The Chinese University of Hong Kong

Department of Financial Technology

FTEC5520 –Applied Blockchain & Cryptocurrency

Lab2 Report 1

Hyperledger Fabric Setup & Practice on AWS(Task4, 5)

Please edit this file directly but submitted a PDF version to blackboard finally.

1. Screen Capture of Main Steps:

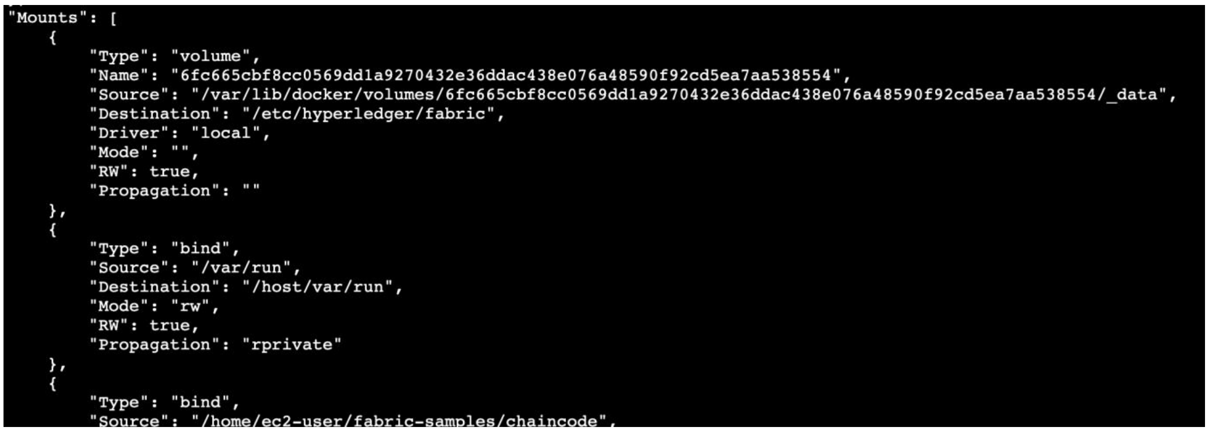
Please replace the sample photos with your own results.
Your screenshots must include these parts at least and detailed description of each screenshot:

Task 4 Run the RESTful API server.

1. On the Fabric client node. Inspect the docker config.

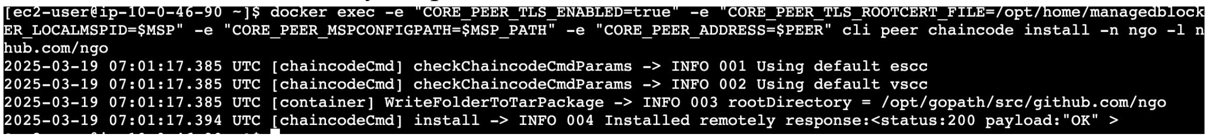
```
docker inspect cli
```

In the **Mounts** section, the Fabric CLI container mounts a folder from the Fabric client node EC2 instance: /home/ec2-user/fabric-samples/chaincode.



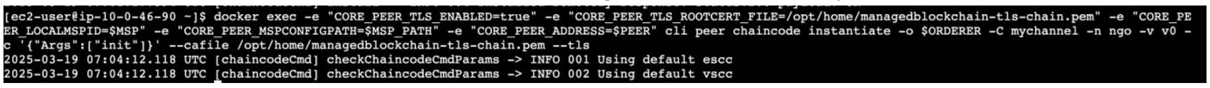
```
"Mounts": [
    {
        "Type": "volume",
        "Name": "6fc665cbf8cc0569dd1a9270432e36ddac438e076a48590f92cd5ea7aa538554",
        "Source": "/var/lib/docker/volumes/6fc665cbf8cc0569dd1a9270432e36ddac438e076a48590f92cd5ea7aa538554/_data",
        "Destination": "/etc/hyperledger/fabric",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
    },
    {
        "Type": "bind",
        "Source": "/var/run",
        "Destination": "/host/var/run",
        "Mode": "rw",
        "RW": true,
        "Propagation": "rprivate"
    },
    {
        "Type": "bind",
        "Source": "/home/ec2-user/fabric-samples/chaincode",
```

2. Install the chaincode on your peer.



```
[ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblock
ER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER" cli peer chaincode install -n ngo -l n
hub.com/ngo
2025-03-19 07:01:17.385 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2025-03-19 07:01:17.385 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2025-03-19 07:01:17.385 UTC [container] WriteFolderToTarPackage -> INFO 003 rootDirectory = /opt/gopath/src/github.com/ngo
2025-03-19 07:01:17.394 UTC [chaincodeCmd] install -> INFO 004 Installed remotely response:<status:200 payload:"OK" >
```

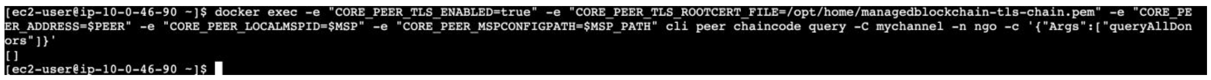
3. Instantiate the chaincode in order to bind ngo chaincode to mychannel.



```
[ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PE
ER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e "CORE_PEER_ADDRESS=$PEER" cli peer chaincode instantiate -o $ORDERER -C mychannel -n ngo -v v0 -
c '{"Args":["init"]}' --cafile /opt/home/managedblockchain-tls-chain.pem --tls
2025-03-19 07:04:12.118 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2025-03-19 07:04:12.118 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
```

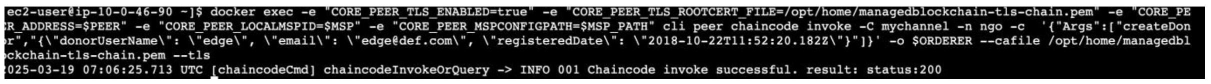
4. Query the chaincode to check whether it is installed.

You will get the following result with no donor:



```
[ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PE
ER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDon
ors"]}'
[]
[ec2-user@ip-10-0-46-90 ~]$
```

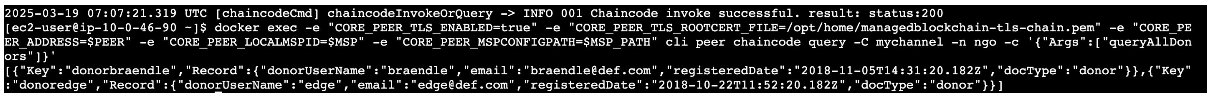
5. Invoke a transaction to add 2 donors.



```
ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PE
R_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode invoke -C mychannel -n ngo -c  '{"Args":["createDon
r","{\"donorUserName\": \"edge\", \"email\": \"edge@def.com\", \"registeredDate\": \"2018-10-22T11:52:20.182Z\"}"]}' -o $ORDERER --cafile /opt/home/managedbl
ckchain-tls-chain.pem --tls
025-03-19 07:06:25.713 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
```

6. Query the chaincode to view the results.
   Query All Users:

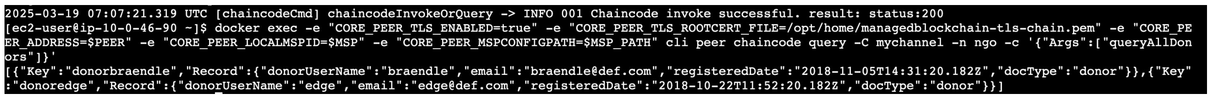
Query with result:



```
2025-03-19 07:07:21.319 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
[ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PE
ER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDon
ors"]}'
[{"Key":"donorbraendle","Record":{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"}},{"Key"
:"donoredge","Record":{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}}]
```

   Query Specific User:

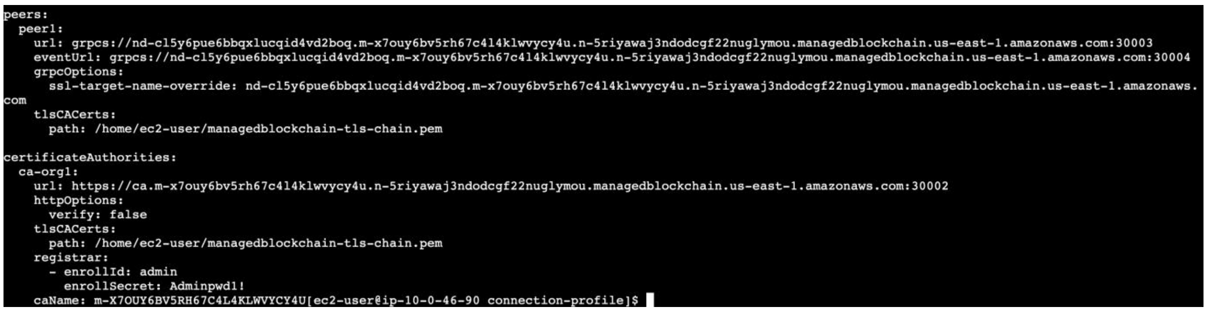
Query with result:



```
2025-03-19 07:07:21.319 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
[ec2-user@ip-10-0-46-90 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" -e "CORE_PE
ER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDon
ors"]}'
[{"Key":"donorbraendle","Record":{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"}},{"Key"
:"donoredge","Record":{"donorUserName":"edge","email":"edge@def.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"}}]
```

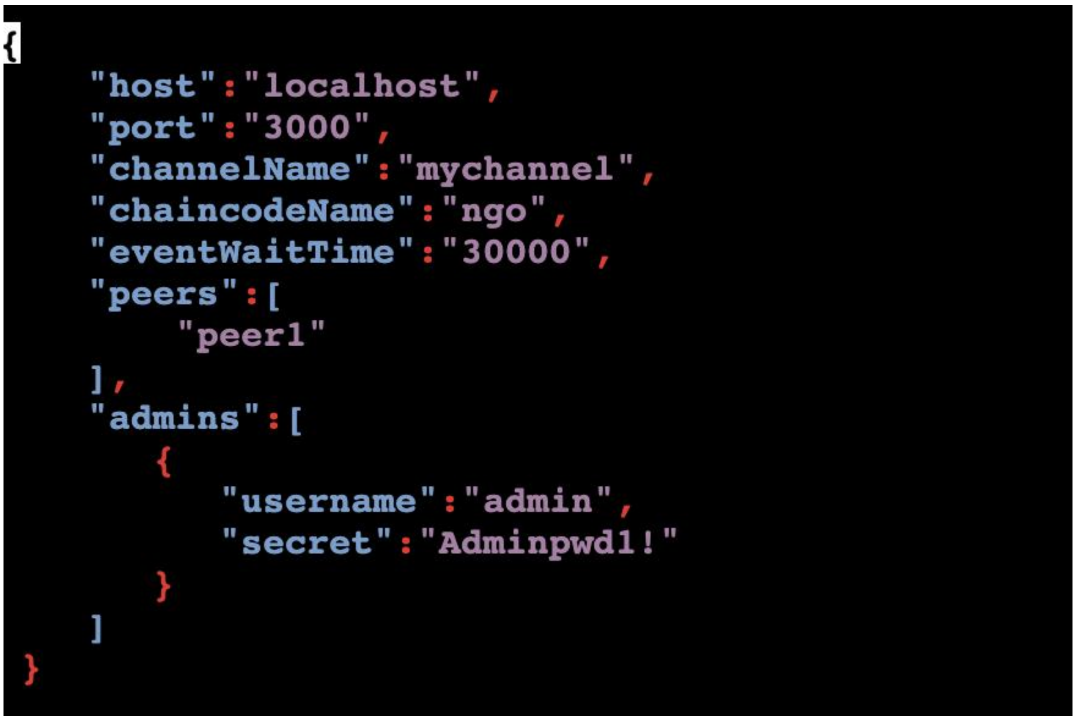
Task 5 Run the RESTful API server.

1. Generate the connection profile used to connect to Fabric network.
   You can see the content of the *ngo-connection-profile.yaml* with the endpoint of the order, peer and CA. The REST API uses the Fabric SDK to connect to order, peer and CA with these endpoints.



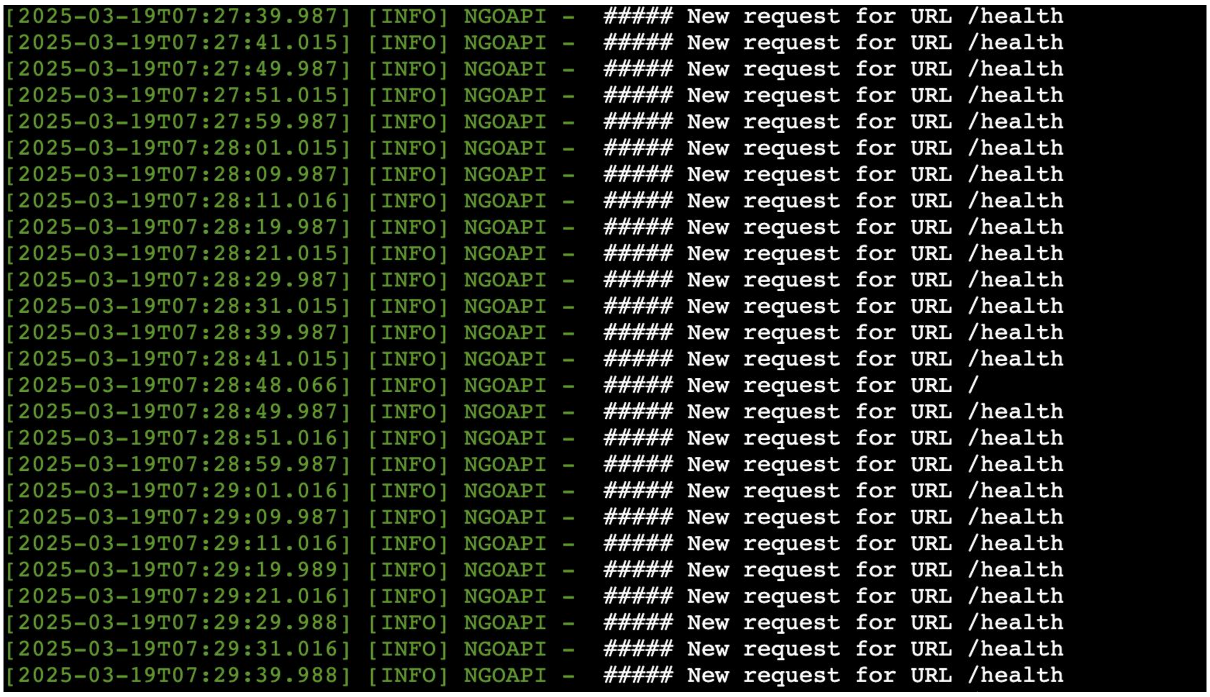
```
peers:
  peer1:
    url: grpcs://nd-cl5y6pue6bbqxlucqid4vd2boq.m-x7ouy6bv5rh67c4l4klwvycy4u.n-5riyawaj3ndodcgf22nuglymou.managedblockchain.us-east-1.amazonaws.com:30003
    eventUrl: grpcs://nd-cl5y6pue6bbqxlucqid4vd2boq.m-x7ouy6bv5rh67c4l4klwvycy4u.n-5riyawaj3ndodcgf22nuglymou.managedblockchain.us-east-1.amazonaws.com:30004
    grpcOptions:
      ssl-target-name-override: nd-cl5y6pue6bbqxlucqid4vd2boq.m-x7ouy6bv5rh67c4l4klwvycy4u.n-5riyawaj3ndodcgf22nuglymou.managedblockchain.us-east-1.amazonaws.
com
    tlsCACerts:
      path: /home/ec2-user/managedblockchain-tls-chain.pem

certificateAuthorities:
  ca-org1:
    url: https://ca.m-x7ouy6bv5rh67c4l4klwvycy4u.n-5riyawaj3ndodcgf22nuglymou.managedblockchain.us-east-1.amazonaws.com:30002
    httpOptions:
      verify: false
    tlsCACerts:
      path: /home/ec2-user/managedblockchain-tls-chain.pem
    registrar:
      - enrollId: admin
        enrollSecret: Adminpwd1!
    caName: m-X7OUY6BV5RH67C4L4KLWVYCY4U[ec2-user@ip-10-0-46-90 connection-profile]$
```

2. Check the config file used by app.js.



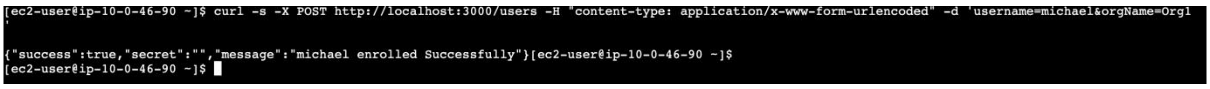
```
{
    "host":"localhost",
    "port":"3000",
    "channelName":"mychannel",
    "chaincodeName":"ngo",
    "eventWaitTime":"30000",
    "peers":[
        "peer1"
    ],
    "admins":[
       {
          "username":"admin",
          "secret":"Adminpwd1!"
       }
    ]
}
```

3. Make sure everything is set correctly. Then run the application.
   The server will run like this:



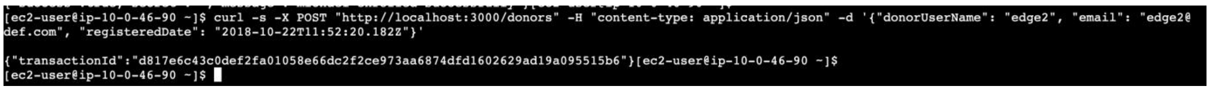
```
[2025-03-19T07:27:39.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:27:41.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:27:49.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:27:51.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:27:59.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:01.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:09.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:11.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:19.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:21.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:29.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:31.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:39.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:41.015] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:48.066] [INFO] NGOAPI -  ##### New request for URL /
[2025-03-19T07:28:49.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:51.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:28:59.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:01.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:09.987] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:11.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:19.989] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:21.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:29.988] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:31.016] [INFO] NGOAPI -  ##### New request for URL /health
[2025-03-19T07:29:39.988] [INFO] NGOAPI -  ##### New request for URL /health
```

4. Test the REST API.
   4.1. Register/enroll a user.
      - Call the REST resource users and create a user called michael.



```
[ec2-user@ip-10-0-46-90 ~]$ curl -s -X POST http://localhost:3000/users -H "content-type: application/x-www-form-urlencoded" -d 'username=michael&orgName=Org1
'

{"success":true,"secret":"","message":"michael enrolled Successfully"}[ec2-user@ip-10-0-46-90 ~]$
[ec2-user@ip-10-0-46-90 ~]$ 
```

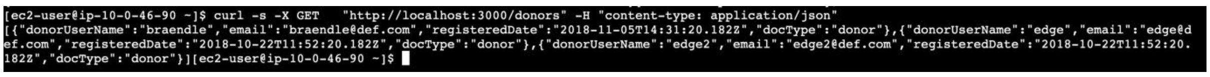
   4.2. Post a Donor through REST API instead of in Fabric layer.
      - Do the same as create a donor in chaincode.



```
[ec2-user@ip-10-0-46-90 ~]$ curl -s -X POST "http://localhost:3000/donors" -H "content-type: application/json" -d '{"donorUserName": "edge2", "email": "edge2@
def.com", "registeredDate": "2018-10-22T11:52:20.182Z"}'

{"transactionId":"d817e6c43c0def2fa01058e66dc2f2ce973aa6874dfd1602629ad19a095515b6"}[ec2-user@ip-10-0-46-90 ~]$
[ec2-user@ip-10-0-46-90 ~]$ 
```

   4.3. Query all the donors with REST API.

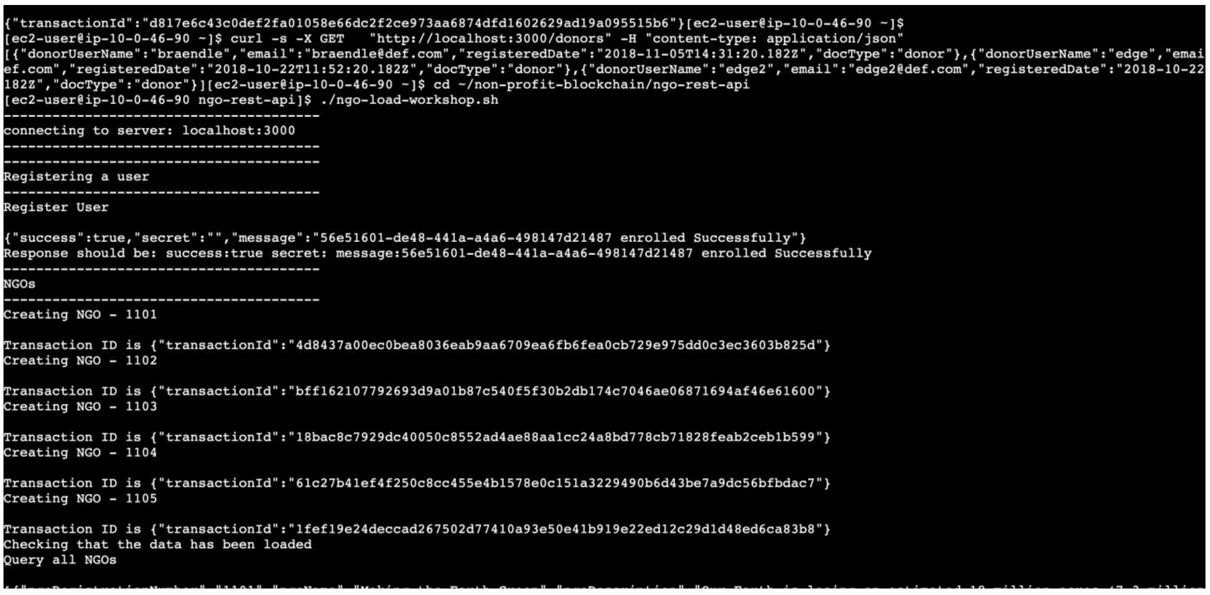
You will see the command with the results:



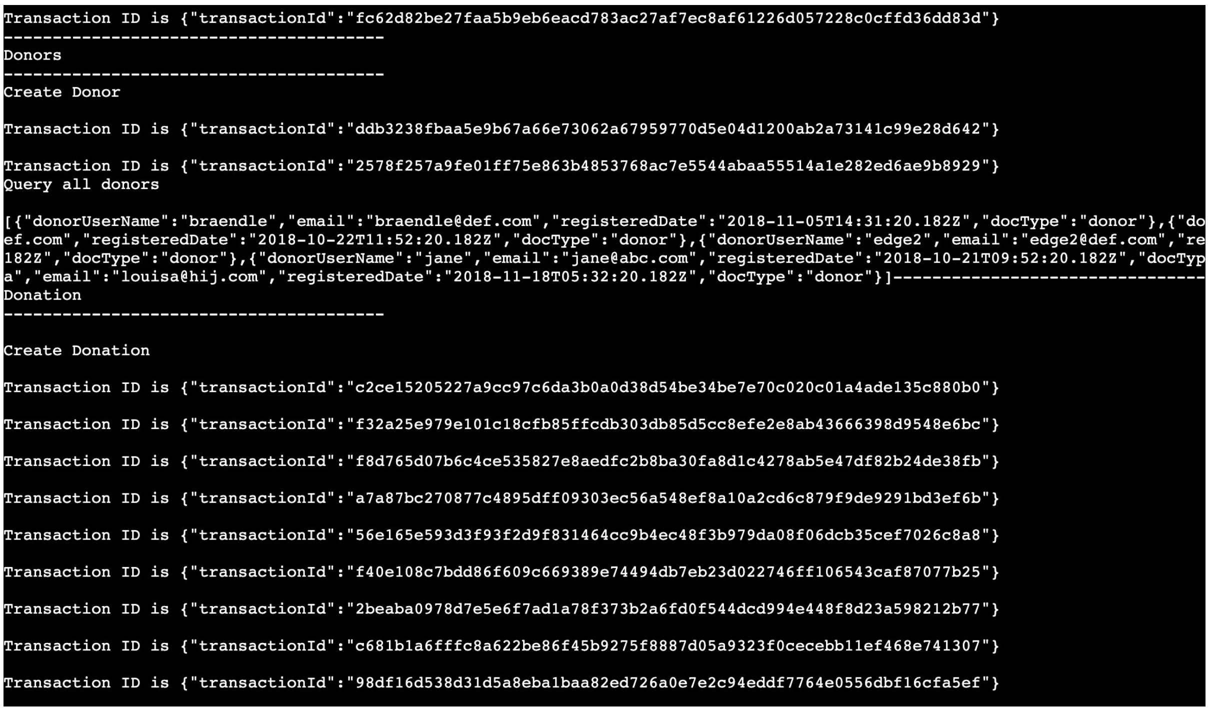
```
[ec2-user@ip-10-0-46-90 ~]$ curl -s -X GET   "http://localhost:3000/donors" -H "content-type: application/json"
[{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"},{"donorUserName":"edge","email":"edge@d
ef.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"},{"donorUserName":"edge2","email":"edge2@def.com","registeredDate":"2018-10-22T11:52:20.
182Z","docType":"donor"}][ec2-user@ip-10-0-46-90 ~]$ 
```

   4.4. Run the script to run the workshop test data.
      - It creates some donors and some donations.

      The first part and the last part of the result would be like this:



```
{"transactionId":"d817e6c43c0def2fa01058e66dc2f2ce973aa6874dfd1602629ad19a095515b6"}[ec2-user@ip-10-0-46-90 ~]$
[ec2-user@ip-10-0-46-90 ~]$ curl -s -X GET   "http://localhost:3000/donors" -H "content-type: application/json"
[{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"},{"donorUserName":"edge","emai
ef.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"},{"donorUserName":"edge2","email":"edge2@def.com","registeredDate":"2018-10-22
182Z","docType":"donor"}][ec2-user@ip-10-0-46-90 ~]$ cd ~/non-profit-blockchain/ngo-rest-api
[ec2-user@ip-10-0-46-90 ngo-rest-api]$ ./ngo-load-workshop.sh
---------------------------------------
connecting to server: localhost:3000
---------------------------------------
---------------------------------------
Registering a user
---------------------------------------
Register User

{"success":true,"secret":"","message":"56e51601-de48-441a-a4a6-498147d21487 enrolled Successfully"}
Response should be: success:true secret: message:56e51601-de48-441a-a4a6-498147d21487 enrolled Successfully
---------------------------------------
NGOs
---------------------------------------
Creating NGO - 1101

Transaction ID is {"transactionId":"4d8437a00ec0bea8036eab9aa6709ea6fb6fea0cb729e975dd0c3ec3603b825d"}
Creating NGO - 1102

Transaction ID is {"transactionId":"bff162107792693d9a01b87c540f5f30b2db174c7046ae06871694af46e61600"}
Creating NGO - 1103

Transaction ID is {"transactionId":"18bac8c7929dc40050c8552ad4ae88aa1cc24a8bd778cb71828feab2ceb1b599"}
Creating NGO - 1104

Transaction ID is {"transactionId":"61c27b41ef4f250c8cc455e4b1578e0c151a3229490b6d43be7a9dc56bfbdac7"}
Creating NGO - 1105

Transaction ID is {"transactionId":"1fef19e24deccad267502d77410a93e50e41b919e22ed12c29d1d48ed6ca83b8"}
Checking that the data has been loaded
Query all NGOs
```



```
Transaction ID is {"transactionId":"fc62d82be27faa5b9eb6eacd783ac27af7ec8af61226d057228c0cffd36dd83d"}
---------------------------------------
Donors
---------------------------------------
Create Donor

Transaction ID is {"transactionId":"ddb3238fbaa5e9b67a66e73062a67959770d5e04d1200ab2a73141c99e28d642"}

Transaction ID is {"transactionId":"2578f257a9fe01ff75e863b4853768ac7e5544abaa55514a1e282ed6ae9b8929"}
Query all donors

[{"donorUserName":"braendle","email":"braendle@def.com","registeredDate":"2018-11-05T14:31:20.182Z","docType":"donor"},{"do
ef.com","registeredDate":"2018-10-22T11:52:20.182Z","docType":"donor"},{"donorUserName":"edge2","email":"edge2@def.com","re
182Z","docType":"donor"},{"donorUserName":"jane","email":"jane@abc.com","registeredDate":"2018-10-21T09:52:20.182Z","docTyp
a","email":"louisa@hij.com","registeredDate":"2018-11-18T05:32:20.182Z","docType":"donor"}]--------------------------------
Donation
---------------------------------------

Create Donation

Transaction ID is {"transactionId":"c2ce15205227a9cc97c6da3b0a0d38d54be34be7e70c020c01a4ade135c880b0"}

Transaction ID is {"transactionId":"f32a25e979e101c18cfb85ffcdb303db85d5cc8efe2e8ab43666398d9548e6bc"}

Transaction ID is {"transactionId":"f8d765d07b6c4ce535827e8aedfc2b8ba30fa8d1c4278ab5e47df82b24de38fb"}

Transaction ID is {"transactionId":"a7a87bc270877c4895dff09303ec56a548ef8a10a2cd6c879f9de9291bd3ef6b"}

Transaction ID is {"transactionId":"56e165e593d3f93f2d9f831464cc9b4ec48f3b979da08f06dcb35cef7026c8a8"}

Transaction ID is {"transactionId":"f40e108c7bdd86f609c669389e74494db7eb23d022746ff106543caf87077b25"}

Transaction ID is {"transactionId":"2beaba0978d7e5e6f7ad1a78f373b2a6fd0f544dcd994e448f8d23a598212b77"}

Transaction ID is {"transactionId":"c681b1a6fffc8a622be86f45b9275f8887d05a9323f0cecebb11ef468e741307"}

Transaction ID is {"transactionId":"98df16d538d31d5a8eba1baa82ed726a0e7e2c94eddf7764e0556dbf16cfa5ef"}
```