

# 하이퍼레저 패브릭 스마트컨트랙트 개발 실습

2022-02-15

빅픽처랩(주)

안 휘

# 실습 목차

- 개발 환경 구성 (1시간)
  - Go 설치
  - asset-transfer-basic 테스트 구동
- 체인코드 살펴보기 (1시간)
- balance-transfer 개발 (1시간)
  - 사용자 생성
  - 사용자 잔액 확인
  - 송금
- Client Server에서 사용자 생성, 잔액 확인, 송금 호출 (1시간)

# Go 설치

- Go 바이너리 다운로드 및 설치

- `cd ~/`
- `wget https://go.dev/dl/go1.16.14.linux-amd64.tar.gz`
- `sudo tar -xvf go1.16.14.linux-amd64.tar.gz`
- `sudo mv go /usr/local`

- 환경변수 설정

- `vim ~/.bashrc`
- 마지막에 다음 세줄 입력
  - `export GOROOT=/usr/local/go`
  - `export GOPATH=$HOME/go`
  - `export PATH=$GOPATH/bin:$GOROOT/bin:$PATH`

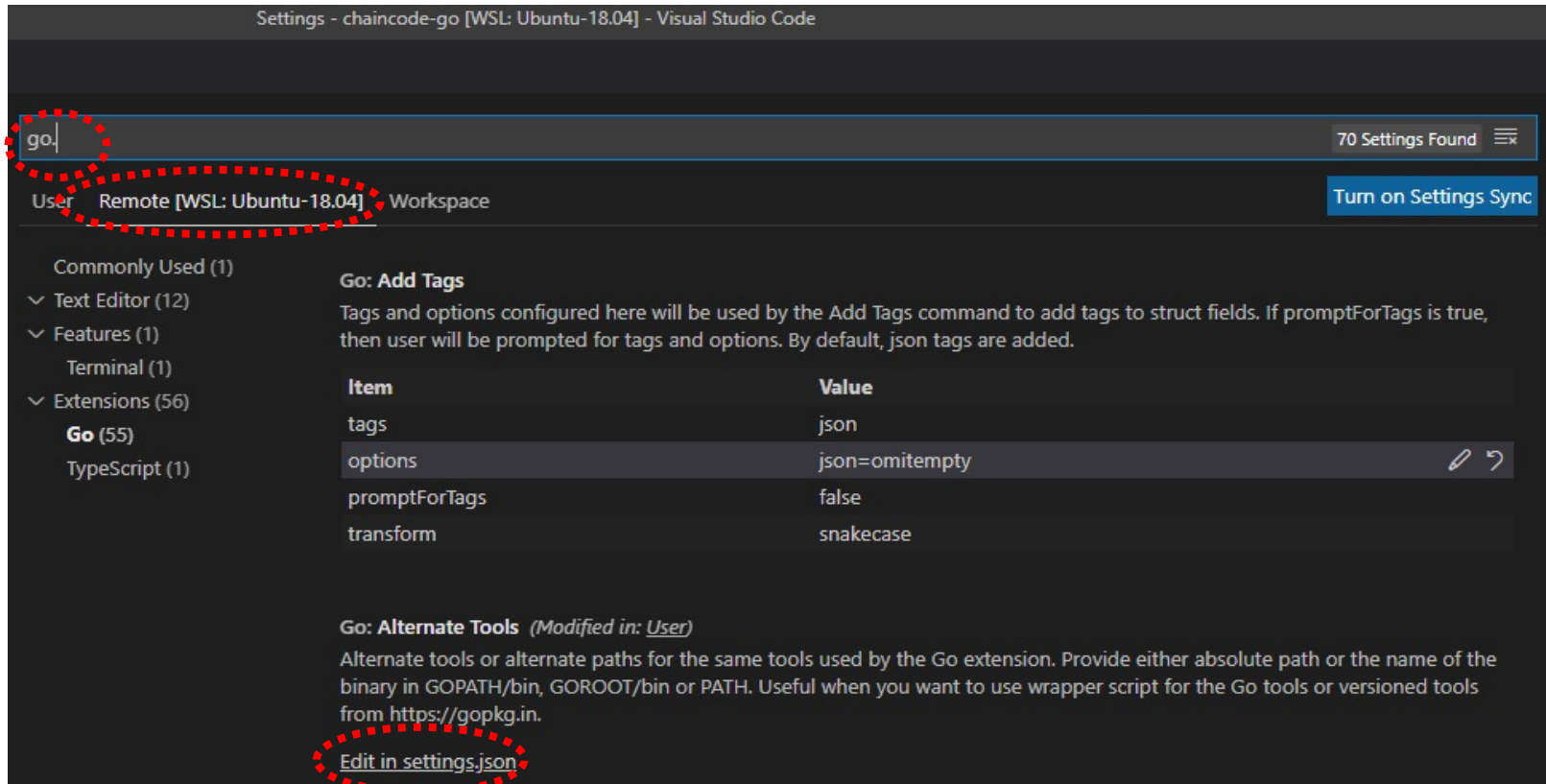
- 설치 확인

- `go version`

# Go 설치

- VSCode 에서 Go 플러그인 설정

- ~/fabric-samples/asset-transfer-basic/chaincode-go 에서 VSCode 오픈
- extension에서 Go 검색
- Install on WSL: Ubuntu-18.04 선택
- VSCode 설정 열기 (Ctrl + ,)



# Go 설치

- VSCode 에서 Go 플러그인 설정

- extension에서 Go 검색
- Install on WSL: Ubuntu-18.04 선택
- VSCode 설정 열기 (Ctrl + ,)
- 설정 JSON 파일에 아래와 같이 입력
  - gopath는 터미널에서 echo \$GOPATH 를 통해 획득

```
{  
  "go.gopath": "/home/byron1st/go",  
  "go.goroot": "/usr/local/go",  
  "go.useLanguageServer": true,  
}
```

- VSCode가 필요한 go 툴을 모두 설치하도록 놔둠
- VSCode 재시작

# asset-transfer-basic 테스트 구동

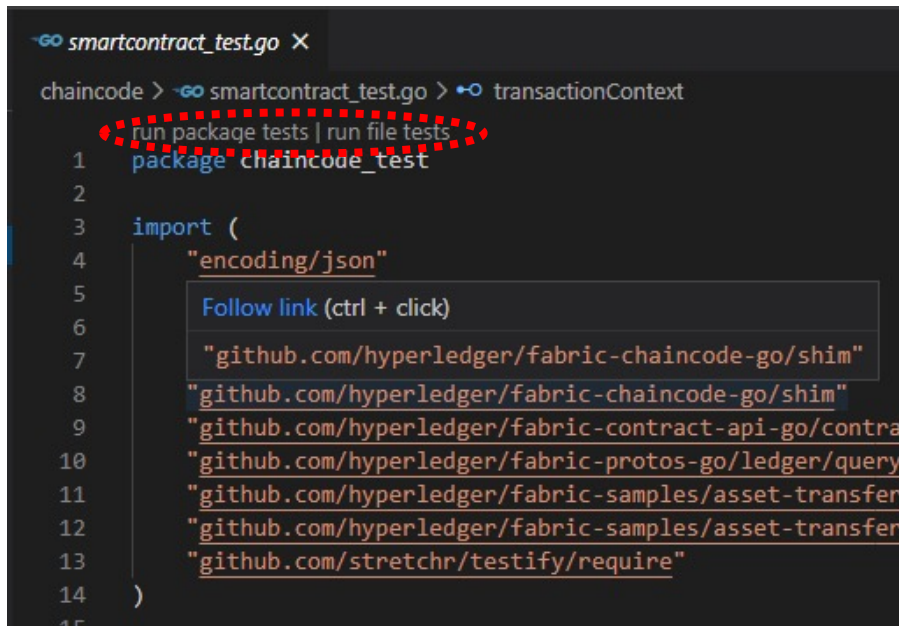
- VSCode 에서 Go 플러그인 설정
  - ~/fabric-samples/asset-transfer-basic/chaincode-go 이동
  - 다음 커맨드 입력
    - `go test github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode -v`

```
byron1st@DESKTOP-D240NBD:~/fabric-samples/asset-transfer-basic/chaincode-go$ go test github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode -v
=== RUN   TestInitLedger
--- PASS: TestInitLedger (0.00s)
=== RUN   TestCreateAsset
--- PASS: TestCreateAsset (0.00s)
=== RUN   TestReadAsset
--- PASS: TestReadAsset (0.00s)
=== RUN   TestUpdateAsset
--- PASS: TestUpdateAsset (0.00s)
=== RUN   TestDeleteAsset
--- PASS: TestDeleteAsset (0.00s)
=== RUN   TestTransferAsset
--- PASS: TestTransferAsset (0.00s)
=== RUN   TestGetAllAssets
--- PASS: TestGetAllAssets (0.00s)
PASS
ok      github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode 0.026s
```

# asset-transfer-basic 테스트 구동

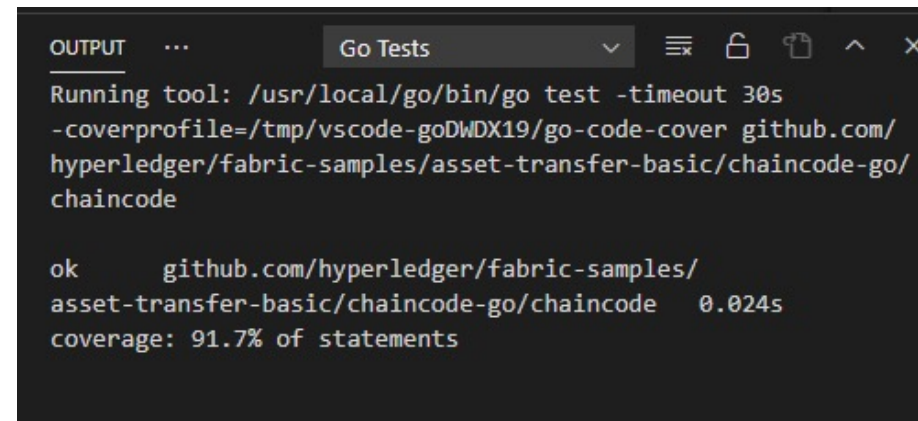
- VSCode 에서 Go 플러그인 설정

- ~/fabric-samples/asset-transfer-basic/chaincode-go를 VSCode에서 오픈
- chaincode/smartcontract\_test.go 파일 선택



```
chaincode > smartcontract_test.go > transactionContext
run package tests | run file tests
package chaincode_test

1
2
3 import (
4     "encoding/json"
5     "github.com/hyperledger/fabric-chaincode-go/shim"
6     "github.com/hyperledger/fabric-chaincode-go/shim"
7     "github.com/hyperledger/fabric-chaincode-go/shim"
8     "github.com/hyperledger/fabric-contract-api-go/contractapi"
9     "github.com/hyperledger/fabric-protos-go/ledger/querypattern"
10    "github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode"
11    "github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode"
12    "github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode"
13    "github.com/stretchr/testify/require"
14 )
15
```



```
OUTPUT ... Go Tests
Running tool: /usr/local/go/bin/go test -timeout 30s
-coverprofile=/tmp/vscode-goDWDX19/go-code-cover github.com/
hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/
chaincode

ok      github.com/hyperledger/fabric-samples/
asset-transfer-basic/chaincode-go/chaincode  0.024s
coverage: 91.7% of statements
```

# 체인코드 살펴보기

- Go 언어에서의 method

- Struct 와 함수 연결
- 입력, 반환 파라미터 표시 방법

```
// SmartContract provides functions for managing an Asset
type SmartContract struct {
    contractapi.Contract
}

...
func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
    ...
}

func (s *SmartContract) ReadAsset(ctx contractapi.TransactionContextInterface, id string) (*
Asset, error)

...
}
```



# 체인코드 살펴보기

- **ctx.getStub()**
  - shim.ChaincodeStubInterface 객체를 반환
    - GetState
    - PutState
      - []byte 타입은 객체를 json.marshal 하여 얻음
    - DelState
    - ...
- **json.marshal / unmarshal**
- **golang tag**

# balance-transfer 개발

- 저장할 모델 정의

- 사용자 ID, 잔액 정보 정도면 충분할 듯

- 함수 3개 만들기

- createUser
  - user ID의 중복 여부 확인
  - 만들면 자동으로 기본 돈 100원 지급
- balance
  - userID 가 없으면 에러
  - 있으면 잔액 반환
- transferMoney
  - senderID, recipientID, value를 받아서 처리
  - 돈이 부족하면 에러 반환

## peer lifecycle chaincode querycommitted

```
Query the committed chaincode definitions by channel on a p

Usage:
  peer lifecycle chaincode querycommitted [flags]

Flags:
  -C, --channelID string          The channel on which to query
  --connectionProfile string      The fully qualified path to the
  -h, --help                      help for querycommitted
  -n, --name string              Name of the chaincode
  -O, --output string            The output format (json, text)
  --peerAddresses stringArray    The addresses of the peers to
  --tlsRootCertFiles stringArray If TLS is enabled, the root
```

`./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go -ccv 2.0 -ccs 2`

# Client Server에서 사용자 생성, 잔액 확인, 송금 호출

- **asset-transfer-basic/application-gateway-javascript**
  - test-network 는 - ca 모드로 실행되어야 함
    - ./network.sh up -ca
    - ./network.sh createChannel
    - ./network.sh deployCC
  - wallet 폴더의 키들은 실행 전에 삭제되어야 함
    - 해당 키 들은 예전에 생성했던 test-network의 키들임