode successfully started")
        }
    }
```

- 이 샘플의 마지막 코드는 Start 함수를 호출하는 main 함수
- main 함수는 인스턴스화 중에 컨테이너에서 체인 코드를 시작

Sample Chaincode – Main Method



Scenario

"Tuna 2020 Traceability Declaration 앞두고, 우리의 목표는 불법적이고, 보고되지 않는, 규제되지 않는 어획을 제거 하는 것이다. Hyperledger Fabric을 사용하여 참치 어업 공급망에 대한 투명성과 명확성을 제공해 보자."

"***Sarah***가 참치를 잡을 때부터 ***Miriam's restaurant***에 납품하기 까지의 과정을 어떻게 개선 할 수 있는지 설명하고, 이 때 데이터의 유효성과 참치 어획량의 지속 가능성을 확인하는 감독관과 같은 다른 당사자도 참여하게 될 것이다."

"Hyperledger Fabric을 사용하여 프로세스의 각 부분을 추적"

전체 시나리오

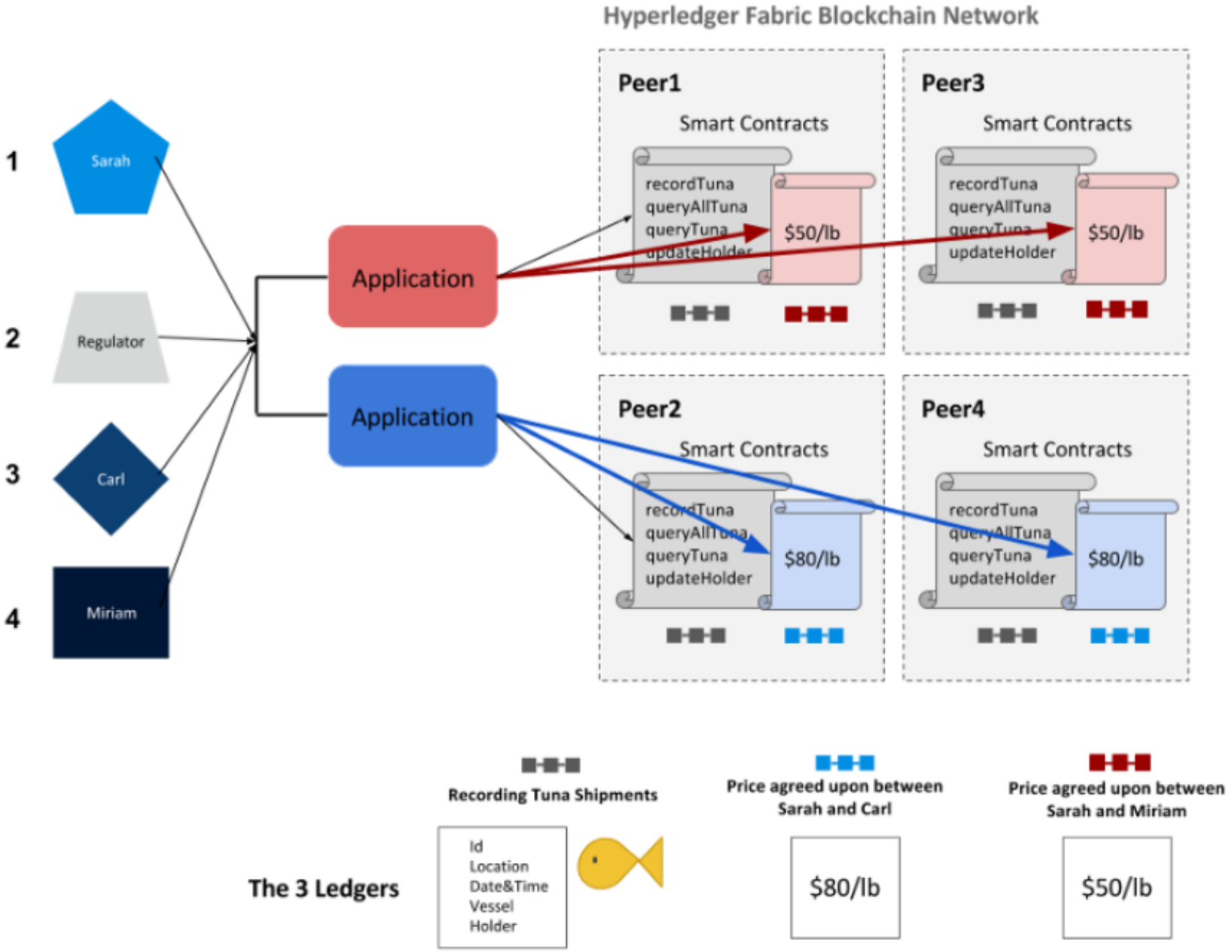
- Sarah는 참치를 잡고 공급망 응용 프로그램의 UI를 사용하여 어획에 대한 모든 세부 사항을 원장에게 기록
- 장부에 도달하기 전에 트랜잭션은 네트워크의 승인하는 동료에게 전달 → 승인
- 승인(보증) 된 트랜잭션은 Ordering Service 전송되어 블록안에서 시간순으로 정렬
- 이 블록은 네트워크의 커밋하는 피어 (peer)에게 전송 → 유효성을 검증한 뒤에 커밋

- 참치가 공급망을 지나 감에 따라 규제 당국은 고유 한 애플리케이션을 사용하여 원장에게 특정 어획에 대한 세부 정보를 질의(query) 가능 (가격 관련 체인 코드에 대한 액세스 권한이 없기 때문에 가격 제외)
- Sarah는 Carl과 계약을 체결 할 수 있으며 1 파운드 당 80 달러의 가격에 동의

  → $ 80 / lb를 규정하는 체인 코드 계약에 파란색 채널을 사용

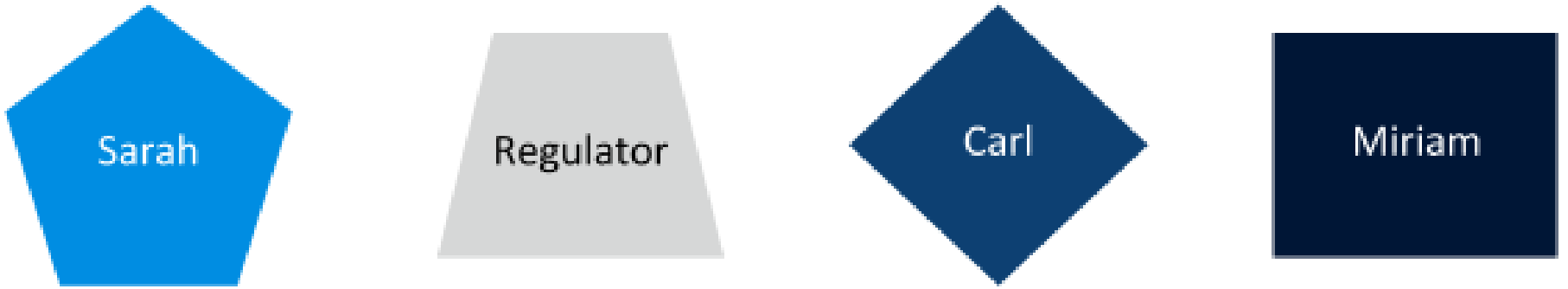
  → 파란색 채널의 원장은이 트랜잭션이 포함 된 블록으로 업데이트

전체 시나리오


Hyperledger Fabric Blockchain Network
1
Sarah
2
Regulator
3
Carl
4
Miriam
Application
Application
Peer1
Smart Contracts
recordTuna
queryAllTuna
queryTuna
updateHolder
$50/lb
Peer3
Smart Contracts
recordTuna
queryAllTuna
queryTuna
updateHolder
$50/lb
Peer2
Smart Contracts
recordTuna
queryAllTuna
queryTuna
updateHolder
$80/lb
Peer4
Smart Contracts
recordTuna
queryAllTuna
queryTuna
updateHolder
$80/lb
Recording Tuna Shipments
Price agreed upon between Sarah and Carl
Price agreed upon between Sarah and Miriam
The 3 Ledgers
Id
Location
Date&Time
Vessel
Holder
$80/lb
$50/lb

Scenario

- ***Sarah***는 지속 가능하고 합법적으로 ***참치를 잡는*** 어부
- ***Regulators***은 참치가 합법적으로 그리고 지속 가능하게 ***잡힌 것을 확인***
- ***Miriam***은 예제에서 ***최종 사용자*** 역할을 할 식당 주인
- ***Carl***은 또 다른 레스토랑 주인으로 Sarah가 Carl에게 ***참치를 판매*** 할 수 있음



Stakeholders

- ***Sarah***
  - 매번 참치를 잡은 다음 고유한 ID 번호, 어획 위치와 시간, 무게, 혈관 유형에 대한 정보를 참치를 잡은 사람과 함께 기록
  - 6가지의 데이터 속성을 가짐
  - 세부 정보는 Chaincode의 Key/Value 쌍으로 World State에 저장

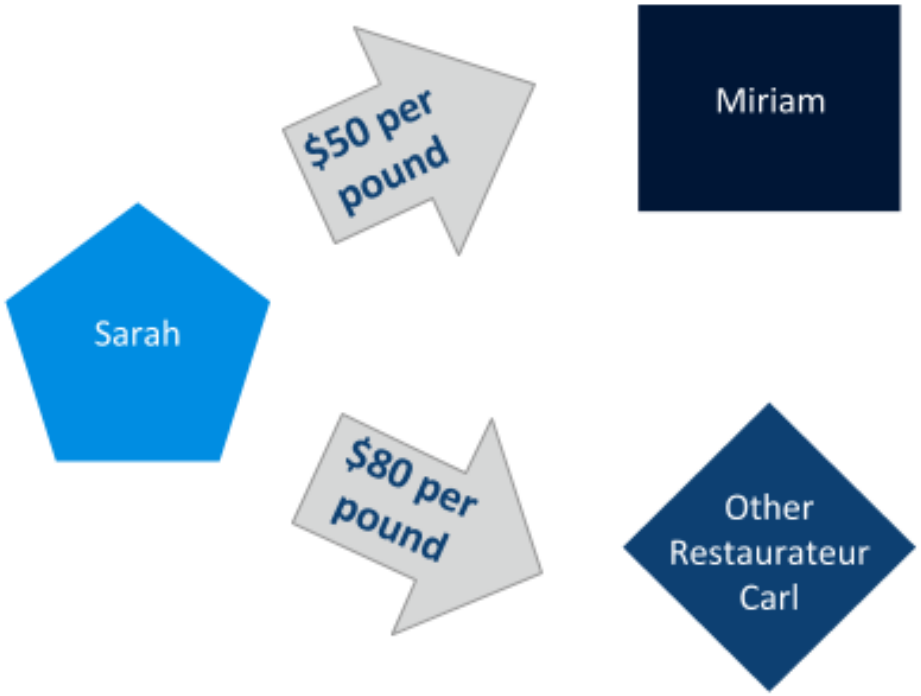
```
var tuna = {
        id: '0001',
        holder: 'Sarah',
        location: { latitude: '41.40238', longitude: '2.170328'},
        when: '20170630123546',
        weight: '58lbs',
        vessel : '9548E'
}
```

Stakeholders

- ***Restaurant Owner (Miriam, Buyer)***
  - 저렴하고 높은 품질의 참치를 찾는 식당 주인
  - 불법과 보고되지 않은 참치의 어획에 따른 중요성으로 인해, 매번 구입하는 참치가 합법적이고 지속 가능하게 잡히는 것인지 믿을 수 있는지 불확실 함
  - Sarah는 합리적인 가격으로 참치를 판매하기 위해 노력하며, 자신이 누구에게 판매하는지, 어떤 가격으로 판매하는지에 대해 스스로 알기를 원함

Stakeholders

- ***Restaurant Owner (Miriam & Cole, Sale)***
  - 일반적으로 Sarah는 Carl과 같은 식당주인에게 1파운드당 $80씨 참치를 판매
  - Miriam에게는 1파운드당 $50의 특별 가격으로 제공
  - 전통적인 블록체인에서는 Sarah와 Miriam의 거래를 전체 네트워크에서 알게 됨



- 문제를 해결하기 위해 거래의 세부 사항을 네트워크에 연결된 모든 사람이 ***사용할 수 없도록***
  - ➔ 단, 자신이 판매하는 참치의 세부 정보는 보이게 하고 싶음
- Hyperledger Fabric의 채널을 사용하여 Sarah와 Miriam의 계약 조건을 개인적으로 동의 가능
- 다른 사람들은 이 거래를 보지 못함

Stakeholders

- ***Regulators, Contract Enforcers***
  - 조정관은 Hyperledger Fabric 블록체인 네트워크에 접속하여 Ledger의 ***세부 사항을 확인*** 가능
  - Ledger를 조회하고 Sarah가 ***합법적으로 참치를 잡았는지***에 대해 확인 가능
  - 규제당국은 조회만 가능하며 Ledger에 항목을 추가할 수 없음

Stakeholders

- Gaining Network Membership
  - Hyperledger Fabric은 승인 된 네트워크이므로 승인 된 참가자만 네트워크에 접속 가능
  - 네트워크 멤버쉽 및 ID를 처리하기 위해 MSP는 사용자 ID를 관리하고 네트워크의 모든 참가자를 인증
  - 시나리오에서는 Regulator, 어부, 승인된 식당주인만이 유일한 네트워크 참가자

  우리는 3개의 분리된 채널에서 실행되는 2개의 분리된 체인코드를 가짐

  - Chaincode
    - 어부와 식당 주인 사이에서 가격 합의를 위해 사용
    - 참치의 소유권을 전송하기 위해 사용

  - Channels
    - Seller(Sarah)와 Buyer(Miriam) 사이에서 가격 합의를 위해 사용
    - Seller(Sarah)와 Buyer(Carl) 사이에서 가격 합의를 위해 사용
    - 참치의 소유권을 전송하기 위해 사용

Asset 정의

장부에 기록 될 참치의 속성

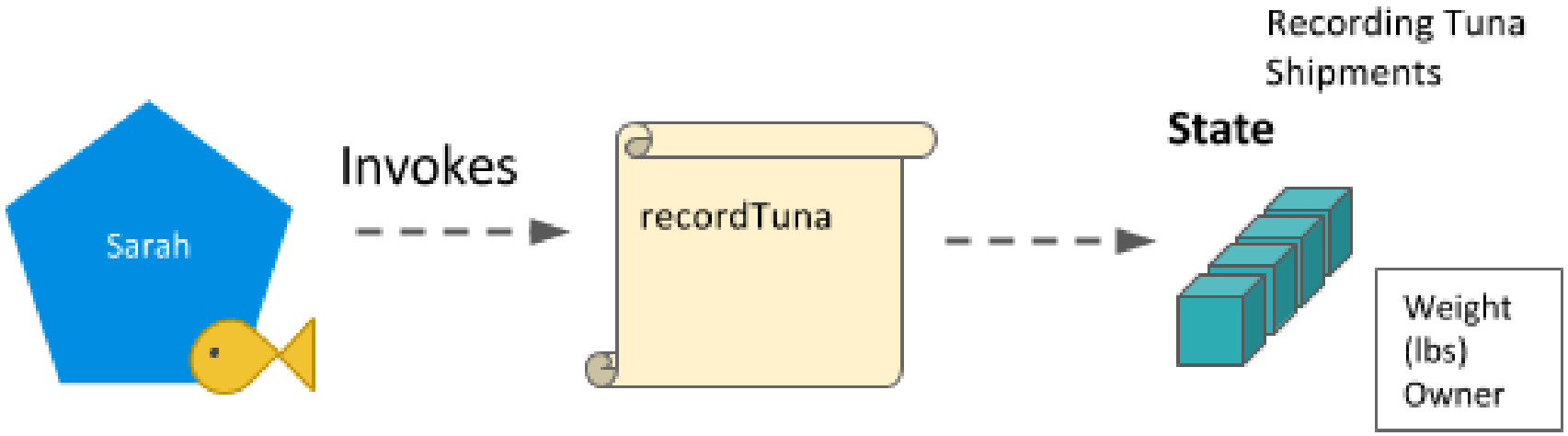
- Vessel (string)
- Location (string)
- Date and Time (datetime)
- Holder (string)

```
type Tuna struct {
        Vessel string ‘json:”vessel”’
        Datetime string ‘json:”datetime”’
        Location string ‘json:”location”’
        Holder string ‘json:”holder”’
}
```

Chaincode 구현 – Invoke Method

- 앞에서 설명한 것처럼 Invoke 메서드는 클라이언트 응용 프로그램에서 트랜잭션을 제안 할 때 호출
- ***recordTuna***

  → Sarah가 각 참치를 잡을 때 ***호출*** 할 것입니다.
- queryTuna
- changeTunaHolder

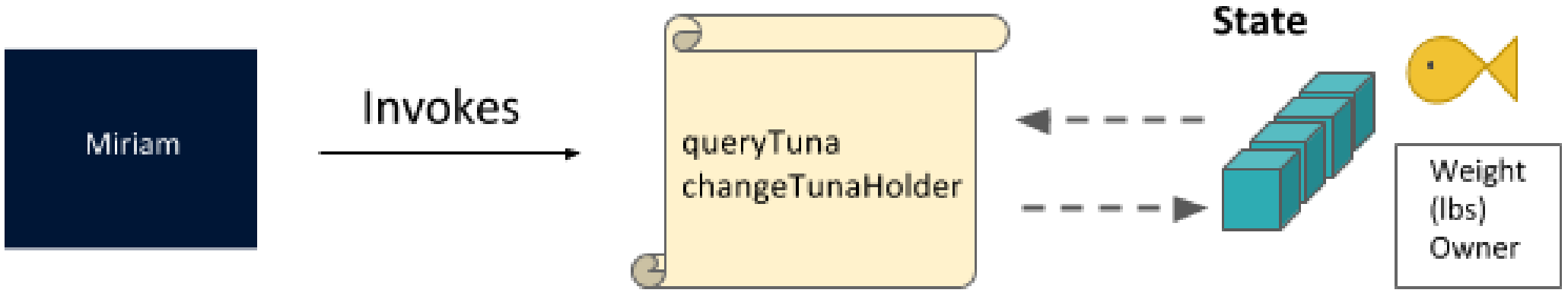


Chaincode 구현 – Invoke Method

- 앞에서 설명한 것처럼 Invoke 메서드는 클라이언트 응용 프로그램에서 트랜잭션을 제안 할 때 호출
- recordTuna
- ***queryTuna***
- ***changeTunaHolder***

  → 공급망을 통과 또는 특정 참치 어류를 수령, 전달할 때 식당 주인 Miriam에 의해 호출

  → 특정 참치의 상태를 보기 위해 식당인 Miriam에 의해 호출

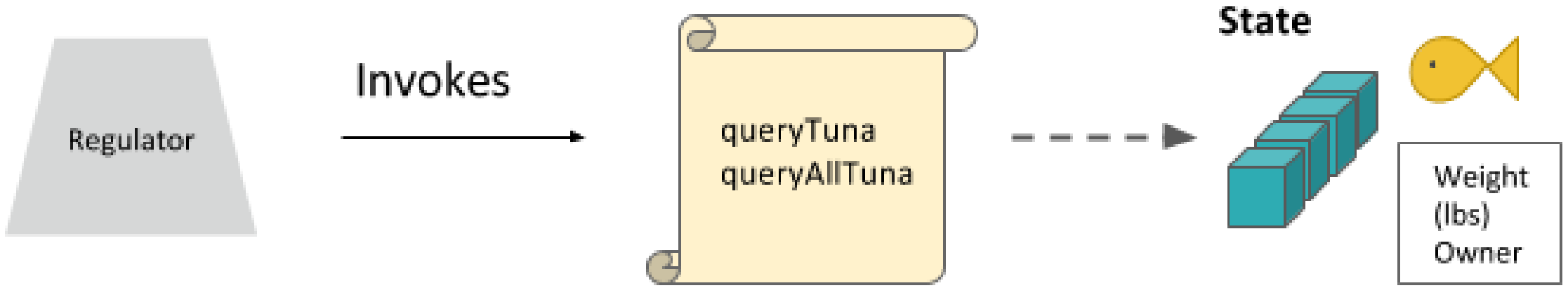


Chaincode 구현 – Invoke Method

- 앞에서 설명한 것처럼 Invoke 메서드는 클라이언트 응용 프로그램에서 트랜잭션을 제안 할 때 호출
- recordTuna
- ***queryTuna***
- ***changeTunaHolder***

  → 감독 당국은 공급망의 지속 가능성을 확인

  → 확인해야 할 필요성에 따라 queryTuna 및 queryAllTuna를 호출



Chaincode 구현 – Invoke Method

```
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {
	// Retrieve the requested Smart Contract function and arguments
	function, args := APIstub.GetFunctionAndParameters()

	// Route to the appropriate handler function to interact with the ledger appropriately
	if function == "queryTuna" {
		return s.queryTuna(APIstub, args)
	} else if function == "initLedger" {
		return s.initLedger(APIstub)
	} else if function == "recordTuna" {
		return s.recordTuna(APIstub, args)
	} else if function == "queryAllTuna" {
		return s.queryAllTuna(APIstub)
	} else if function == "changeTunaHolder" {
		return s.changeTunaHolder(APIstub, args)
	}

	return shim.Error("Invalid Smart Contract function name.")
}
```

Chaincode 구현 – queryTuna

- 특정 참치의 기록을 보기 위해 어부, 레귤레이터 또는 식당가가 사용
- 참치를 조회하기 위해 하나의 인자 값을 갖음

```
func (s *SmartContract) queryTuna(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
	if len(args) != 1 {
		return shim.Error("Incorrect number of arguments. Expecting 1")
	}

	tunaAsBytes, _ := APIstub.GetState(args[0])

	if tunaAsBytes == nil {
		return shim.Error("Could not locate tuna")
	}

	return shim.Success(tunaAsBytes)
}
```

Chaincode 구현 – initLedger

- 우리의 네트워크에 테스트 데이터를 추가

```
func (s *SmartContract) initLedger(APIstub shim.ChaincodeStubInterface) sc.Response {
	tuna := []Tuna{
		Tuna{Vessel: "923F", Location: "67.0006, -70.5476", Timestamp: "1504054225", Holder: "Miriam"},
		Tuna{Vessel: "M83T", Location: "91.2395, -49.4594", Timestamp: "1504057825", Holder: "Dave"},
		Tuna{Vessel: "T012", Location: "58.0148, 59.01391", Timestamp: "1493517025", Holder: "Igor"},
		Tuna{Vessel: "P490", Location: "-45.0945, 0.7949", Timestamp: "1496105425", Holder: "Amalea"},
		...
		Tuna{Vessel: "EI89", Location: "-132.3207, -34.0983", Timestamp: "1485066691", Holder: "Yuan"},
		Tuna{Vessel: "129R", Location: "153.0054, 12.6429", Timestamp: "1485153091", Holder: "Carlo"},
		Tuna{Vessel: "49W4", Location: "51.9435, 8.2735", Timestamp: "1487745091", Holder: "Fatima"},
	}

	i := 0
	for i < len(tuna) {
		fmt.Println("i is ", i)
		tunaAsBytes, _ := json.Marshal(tuna[i])
		APIstub.PutState(strconv.Itoa(i+1), tunaAsBytes)
		fmt.Println("Added", tuna[i])
		i = i + 1
	}
	return shim.Success(nil)
```

Chaincode 구현 – recordTuna

- Sarah가 참치 잡이를 기록하는 데 사용하는 방법
- 5 개의 인자 (원장에 저장 될 속성)를 사용

```
func (s *SmartContract) recordTuna(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
	if len(args) != 5 {
		return shim.Error("Incorrect number of arguments. Expecting 5")
	}

	var tuna = Tuna{ Vessel: args[1], Location: args[2], Timestamp: args[3], Holder: args[4]}

	tunaAsBytes, _ := json.Marshal(tuna)

	err := APIstub.PutState(args[0], tunaAsBytes)

	if err != nil {
		return shim.Error(fmt.Sprintf("Failed to record tuna catch: %s", args[0]))
	}

	return shim.Success(nil)
}
```

Chaincode 구현 – queryAllTuna

- 모든 레코드를 평가
- 모든 참치 기록이 장부에 추가되어 있으며, 결과를 포함하는 JSON 문자열을 반환

```
func (s *SmartContract) queryAllTuna(APIstub shim.ChaincodeStubInterface) sc.Response {
	startKey := "0"
	endKey := "999"
	resultsIterator, err := APIstub.GetStateByRange(startKey, endKey)
	if err != nil {
		return shim.Error(err.Error())
	}
	defer resultsIterator.Close()
	// buffer is a JSON array containing QueryResults
	var buffer bytes.Buffer
	buffer.WriteString("[")
	bArrayMemberAlreadyWritten := false
	for resultsIterator.HasNext() {
		queryResponse, err := resultsIterator.Next()
		if err != nil {
			return shim.Error(err.Error())
		}
```

continue...

Chaincode 구현 – queryAllTuna

- 모든 레코드를 평가
- 모든 참치 기록이 장부에 추가되어 있으며, 결과를 포함하는 JSON 문자열을 반환

```
                // Add a comma before array members, suppress it for the first array member
                if bArrayMemberAlreadyWritten == true {
                        buffer.WriteString(",")
                }
                buffer.WriteString("{₩"Key₩":")
                buffer.WriteString("₩"")
                buffer.WriteString(queryResponse.Key)
                buffer.WriteString("₩"")
                buffer.WriteString(", ₩"Record₩":")
                // Record is a JSON object, so we write as-is
                buffer.WriteString(string(queryResponse.Value))
                buffer.WriteString("}")
                bArrayMemberAlreadyWritten = true
        }
        buffer.WriteString("]")
        fmt.Printf("- queryAllTuna:₩n%s₩n", buffer.String())
        return shim.Success(buffer.Bytes())

}
```

Chaincode 구현 – changeTunaHolder

- 참치가 공급망의 다른 당사자들에게 전달됨에 따라, World State의 데이터는 누가 소유하고 있는지를 업데이트 할 수 있음 → 참치 ID와 새로운 소유자 이름의 두 인수를 취함

```
func (s *SmartContract) changeTunaHolder(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
	if len(args) != 2 {
		return shim.Error("Incorrect number of arguments. Expecting 2")
	}
	tunaAsBytes, _ := APIstub.GetState(args[0])
	if tunaAsBytes != nil {
		return shim.Error("Could not locate tuna")
	}
	tuna := Tuna{}
	json.Unmarshal(tunaAsBytes, &tuna)
	// Normally check that the specified argument is a valid holder of tuna but here we are skipping this check for this example.
	tuna.Holder = args[1]
	tunaAsBytes, _ = json.Marshal(tuna)
	err := APIstub.PutState(args[0], tunaAsBytes)
	if err != nil {
		return shim.Error(fmt.Sprintf("Failed to change tuna holder: %s", args[0]))
	}
	return shim.Success(nil)
}
```

Tuna Application – 1) Clone the repo

$ git clone https://github.com/hyperledger/education.git

$ cd education/LFS171x/fabric-material/tuna-app

$ npm install

이전 컨테이너 존재할 경유 삭제

$ docker rm –f $ (docker ps –qa)

```
bcadmin@ubuntu:~$ git clone https://github.com/hyperledger/education.git
Cloning into 'education'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 756 (delta 0), reused 2 (delta 0), pack-reused 749
Receiving objects: 100% (756/756), 23.21 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (144/144), done.
Checking connectivity... done.
bcadmin@ubuntu:~$
```

```
bcadmin@ubuntu:~/education$ ll
total 36
drwxrwxr-x  4 bcadmin bcadmin  4096 Nov 10 01:34 ./
drwxr-xr-x 11 bcadmin bcadmin  4096 Nov 10 01:33 ../
drwxrwxr-x  8 bcadmin bcadmin  4096 Nov 10 01:34 .git/
-rw-rw-r--  1 bcadmin bcadmin    24 Nov 10 01:34 .gitignore
drwxrwxr-x  8 bcadmin bcadmin  4096 Nov 10 01:34 LFS171x/
-rw-rw-r--  1 bcadmin bcadmin 11358 Nov 10 01:34 LICENSE
-rw-rw-r--  1 bcadmin bcadmin    56 Nov 10 01:34 README.md
bcadmin@ubuntu:~/education$
```

Tuna Application – 2) Start Fabric Network

$ ./startFabric.sh

```
bcadmin@ubuntu:~/education/LFS171x/fabric-material/tuna-app$ ll
total 168
drwxrwxr-x 4 bcadmin bcadmin  4096 Nov 10 01:34 ./
drwxrwxr-x 5 bcadmin bcadmin  4096 Nov 10 01:34 ../
drwxrwxr-x 2 bcadmin bcadmin  4096 Nov 10 01:34 client/
-rw-rw-r-- 1 bcadmin bcadmin 20935 Nov 10 01:34 controller.js
-rw-rw-r-- 1 bcadmin bcadmin   171 Nov 10 01:34 .gitignore
-rw-rw-r-- 1 bcadmin bcadmin   579 Nov 10 01:34 package.json
-rw-rw-r-- 1 bcadmin bcadmin 96671 Nov 10 01:34 package-lock.json
-rw-rw-r-- 1 bcadmin bcadmin   567 Nov 10 01:34 README.md
-rw-rw-r-- 1 bcadmin bcadmin  2939 Nov 10 01:34 registerAdmin.js
-rw-rw-r-- 1 bcadmin bcadmin  3261 Nov 10 01:34 registerUser.js
-rw-rw-r-- 1 bcadmin bcadmin   462 Nov 10 01:34 routes.js
-rw-rw-r-- 1 bcadmin bcadmin  1204 Nov 10 01:34 server.js
drwxrwxr-x 2 bcadmin bcadmin  4096 Nov 10 01:34 src/
-rwxrwxr-x 1 bcadmin bcadmin  1820 Nov 10 01:34 startFabric.sh*
bcadmin@ubuntu:~/education/LFS171x/fabric-material/tuna-app$ ./startFabric.sh
```

1. 기존 Network Artifacts가 Docker 에서 제거되고 피어 & CouchDB 가 있는 단 하나의 조직이 Orderer와 함께 생성
2. startFabric.sh는 start.sh를 호출

Tuna Application – 2) Start Fabric Network

$ ./startFabric.sh

```
2018-11-09 16:38:53.240 UTC [grpc] DialContext -> DEBU 03d parsed scheme: ""
2018-11-09 16:38:53.242 UTC [grpc] DialContext -> DEBU 03e scheme "" not registered, fallback to default scheme
2018-11-09 16:38:53.243 UTC [grpc] watcher -> DEBU 03f ccResolverWrapper: sending new addresses to cc: [{peer0.org1.example.com:7051 0  <nil>}]
2018-11-09 16:38:53.243 UTC [grpc] switchBalancer -> DEBU 040 ClientConn switching balancer to "pick_first"
2018-11-09 16:38:53.243 UTC [grpc] HandleSubConnStateChange -> DEBU 041 pickfirstBalancer: HandleSubConnStateChange: 0xc4201fdf70, CONNECTING
2018-11-09 16:38:53.247 UTC [grpc] HandleSubConnStateChange -> DEBU 042 pickfirstBalancer: HandleSubConnStateChange: 0xc4201fdf70, READY
2018-11-09 16:38:53.250 UTC [msp] GetDefaultSigningIdentity -> DEBU 043 Obtaining default signing identity
2018-11-09 16:38:53.250 UTC [grpc] DialContext -> DEBU 044 parsed scheme: ""
2018-11-09 16:38:53.250 UTC [grpc] DialContext -> DEBU 045 scheme "" not registered, fallback to default scheme
2018-11-09 16:38:53.252 UTC [grpc] watcher -> DEBU 046 ccResolverWrapper: sending new addresses to cc: [{orderer.example.com:7050 0  <nil>}]
2018-11-09 16:38:53.252 UTC [grpc] switchBalancer -> DEBU 047 ClientConn switching balancer to "pick_first"
2018-11-09 16:38:53.252 UTC [grpc] HandleSubConnStateChange -> DEBU 048 pickfirstBalancer: HandleSubConnStateChange: 0xc4205f8ff0, CONNECTING
2018-11-09 16:38:53.260 UTC [grpc] HandleSubConnStateChange -> DEBU 049 pickfirstBalancer: HandleSubConnStateChange: 0xc4205f8ff0, READY
2018-11-09 16:38:53.262 UTC [msp/identity] Sign -> DEBU 04a Sign: plaintext: 0AAA070A6A08031A0B089DF096DF0510...1A0E0A0A696E69744C65646765720A00
2018-11-09 16:38:53.264 UTC [msp/identity] Sign -> DEBU 04b Sign: digest: 3A49369790E231D42AA990BAFB00DC3EC0DD14B064F60F728D8878534054A0D5
2018-11-09 16:38:53.290 UTC [msp/identity] Sign -> DEBU 04c Sign: plaintext: 0AAA070A6A08031A0B089DF096DF0510...E0FCA27E0EFCC02B2059AFF1C84A9DDB
2018-11-09 16:38:53.292 UTC [msp/identity] Sign -> DEBU 04d Sign: digest: 4A9D030A30DF2E38096E627DED146ECA4E016B9FF28115FDB293B5C6BE043A76
2018-11-09 16:38:53.303 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> DEBU 04e ESCC invoke result: version:1 response:<status:200 > payload:"\n \315\321y\037+,pB\270\222\266zDG\364A%9\004\313\033\034\20
0\311\224\316+l#Q\303@\022\216\010\n\365\007\022\030\n\004lscc\022\020\n\016\n\010tuna-app\022\002\010\001\022\330\007\n\010tuna-app\022\313\007\032`\n\0011\032[{\"vessel\":\"923F\",\"timestamp\":\"150
4054225\",\"location\":\"67.0006, -70.5476\",\"holder\":\"Miriam\"}\032_\n\00210\032Y{\"vessel\":\"49W4\",\"timestamp\":\"1487745091\",\"location\":\"51.9435, 8.2735\",\"holder\":\"Fatima\"}\032^\n\001
2\032Y{\"vessel\":\"M83T\",\"timestamp\":\"1504057825\",\"location\":\"91.2395, -49.4594\",\"holder\":\"Dave\"}\032^\n\0013\032Y{\"vessel\":\"T012\",\"timestamp\":\"1493517025\",\"location\":\"58.0148,
 59.01391\",\"holder\":\"Igor\"}\032_\n\0014\032Z{\"vessel\":\"P490\",\"timestamp\":\"1496105425\",\"location\":\"-45.0945, 0.7949\",\"holder\":\"Amalea\"}\032_\n\0015\032Z{\"vessel\":\"S439\",\"timest
amp\":\"1493512301\",\"location\":\"-107.6043, 19.5003\",\"holder\":\"Rafa\"}\032`\n\0016\032[{\"vessel\":\"J205\",\"timestamp\":\"1494117101\",\"location\":\"-155.2304, -15.8723\",\"holder\":\"Shen\"}
\032_\n\0017\032Z{\"vessel\":\"S22L\",\"timestamp\":\"1496104301\",\"location\":\"103.8842, 22.1277\",\"holder\":\"Leila\"}\032`\n\0018\032[{\"vessel\":\"EI89\",\"timestamp\":\"1485066691\",\"location\
":\"-132.3207, -34.0983\",\"holder\":\"Yuan\"}\032_\n\0019\032Z{\"vessel\":\"129R\",\"timestamp\":\"1485153091\",\"location\":\"153.0054, 12.6429\",\"holder\":\"Carlo\"}\032\003\010\310\001\"\017\022\0
10tuna-app\032\0031.0" endorsement:<endorser:"\n\007Org1MSP\022\226\006-----BEGIN CERTIFICATE-----\nMIICGjCCAcCgAwIBAgIRAPlwF/rUZUP9mqN4wSml4iswCgYIKoZIzj0EAwIwczEL\nMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGl
mb3JuaWExFjAUBgNVBAcTDVNhbiBG\ncmFuY2lzY28xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMTE2Nh\nLm9yZzEuZXhhbXBsZS5jb20wHhcNMTcwODMxMDkxNDMyWhcNMjcwODI5MDkxNDMy\nWjBbMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2
FsaWZvcm5pYTEWMBQGA1UEBxMN\nU2FuIEZyYW5jaXNjbzEfMB0GA1UEAxMWcGVlcjAub3JnMS5leGFtcGxlLmNvbTBZ\nMBMGByqGSM49AgEGCCqGSM49AwEHA0IABHihxW6ks3B2+5XdbAVq3CBgxRRRZ22x\nzzpqnD86nKkz7fBElBuhlXl2K6rTxyY2OBOB0ts8k
eqZ93xueRGymrajTTBLMA4G\nA1UdDwEB/wQEAwIHgDAMBgNVHRMBAf8EAjAAMCsGA1UdIwQkMCKAIEI5qg3Ndtru\nuLoM2nAYUdFFBNMarRst3dusalc2Xkl8MAoGCCqGSM49BAMCA0gAMEUCIQD4j0Rn\ne1rrd0FSCzsR6u+IuuPK5dI/kR/bh7+VLf0TNgIgCfUt
kJvfvzVEwZLFoFyjoHtr\ntvwzNUS1U0hEqIaDeo4=\n-----END CERTIFICATE-----\n" signature:"0D\002 \021\21669\360ck\356\025\215\350\017\204\334\361\201\255\315j\2658\033M\343H_\322\375\211\244\304\301\002 P\25
5\270\304\016r\321\240\275d\305\342\036%\202C\340\374\242~\016\374\300+ Y\257\361\310J\235\333" >
2018-11-09 16:38:53.305 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 04f Chaincode invoke successful. result: status:200

Total execution time : 71 secs ...


Start with the registerAdmin.js, then registerUser.js, then server.js
```

2018-11-09 16:38:53.305 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 04f Chaincode invoke successful. result: status:200

Tuna Application – 3) RegisterAdmin

$ node registerAdmin.js ← ***필요할 경유 sudo 사용***

- 기본적으로 관리자가 이미 등록되어 있는 경우 CA 클라이언트로 등록하지 않은 경우 확인하는 Fabric CA 클라이언트와 비교하며, enrollmentID, Secret과 같은 기본 매개 변수가 설정
- 그런 다음 사용자는 공용 개인 키 쌍과 함께 구성원 ID가있는 FabricClient로 생성

```
// need to enroll it with CA server
return fabric_ca_client.enroll({
  enrollmentID: 'admin',
  enrollmentSecret: 'adminpw'
}).then((enrollment) => {
  console.log('Successfully enrolled admin user "admin"');
  return fabric_client.createUser(
      {username: 'admin',
          mspid: 'Org1MSP',
          cryptoContent: { privateKeyPEM: enrollment.key.toBytes(), signedCertPEM: enrollment.certificate }
      });
}).then((user) => {
  admin_user = user;
  return fabric_client.setUserContext(admin_user);
}).catch((err) => {
  console.error('Failed to enroll and persist admin. Error: ' + err.stack ? err.stack : err);
  throw new Error('Failed to enroll admin');
});
```

Tuna Application – 4) RegisterUser

$ node registerUser.js ← ***필요할 경유 sudo 사용***

```
bcadmin@ubuntu:~/education/LFS171x/fabric-material/tuna-app$ sudo node registerUser.js
 Store path:/home/bcadmin/.hfc-key-store
Successfully loaded admin from persistence
Successfully registered user1 - secret:uzyeGYPIYDLc
Successfully enrolled member user "user1"
User1 was successfully registered and enrolled and is ready to intreact with the fabric network
bcadmin@ubuntu:~/education/LFS171x/fabric-material/tuna-app$
```

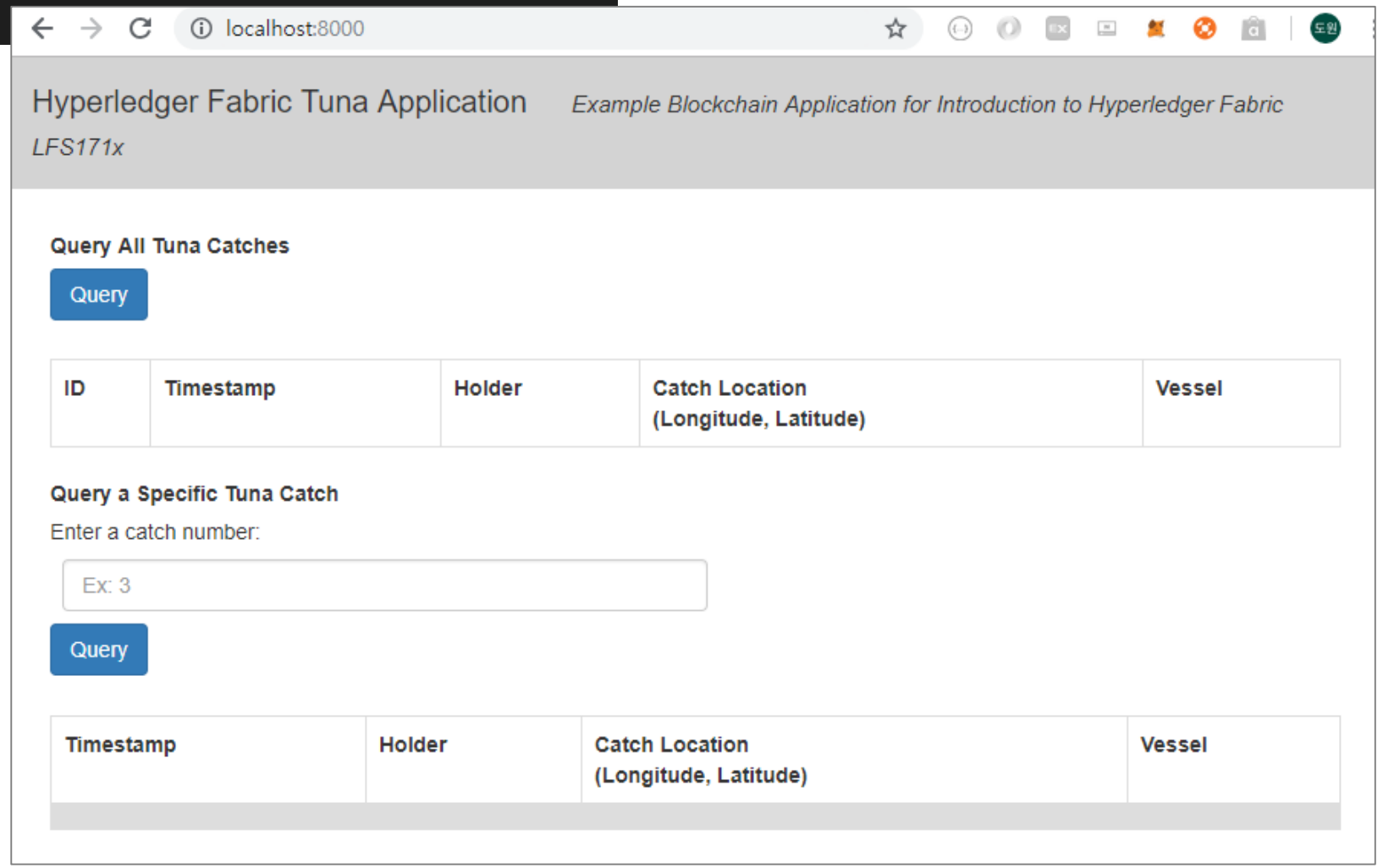
- ***기존에 생성된 User때문에 오류 발생할 경우 ~/.hfc-key-store/* 삭제 후***

  → registerAdmin.js, registerUser.js 실행

Tuna Application – 5) Run the server

$ node server.js

```
bcadmin@ubuntu:~/education/LFS171x/fabric-material/tuna-app$ node server.js
Live on port: 8000
```



Tuna Application – 5) Test

- Query All Tuna Catches — triggers— ***queryAllTuna.js***
- Query Specific Tuna — triggers — ***queryTuna.js***
- Create Tuna Record — triggers— ***recordTuna.js***
- Change Tuna Holder — triggers — ***changeTunaHolder.js***

- Chaincode 위치

```
bcadmin@ubuntu:~/education/LFS171x/fabric-material/chaincode/tuna-app$ ll
total 16
drwxrwxr-x 2 bcadmin bcadmin 4096 Nov 10 01:34 ./
drwxrwxr-x 4 bcadmin bcadmin 4096 Nov 10 01:38 ../
-rw-rw-r-- 1 bcadmin bcadmin 7296 Nov 10 01:34 tuna-chaincode.go
bcadmin@ubuntu:~/education/LFS171x/fabric-material/chaincode/tuna-app$
```

Tuna Application – 5) queryAllTuna.js

Query All Tuna Catches

Query

ID	Timestamp	Holder	Catch Location (Longitude, Latitude)	Vessel
1	1504054225	Miriam	67.0006, -70.5476	923F
2	1504057825	Dave	91.2395, -49.4594	M83T
3	1493517025	Igor	58.0148, 59.01391	T012
4	1496105425	Amalea	-45.0945, 0.7949	P490
5	1493512301	Rafa	-107.6043, 19.5003	S439
6	1494117101	Shen	-155.2304, -15.8723	J205
7	1496104301	Leila	103.8842, 22.1277	S22L
8	1485066691	Yuan	-132.3207, -34.0983	EI89
9	1485153091	Carlo	153.0054, 12.6429	129R
10	1487745091	Fatima	51.9435, 8.2735	49W4

Tuna Application – 5) queryTuna.js

Query a Specific Tuna Catch

Enter a catch number:

3

Query

Timestamp	Holder	Catch Location (Longitude, Latitude)	Vessel
1493517025	Igor	58.0148, 59.01391	T012

Tuna Application – 5) recordTuna.js

Create Tuna Record

Enter catch id:

11

Enter name of vessel:

0239L

Enter longitude:

28.012

Enter latitude:

150.405

Enter timestamp:

4982342301

Enter name of holder:

Dowon

Create

Query All Tuna Catches

Query

ID	Timestamp	Holder	Catch Location (Longitude, Latitude)	Vessel
1	1504054225	Miriam	67.0006, -70.5476	923F
2	1504057825	Dave	91.2395, -49.4594	M83T
3	1493517025	Igor	58.0148, 59.01391	T012
4	1496105425	Amalea	-45.0945, 0.7949	P490
5	1493512301	Rafa	-107.6043, 19.5003	S439
6	1494117101	Shen	-155.2304, -15.8723	J205
7	1496104301	Leila	103.8842, 22.1277	S22L
8	1485066691	Yuan	-132.3207, -34.0983	EI89
9	1485153091	Carlo	153.0054, 12.6429	129R
10	1487745091	Fatima	51.9435, 8.2735	49W4
11	4982342301	Dowon	28.012, 150.405	0239L

Tuna Application – 5) changeTunaHolder.js

Change Tuna Holder

Enter a catch id between 1 and 10:

10

Enter name of new holder:

Dowon

Change

10	1487745091	Dowon	51.9435, 8.2735	49W4
11	4982342301	Kenneth	28.012, 150.405	0239L