Steps to Develop and Deploy a Simple Chaincode on Hyperledger Fabric on Windows

Developing and deploying a simple chaincode (smart contract) on Hyperledger Fabric involves several steps, including setting up a local Hyperledger Fabric environment, writing the chaincode, and deploying it on the network. Here's a complete step-by-step guide to help you through the process on Windows.

Overview:

• Hyperledger Fabric: An enterprise-grade, permissioned blockchain framework.
• Chaincode: Fabric's version of smart contracts, typically written in Go, Node.js, or Java.

Steps to Develop and Deploy Chaincode on Hyperledger Fabric:

1. Set Up Hyperledger Fabric Environment

To start, you need to have a running Hyperledger Fabric network. This can be done locally using Docker and Fabric samples.

Prerequisites:

• Docker & Docker Compose: To manage containerized services.
• Go, Node.js, or Java: Depending on the language you want to use for writing chaincode.
• Hyperledger Fabric Samples: A collection of pre-configured networks and example chaincodes.

Install Prerequisites:

1. **Install Docker**: Download and install Docker Desktop for Windows from the official Docker website (https://www.docker.com/get-started).
2. **Install Go**: Follow the instructions at Go's official site (https://golang.org/dl/).
3. **Install Hyperledger Fabric**: Download the Hyperledger Fabric binaries and samples:

curl -sSL https://bit.ly/2ysbOFE | bash -s

Starting the Fabric Network:

1. Navigate to the `test-network` directory:

cd fabric-samples/test-network

2. Bring up the test network:

./network.sh up

3. Create a channel:

./network.sh createChannel

4. Deploy the chaincode using the `network.sh` script or custom deployment scripts.

2. Write Simple Chaincode

Here's an example of a simple chaincode in Go that stores and retrieves key-value pairs.

1. Create a new directory for your chaincode:

mkdir -p fabric-samples/chaincode/simple-go

2. Create a new file called `simple.go` with the following content:

```
package main

import (
  "fmt"
  "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

type SimpleContract struct {
  contractapi.Contract
}

func (s *SimpleContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
  return nil
}

func (s *SimpleContract) Put(ctx contractapi.TransactionContextInterface, key string, value
```

```
string) error {
    return ctx.GetStub().PutState(key, []byte(value))
}

func (s *SimpleContract) Get(ctx contractapi.TransactionContextInterface, key string)
(string, error) {
    value, err := ctx.GetStub().GetState(key)
    if err != nil {
        return "", err
    }
    if value == nil {
        return "", fmt.Errorf("No value found for key: %s", key)
    }
    return string(value), nil
}

func main() {
    chaincode, err := contractapi.NewChaincode(&SimpleContract{})
    if err != nil {
        fmt.Printf("Error creating SimpleContract chaincode: %s", err.Error())
        return
    }
    if err := chaincode.Start(); err != nil {
        fmt.Printf("Error starting SimpleContract chaincode: %s", err.Error())
    }
}
```

3. Package and Install Chaincode

Once the chaincode is written, package and deploy it to the network using the following command:

```
./network.sh deployCC -ccn simple-go -ccp ../chaincode/simple-go -ccl go
```

4. Invoke Chaincode Functions

After deploying the chaincode, use the Fabric CLI to interact with it.

1. To store a key-value pair on the ledger:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls \
--cafile $ORDERER_CA -C mychannel -n simple-go -c '{"function":"Put","Args":["myKey",
```

"myValue"]}'

2. To retrieve the stored value:

peer chaincode query -C mychannel -n simple-go -c '{"Args":["Get", "myKey"]}'

5. Interacting with Hyperledger Fabric Using a REST API (Optional)

You can also use the Hyperledger Fabric SDK to build a web application (e.g., using Node.js) that interacts with the blockchain by invoking chaincode functions via REST API calls.

6. Shutting Down the Network

Once you're done testing and deploying the chaincode, shut down the network by running:

./network.sh down

Conclusion:

This guide provided a step-by-step overview of how to develop and deploy a simple chaincode on Hyperledger Fabric on Windows. For production environments, additional factors such as security, identity management, and persistence need to be considered.