Contents

[Synopsis: 2](#_Toc179742668)

[Problem: 2](#_Toc179742669)

[Solution: 2](#_Toc179742670)

[Future Enhancements: 2](#_Toc179742671)

[Chaincode and network setting: 3](#_Toc179742672)

[ Create chaincode in eclipse: 3](#_Toc179742673)

[ Build project 3](#_Toc179742674)

[ Project setting 3](#_Toc179742675)

[Package 6](#_Toc179742676)

[Install : 8](#_Toc179742677)

[Approve 9](#_Toc179742678)

[Commit 9](#_Toc179742679)

[Invoke/ Execution chaincode: 10](#_Toc179742680)

[Initiate the chaincode 10](#_Toc179742681)

[Create parcel 10](#_Toc179742682)

[Query the created parcel 10](#_Toc179742683)

[Update the parcel status: 10](#_Toc179742684)

[Query the new status: 11](#_Toc179742685)

[Failure case (Query not existed parcel): 11](#_Toc179742686)

# Synopsis:

The project aims to revolutionize the courier supply chain management system by introducing a blockchain-based solution using **Hyperledger Fabric**. It seeks to address challenges like lack of transparency, inefficient tracking, and manual processes that slow down operations and increase errors. The system will ensure transparent, secure, and efficient management of parcels by automating manual tasks and providing real-time tracking capabilities, while maintaining an immutable audit trail.

# Problem:

Current courier logistics operations face numerous inefficiencies:

1. **Lack of transparency**: Inability to track parcels in real-time, leading to uncertainty for both couriers and customers.
2. **Manual processes**: Dependence on paper-based systems causes delays and errors, complicating parcel management.
3. **Audit challenges**: Lack of a reliable system for securely maintaining a transparent audit trail of parcel movements and transactions.

# Solution:

The project proposes the creation of a blockchain-based Courier Supply Chain Management System. Using Hyperledger Fabric, this system will:

Enhance transparency by providing real-time parcel tracking through a secure, tamper-proof ledger.

Automate operations, such as adding new parcels and updating their status, thereby reducing paperwork and errors.

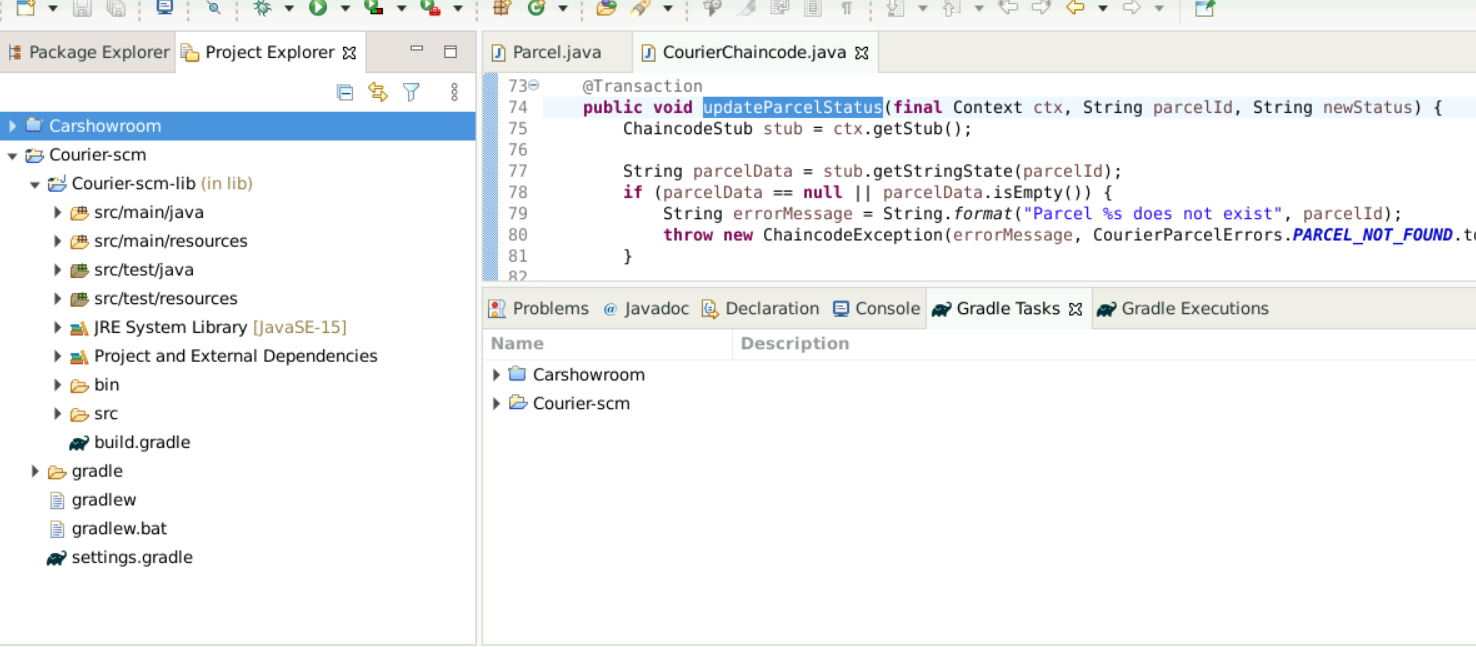
Secure data through blockchain, ensuring tamper-proof records and allowing efficient auditing of all transactions and status updates.

# Future Enhancements:

1. **Integration with IoT devices**: Use IoT sensors to automatically update parcel status based on its location or environmental conditions (e.g., temperature, humidity).
2. **Scalability for multi-carrier support**: Expand the system to accommodate multiple courier companies, creating a unified platform for supply chain management.
3. **AI-based route optimization**: Incorporate AI algorithms to suggest optimized delivery routes based on traffic, parcel priority, and delivery constraints.
4. **Mobile application**: Develop a user-friendly mobile interface for customers and couriers to interact with the system, enhancing user experience.

# Chaincode and network setting:

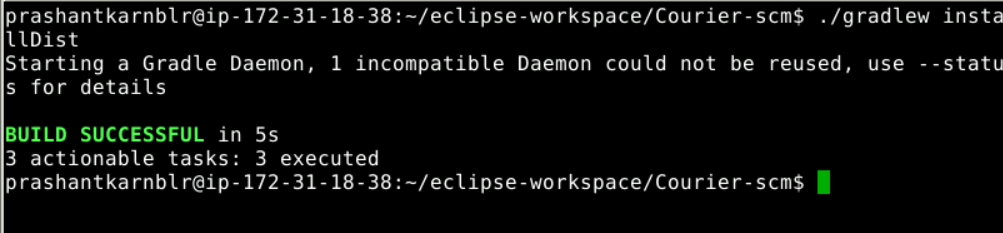
## Create chaincode in eclipse:



## Build project

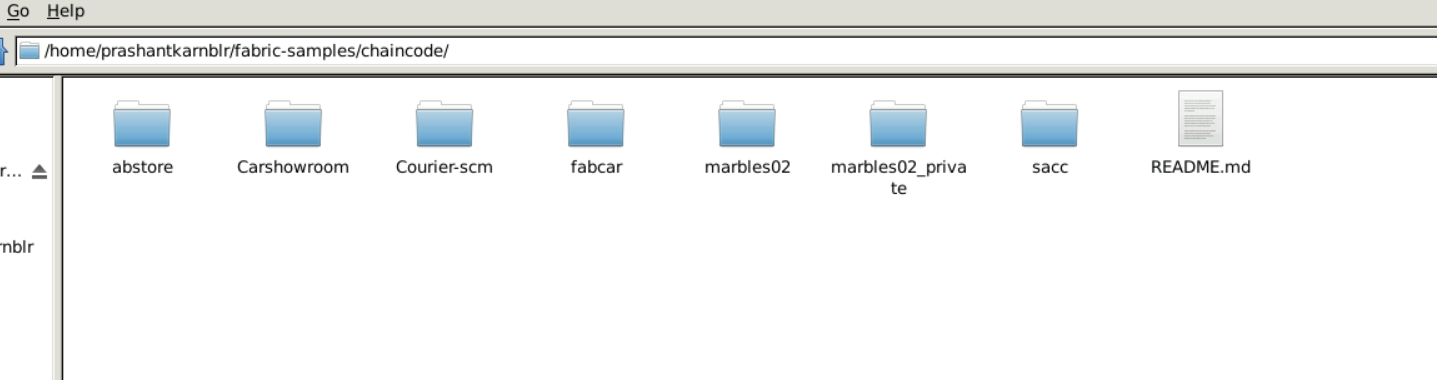
Execute the command from project root :

**./gradlew installDist**

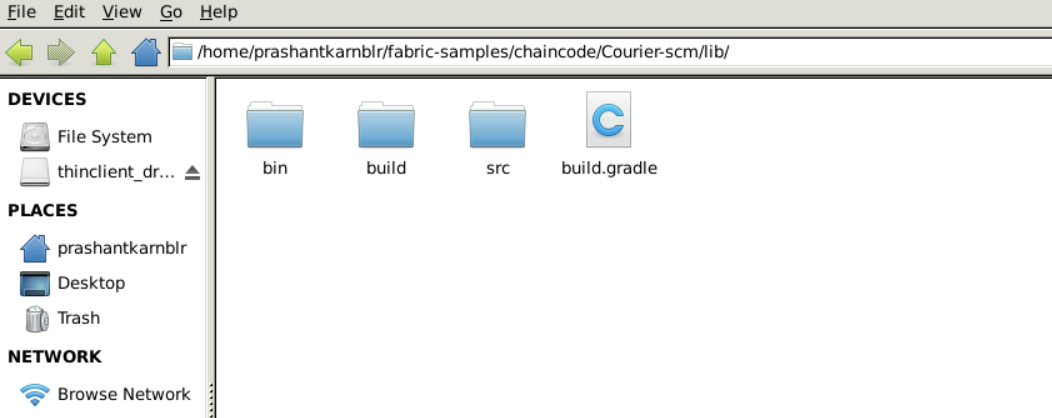


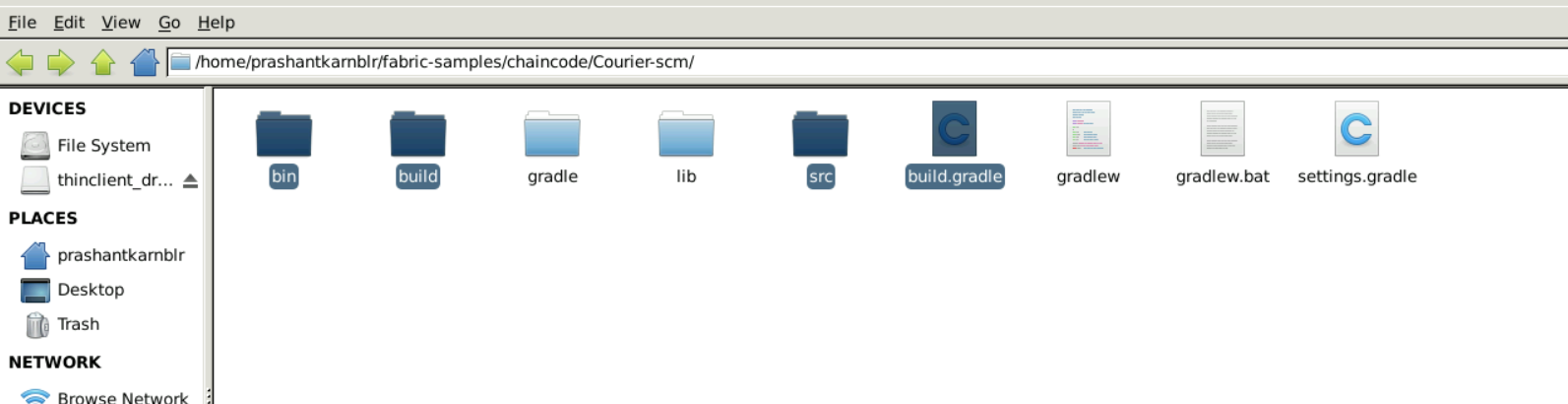
## Project setting

Copy the Courier-scm project into fabric-samples/chaincode

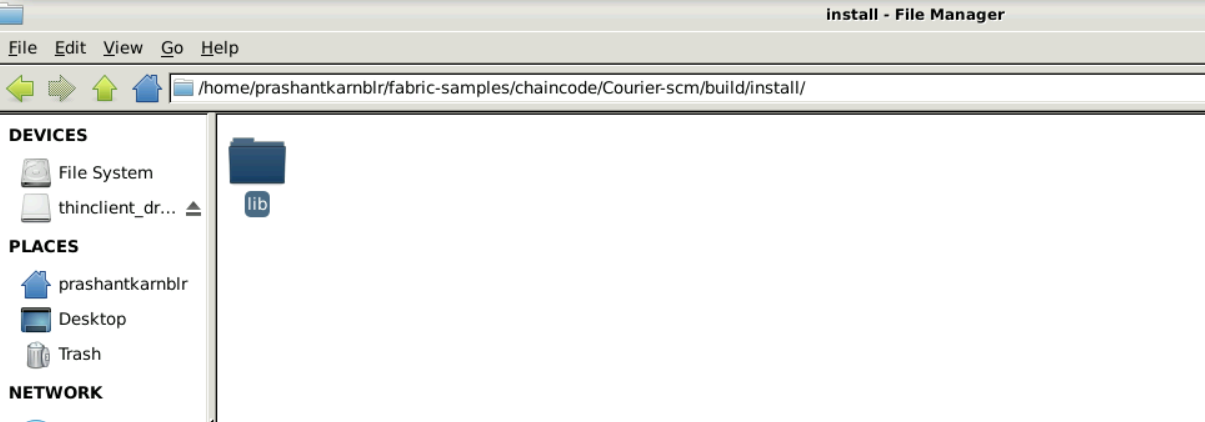


* Go inside lib folder of the project and copies the 4 files and folders and paste into fabric-samples/chaincode/Courier-scm



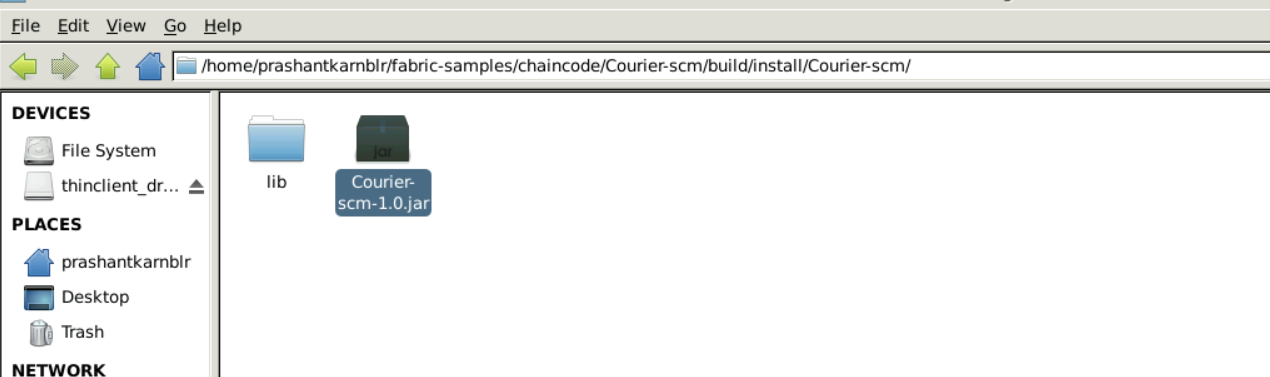


* Navigate to fabric-samples/chaincode/Courier-scm/build/install and change lib folder name to project name **Courier-scm**





* Navigate to fabric-samples/chaincode/Courier-scm/build/install/Courier-scm and change lib-1.0.jar to **Courier-scm-1.0.jar**



**Setting up the Hyperledger Fabric Test network**

1. Navigate to fabric-samples/test-networks:

***cd fabric-samples/test-network***

1. Stop the previously running test network by running the following command:

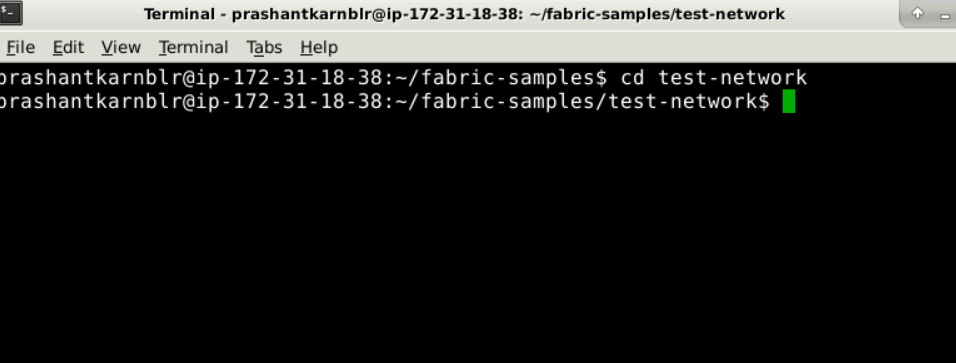
***sudo ./network.sh down***

1. Remove the unused docker images by running the following command:

***sudo docker system prune***

1. Start the test network by executing the following command:

***sudo ./network.sh up -ca -s couchdb***

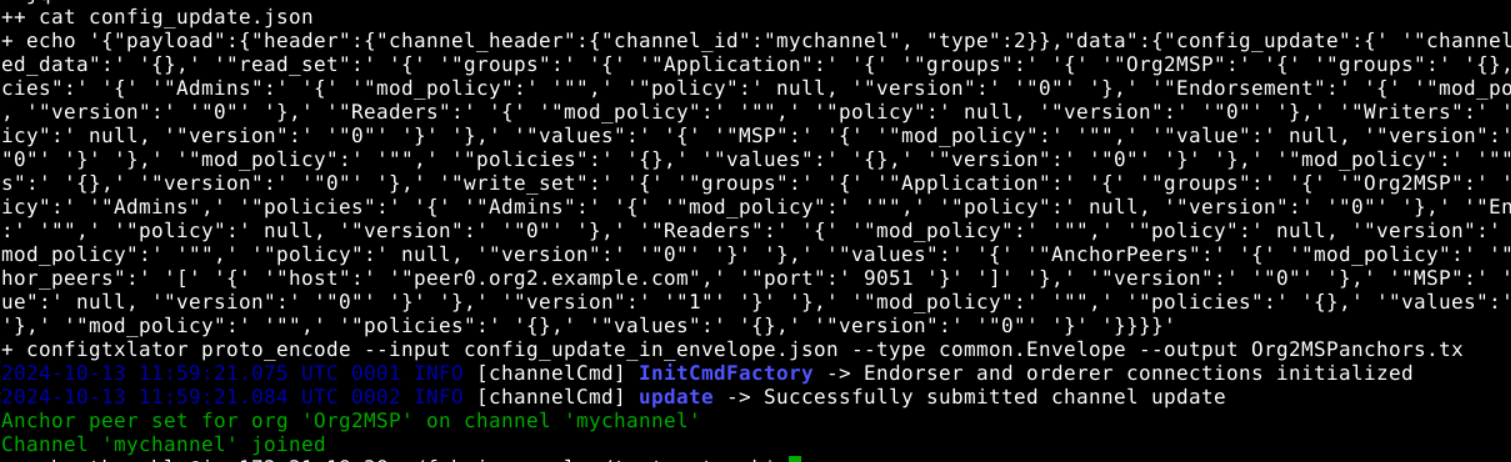


1. To create a communication channel for the peers in the test network, use the following command:

***sudo ./network.sh createChannel -c mychannel***

Create channel:





# Package

Navigate to the **fabric-samples/test-network** folder and create **lifecycle\_setup.sh** life using the following command:

*nano lifecycle\_setup\_org1.sh*

**Add this script:**

***#!/bin/sh***

***export PATH=${PWD}/../bin:${PWD}:$PATH***

***export FABRIC\_CFG\_PATH=$PWD/../config/***

***export CORE\_PEER\_TLS\_ENABLED=true***

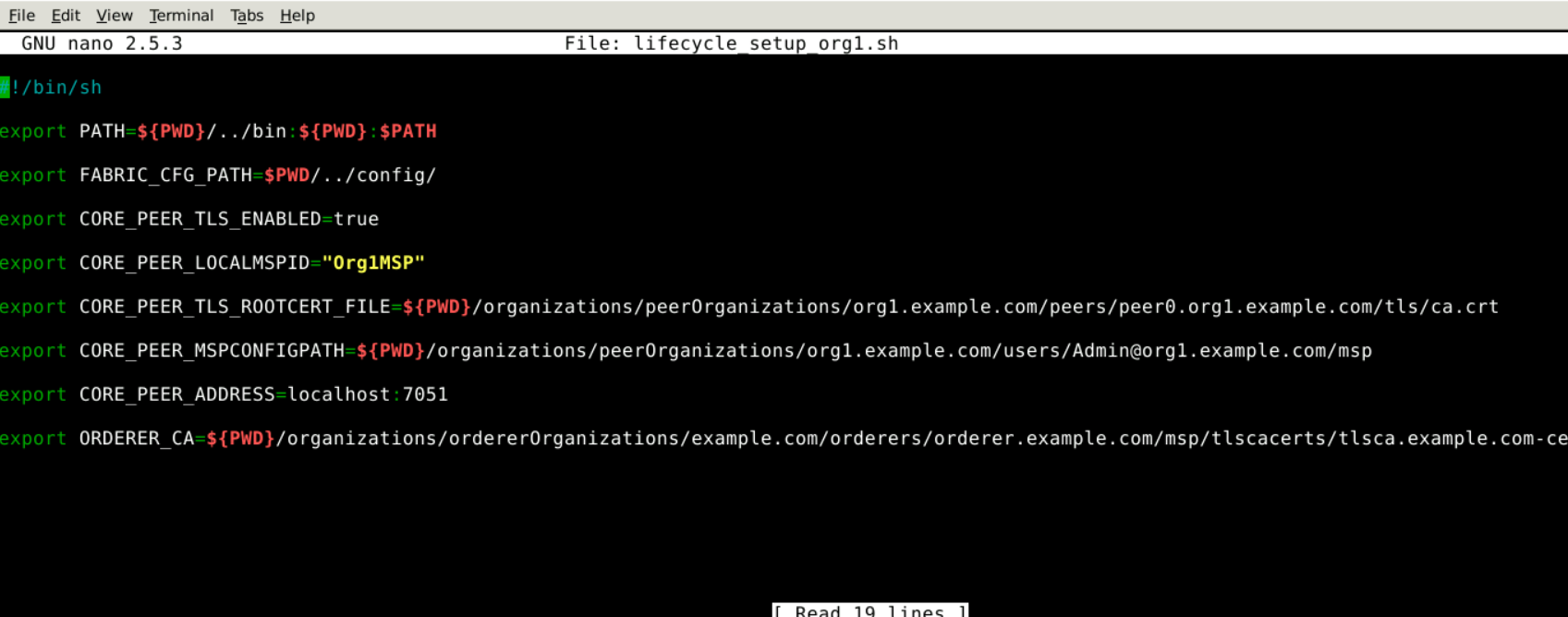
***export CORE\_PEER\_LOCALMSPID="Org1MSP"***

***export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt***

***export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp***

***export CORE\_PEER\_ADDRESS=localhost:7051***

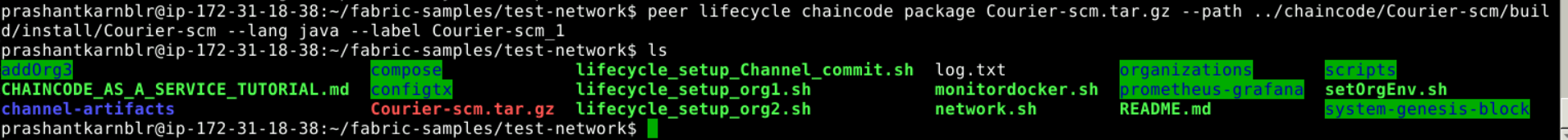
***export ORDERER\_CA=${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem***



Ctrl + X 🡪press Y

Execute below command to create package:

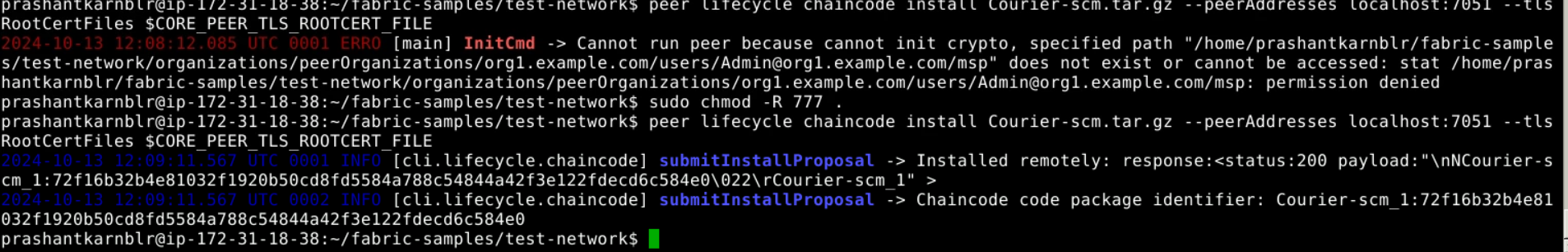
*peer lifecycle chaincode package Courier-scm.tar.gz --path ../chaincode/Courier-scm/build/install/Courier-scm --lang java --label Courier-scm\_1*



# Install :

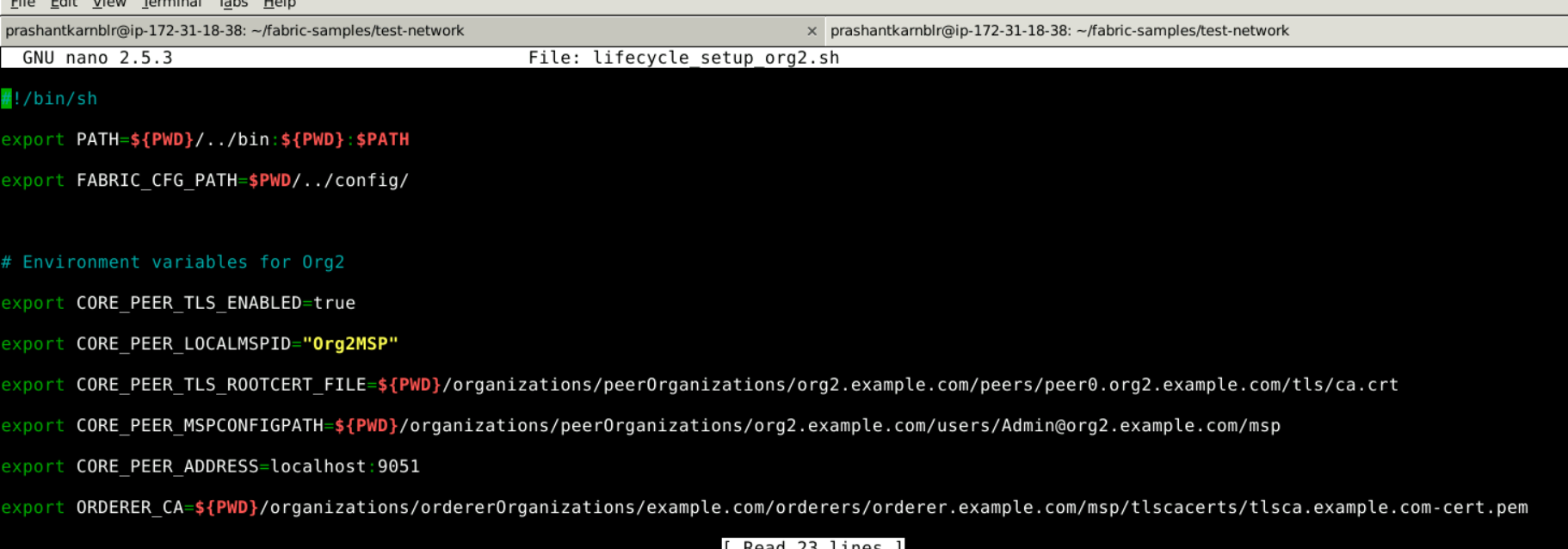
To install the packaged chaincode on two different organizations

*peer lifecycle chaincode install Courier-scm.tar.gz --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE*



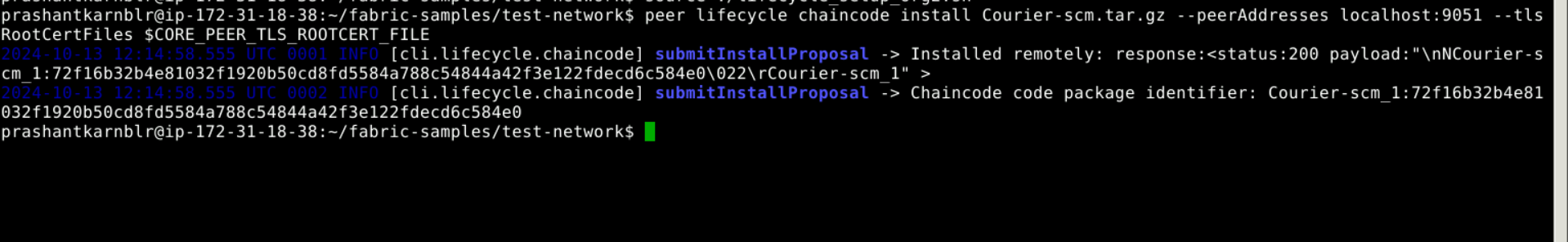
Open new tab to install in org2:

*nano lifecycle\_setup\_org2.sh*



*source ./lifecycle\_setup\_org2.sh*

*peer lifecycle chaincode install Courier-scm.tar.gz --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE*



**Verify the transaction id. For org1 and org2 it should be same.**

m\_1:72f16b32b4e81032f1920b50cd8fd5584a788c54844a42f3e122fdecd6c584e0\022\rCourier-scm\_1" >

2024-10-13 12:09:11.567 UTC 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: Courier-scm\_1:**72f16b32b4e81032f1920b50cd8fd5584a788c54844a42f3e122fdecd6c584e0**

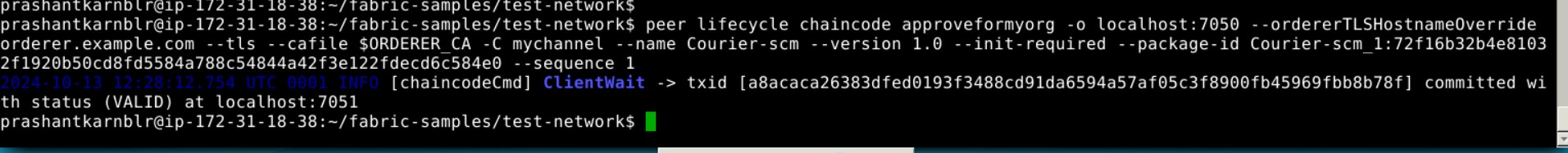
Chaincode code package identifier: Courier-scm\_1:**72f16b32b4e81032f1920b50cd8fd5584a788c54844a42f3e122fdecd6c584e0**

# Approve

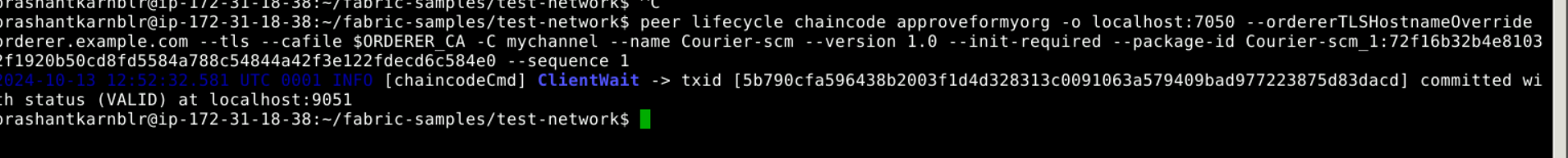
Approve for org1:

Execute the below command for org1 and org2:

*peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER\_CA -C mychannel --name Courier-scm --version 1.0 --init-required --package-id Courier-scm\_1:72f16b32b4e81032f1920b50cd8fd5584a788c54844a42f3e122fdecd6c584e0 --sequence 1*



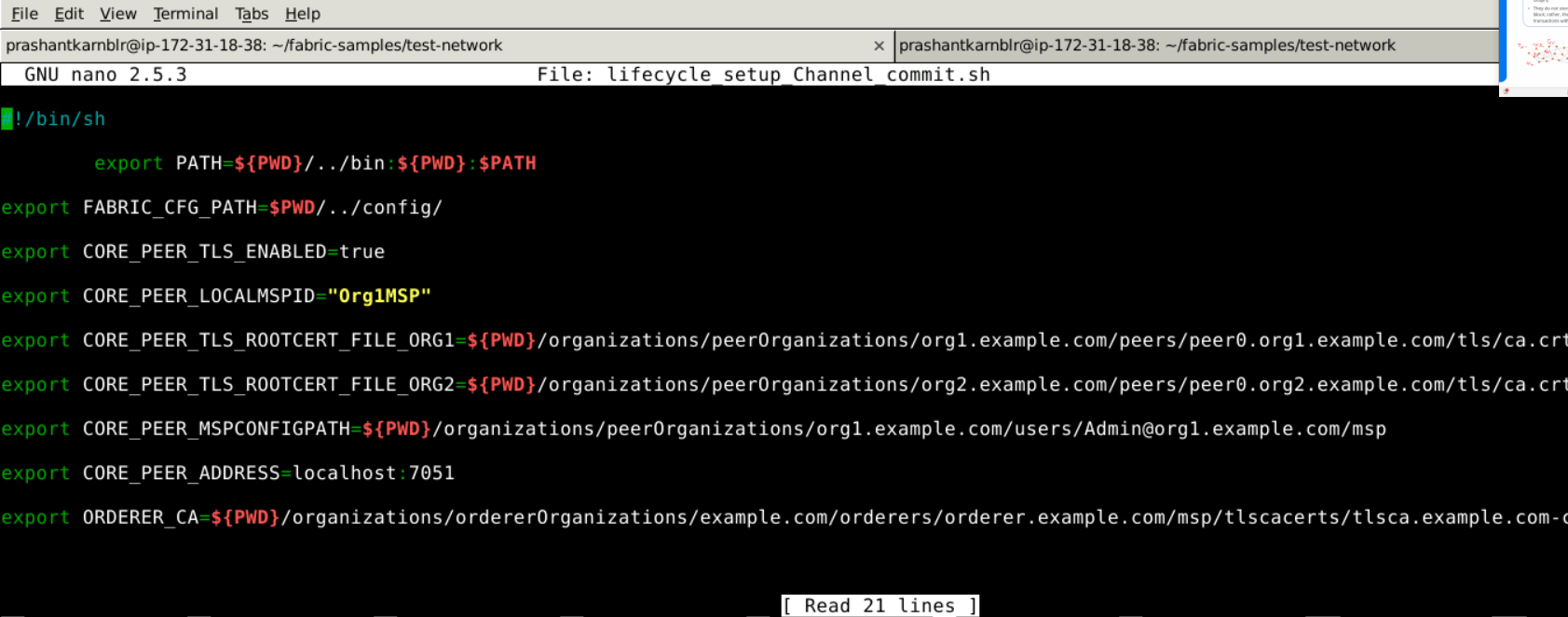
Approve for org2:



# Commit

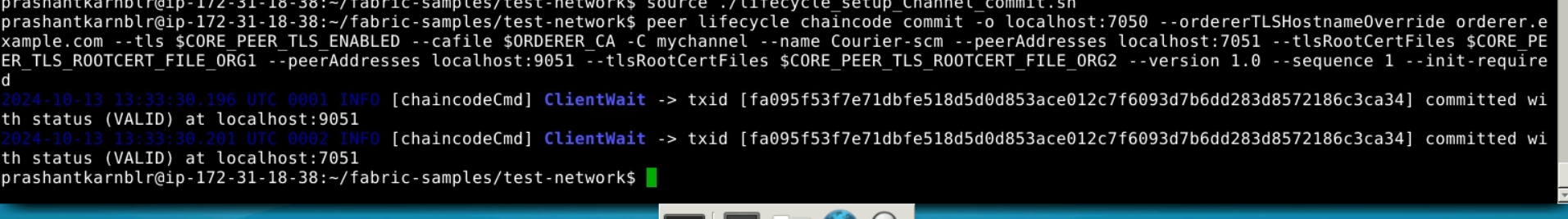
**We will allow org1 to commit the chain**

*nano lifecycle\_setup\_Channel\_commit.sh*



*source ./lifecycle\_setup\_Channel\_commit.sh*

*peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel --name Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 --version 1.0 --sequence 1 --init-required*



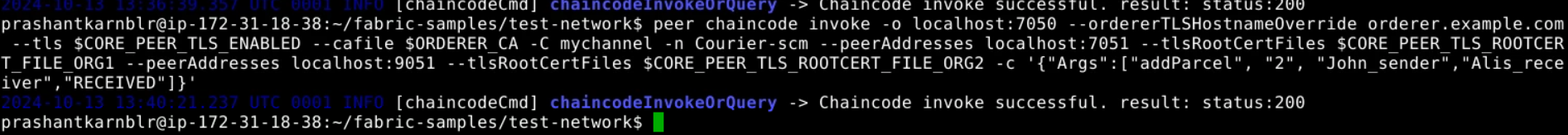
# Invoke/ Execution chaincode:

## Initiate the chaincode

*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 --isInit -c '{"Args":[]}'*

## Create parcel

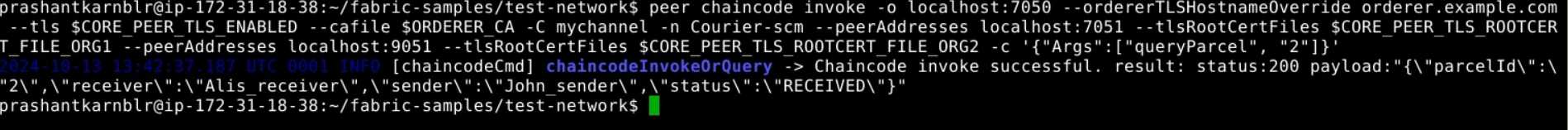
*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["addParcel", "2", "John\_sender","Alis\_receiver","RECEIVED"]}'*



## Query the created parcel

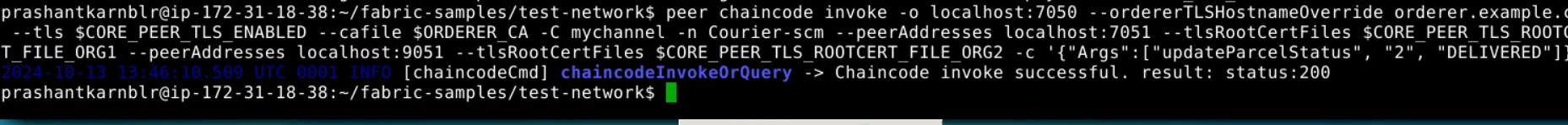
Query :

*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["queryParcel", "2"]}'*



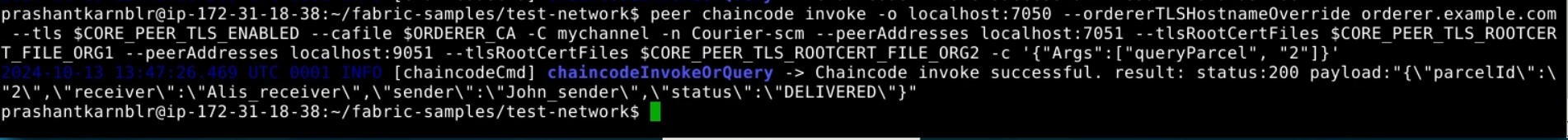
## Update the parcel status:

*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["updateParcelStatus", "2", "DELIVERED"]}'*



## Query the new status:

*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["queryParcel", "2"]}'*



## Failure case (Query not existed parcel):

*peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n Courier-scm --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["queryParcel", "2"]}'*

