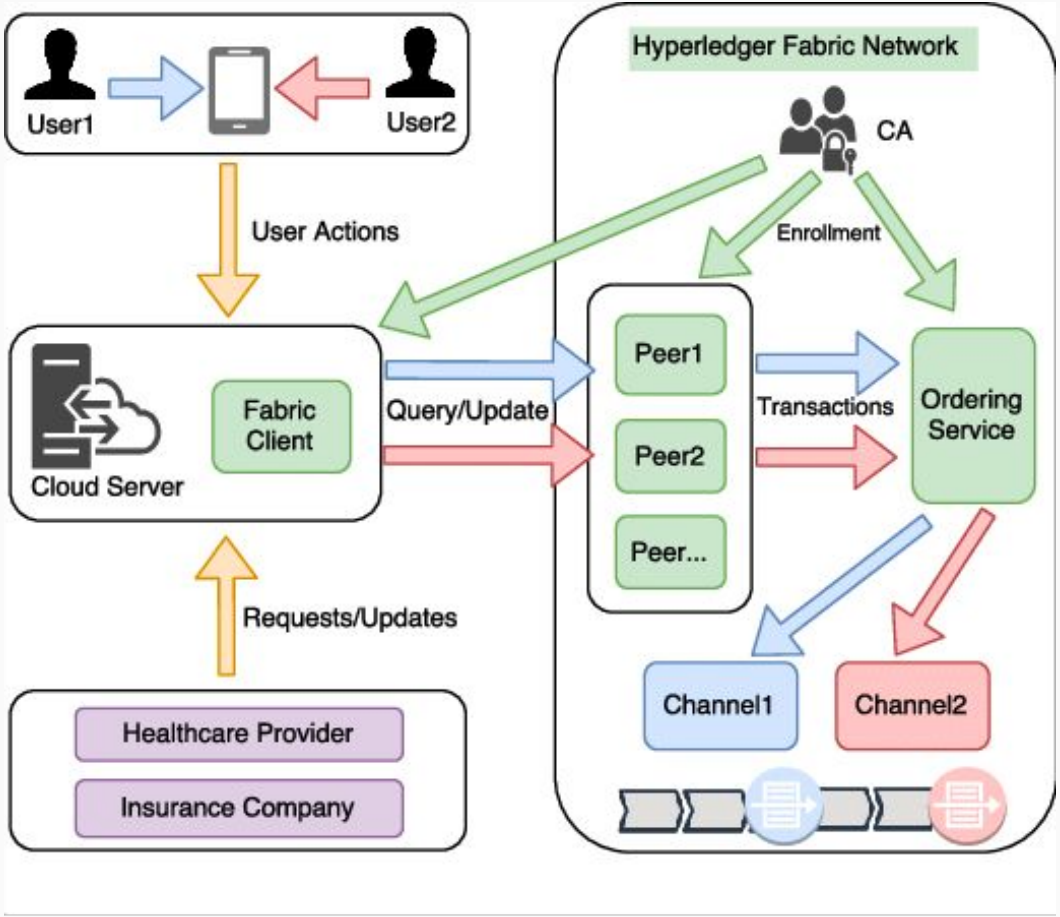
Hyperledger #2
Forming the business models, transactions

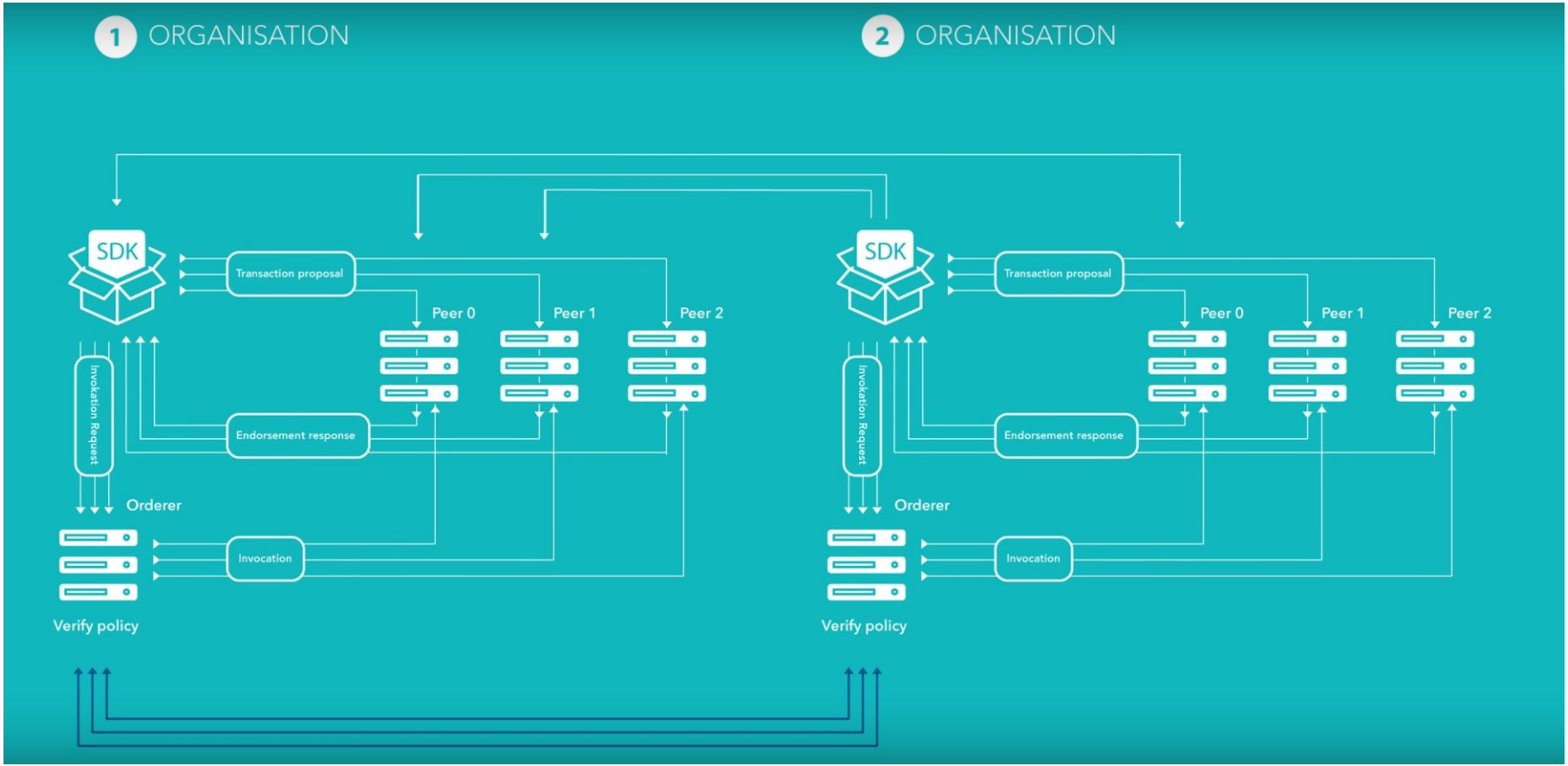
Trung

Growth Session #17 - August 23-24 2018

Revision Time



Revision Time



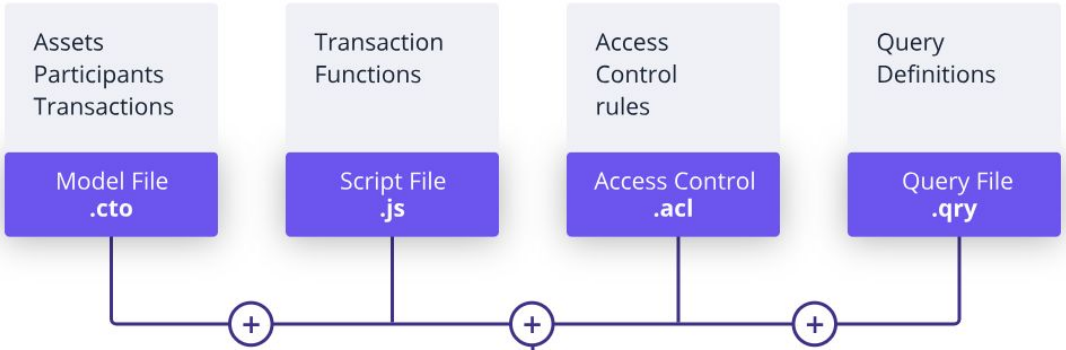
The setup

- CA, Orderer and Peers are up for development (dockerised).
- Modeling and start writing chaincode?

The struggles

```
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterfac
   function, args := stub.GetFunctionAndParameters()
   fmt.Println("invoke is running " + function)
   // Handle different functions
   if function == "initMarble" { //create a new marble
      return t.initMarble(stub, args)
   } else if function == "transferMarble" { //change owner of a
      return t.transferMarble(stub, args)
   } else if function == "transferMarblesBasedOnColor" { //trans
      return t.transferMarblesBasedOnColor(stub, args)
   } else if function == "delete" { //delete a marble
      return t.delete(stub, args)
   } else if function == "readMarble" { //read a marble
      return t.readMarble(stub, args)
   } else if function == "queryMarblesByOwner" { //find marbles
      return t.queryMarblesByOwner(stub, args)
   } else if function == "queryMarbles" { //find marbles based c
      return t.queryMarbles(stub, args)
   } else if function == "getHistoryForMarble" { //get history c
      return t.getHistoryForMarble(stub, args)
   } else if function == "getMarblesByRange" { //get marbles bas
      return t.getMarblesByRange(stub, args)
   }

   fmt.Println("invoke did not find func: " + function) //error
   return shim.Error("Received unknown function invocation")
}
```

Introducing Hyperledger Composer

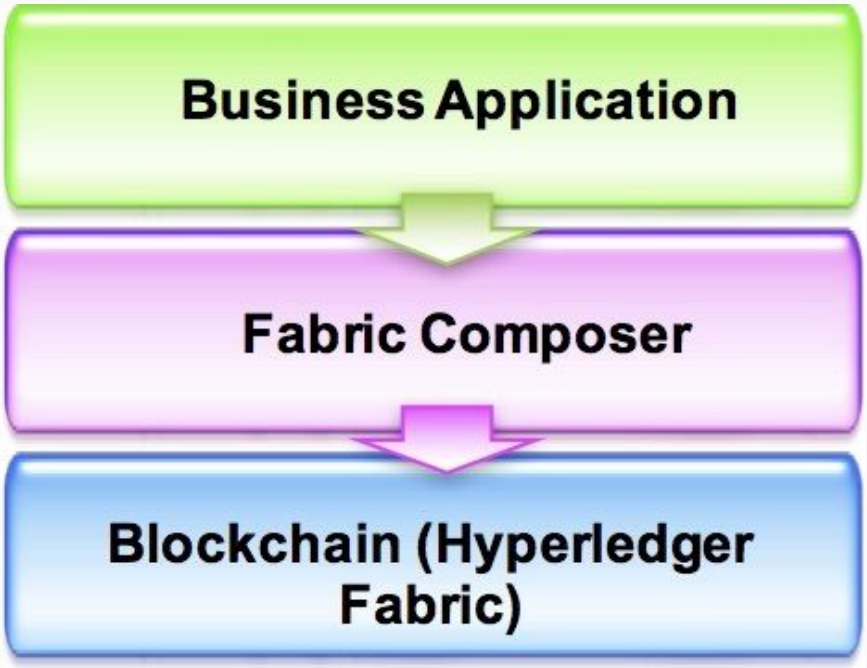


Use Composer to create a Business Network Definition, comprised of Model (.cto), Script (.js), ACL (.acl) and Query (.qry) files.

Package up your Business Network Definition and export it as an archive (.bna file) ready to deploy it somewhere.

Use ID Cards (which include connection profiles and credentials) to deploy your Business Network Definition to a distributed ledger.

What is Hyperledger Composer



- HC is a set of abstractions, tools and APIs to model, build, integrate and deploy a blockchain solution.
- Run on top of HyperLedger Fabric
- More intuitive in the way we approach the network, making the transactions

Comparison

```
type marble struct {
    ObjectType string `json:"docType"` //docType is used to distir
    Name       string `json:"name"`    //the fieldtags are needed
    Color      string `json:"color"`
    Size       int    `json:"size"`
    Owner      string `json:"owner"`
}
```

```
enum MarbleSize {
    o SMALL
    o MEDIUM
    o LARGE
}

asset Marble identified by marbleId {
    o String marbleId
    o MarbleSize size
    o MarbleColor color
    --> Player owner
}

participant Player identified by email {
    o String email
    o String firstName
    o String lastName
}
```

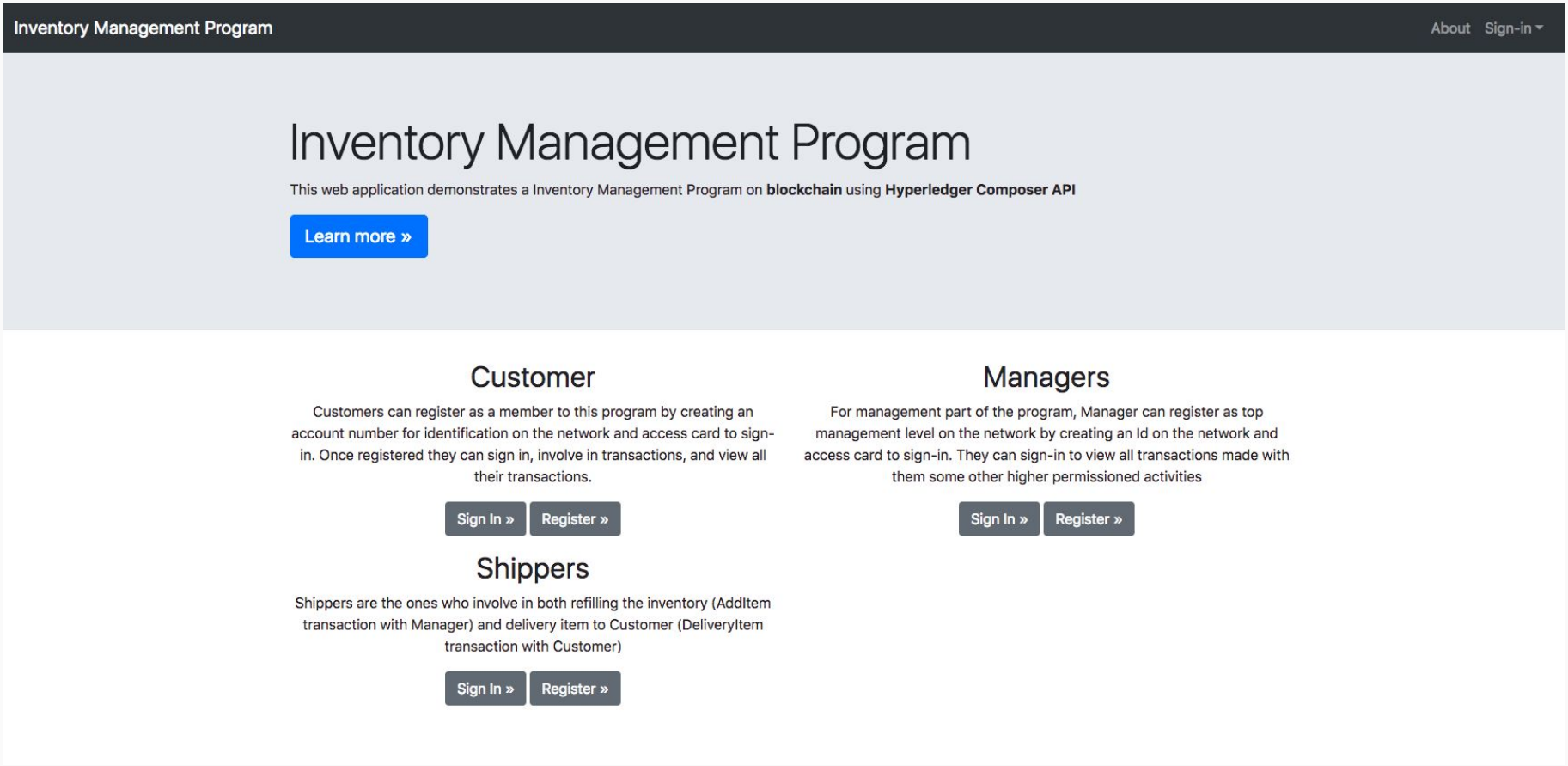
Comparison - in chaincode

```
func (t *SimpleChaincode) transferMarble(stub shim.ChaincodeStubInterface, args []string) pb.Response {

	//   0       1
	// "name", "bob"
	if len(args) < 2 {
		return shim.Error("Incorrect number of arguments. Expecting 2")
	}

	marbleName := args[0]
	newOwner := strings.ToLower(args[1])
	fmt.Println("- start transferMarble ", marbleName, newOwner)

	marbleAsBytes, err := stub.GetState(marbleName)
	if err != nil {
		return shim.Error("Failed to get marble:" + err.Error())
	} else if marbleAsBytes == nil {
		return shim.Error("Marble does not exist")
	}

	marbleToTransfer := marble{}
	err = json.Unmarshal(marbleAsBytes, &marbleToTransfer) //unmarshal it aka JSON.parse()
	if err != nil {
		return shim.Error(err.Error())
	}
	marbleToTransfer.Owner = newOwner //change the owner

	marbleJSONasBytes, _ := json.Marshal(marbleToTransfer)
	err = stub.PutState(marbleName, marbleJSONasBytes) //rewrite the marble
	if err != nil {
		return shim.Error(err.Error())
	}

	fmt.Println("- end transferMarble (success)")
	return shim.Success(nil)
}
```

Comparison - from Composer

```
function tradeMarble(tradeMarble) {
    tradeMarble.marble.owner = tradeMarble.newOwner;
    return getAssetRegistry('org.hyperledger_composer.marbles.Mark
    .then(function (assetRegistry) {
        return assetRegistry.update(tradeMarble.marble);
    });
}
```

Demo


Inventory Management Program
About
Sign-in
Inventory Management Program
This web application demonstrates a Inventory Management Program on blockchain using Hyperledger Composer API
Learn more »
Customer
Customers can register as a member to this program by creating an account number for identification on the network and access card to sign-in. Once registered they can sign in, involve in transactions, and view all their transactions.
Sign In »
Register »
Managers
For management part of the program, Manager can register as top management level on the network by creating an Id on the network and access card to sign-in. They can sign-in to view all transactions made with them some other higher permissioned activities
Sign In »
Register »
Shippers
Shippers are the ones who involve in both refilling the inventory (AddItem transaction with Manager) and delivery item to Customer (DeliveryItem transaction with Customer)
Sign In »
Register »

I believe Hyperledger Composer is the right choice to get started with blockchain development.

It allows them to focus on the business logic, and avoids a lot of error-prone boilerplate code.

Thanks!

Contact Nimbl3

hello@nimbl3.com

399 Sukhumvit Road, Interchange 21
Klongtoey nua, Wattana
Bangkok 10110

28C Stanley St,
Singapore 068737

20th Floor, Central Tower
28 Queen's Road
Central, Hong Kong

nimbl3.com