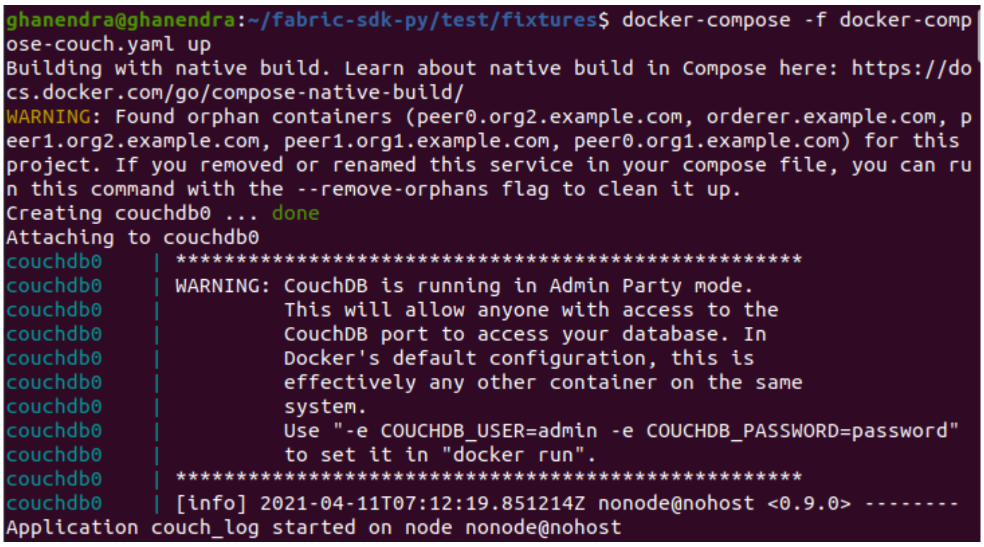
MT19213
Ghanendra Singh
IBC Project Report 2

Two weeks Work Progress
(30-03-21 - 11-04-21)
Below are the key points which I was able to implement in two weeks.

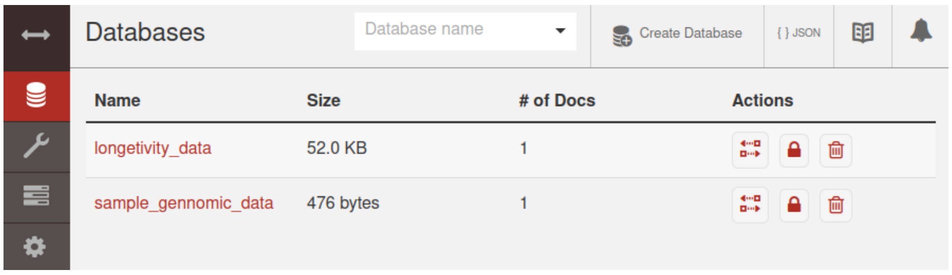
- Installation of hyperledger/fabric-couchdb docker container using *docker pull hyperledger/fabric-couchdb* command at terminal inside fabric-sdk folder.
- Modified core.yaml file contents from default state database from levelDB to CouchDB located at fabric-sdk-py/fabric-bin/config for storing private data.
- Identified Assets (Ageing genomic data) ---> modelled as JSON data for couchDB.
- Setup and running couchdb-python locally at http://127.0.0.1:5984/ by following a simple example from 1. Getting started with couchdb-python.
- Docker Sample application https://docs.docker.com/get-started/02_our_app/
- Access CouchDB Project Fauxton interface to interact with data locally at http://127.0.0.1:5984/_utils Fauxton Visual Guide
- Lists all the databases created in couchDB locally at http://127.0.0.1:5984/_all_dbs/
- Tutorial: https://www.tutorialkart.com/couchdb-tutorial/
- https://deeptiman.medium.com/couchdb-as-a-state-database-in-hyperledger-fabric-adb5d820c82e
- http://www.dev.fyicenter.com/1001245_CouchDB_Container_Used_in_Hyperledger_Fabric.html
- Started couchdb: ghanendra@ghanendra:~/fabric-sdk-py/test/fixtures$ docker-compose -f docker-compose-couch.yaml up
- Stored enrollment data using ca service on the couchdb wallet store.
- Learnt about shim APIs defined inside chaincode used for reading and writing private data. Get_Transient() API is used to access private data and GetPrivateData QueryResult() API for accessing private data from couchDB.
- Collection_config_policy defined in hfc/fabric/channel, important for selective private data sharing among organisations. Private Data
- Installed marbles_cc_private chaincode example file written in go onto channel. It will be used as a reference to create my own chaincode for next week's work.
- Failed to define Transient_map data field defined during chaincode instantiation located in hfc/utils as it is used to pass private data. Hyperledger Fabric SDK for node.js Tutorial: How to use private data

CouchDB Running from Docker Container



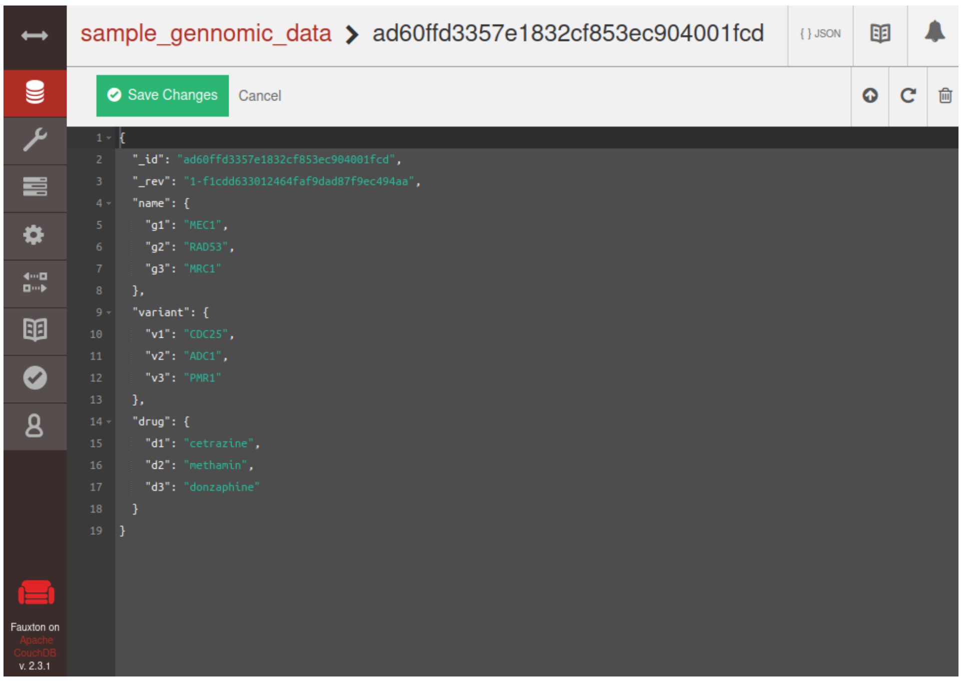
```
ghanendra@ghanendra:~/fabric-sdk-py/test/fixtures$ docker-compose -f docker-comp
ose-couch.yaml up
Building with native build. Learn about native build in Compose here: https://do
cs.docker.com/go/compose-native-build/
WARNING: Found orphan containers (peer0.org2.example.com, orderer.example.com, p
eer1.org2.example.com, peer1.org1.example.com, peer0.org1.example.com) for this
project. If you removed or renamed this service in your compose file, you can ru
n this command with the --remove-orphans flag to clean it up.
Creating couchdb0 ... done
Attaching to couchdb0
couchdb0      | ****************************************************
couchdb0      | WARNING: CouchDB is running in Admin Party mode.
couchdb0      |          This will allow anyone with access to the
couchdb0      |          CouchDB port to access your database. In
couchdb0      |          Docker's default configuration, this is
couchdb0      |          effectively any other container on the same
couchdb0      |          system.
couchdb0      |          Use "-e COUCHDB_USER=admin -e COUCHDB_PASSWORD=password"
couchdb0      |          to set it in "docker run".
couchdb0      | ****************************************************
couchdb0      | [info] 2021-04-11T07:12:19.851214Z nonode@nohost <0.9.0> --------
Application couch_log started on node nonode@nohost
```

CouchDB all databases



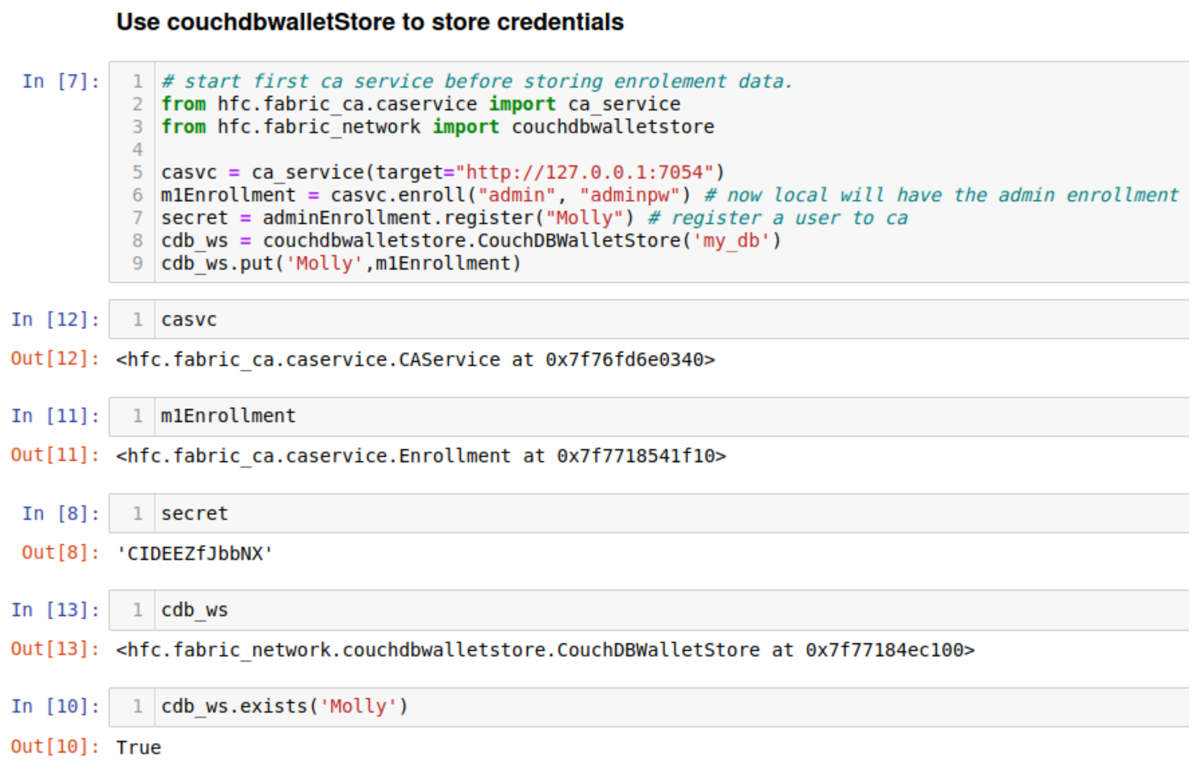
Name	Size	# of Docs	Actions
longetivity_data	52.0 KB	1	
sample_gennomic_data	476 bytes	1	

CouchDB Genomic Data Storage


sample_gennomic_data > ad60ffd3357e1832cf853ec904001fcd
{ } JSON
Save Changes
Cancel
Fauxton on Apache CouchDB v. 2.3.1

```
{
  "_id": "ad60ffd3357e1832cf853ec904001fcd",
  "_rev": "1-f1cdd633012464faf9dad87f9ec494aa",
  "name": {
    "g1": "MEC1",
    "g2": "RAD53",
    "g3": "MRC1"
  },
  "variant": {
    "v1": "CDC25",
    "v2": "ADC1",
    "v3": "PMR1"
  },
  "drug": {
    "d1": "cetrazine",
    "d2": "methamin",
    "d3": "donzaphine"
  }
}
```

CouchDB Wallet store

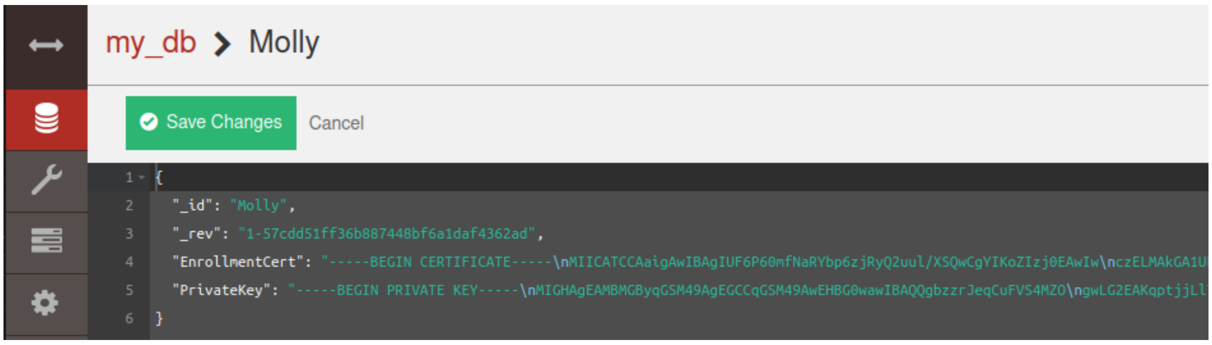


Use couchdbwalletStore to store credentials

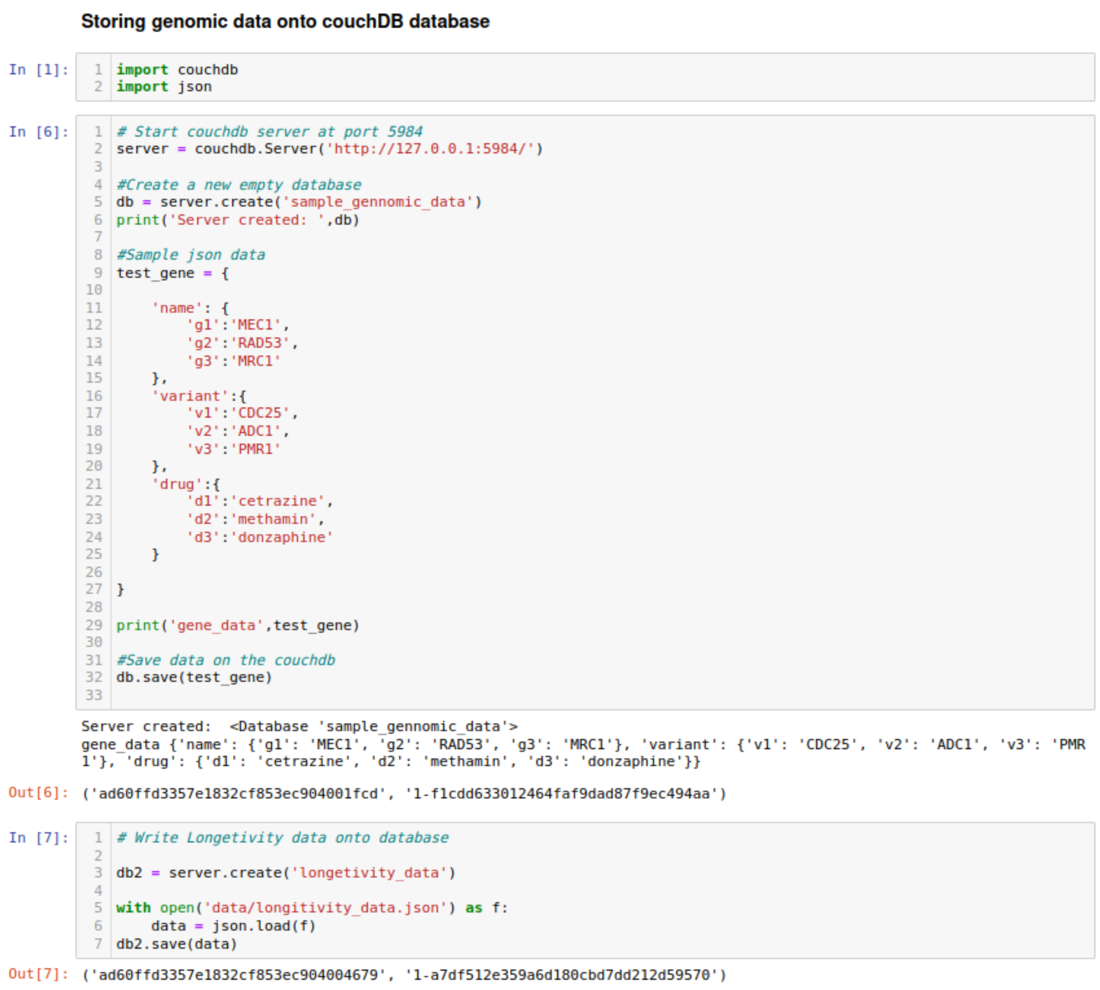
```
In [7]:
# start first ca service before storing enrolement data.
from hfc.fabric_ca.caservice import ca_service
from hfc.fabric_network import couchdbwalletstore

casvc = ca_service(target="http://127.0.0.1:7054")
m1Enrollment = casvc.enroll("admin", "adminpw") # now local will have the admin enrollment
secret = adminEnrollment.register("Molly") # register a user to ca
cdb_ws = couchdbwalletstore.CouchDBWalletStore('my_db')
cdb_ws.put('Molly',m1Enrollment)
```

```
In [12]: casvc
Out[12]: <hfc.fabric_ca.caservice.CAService at 0x7f76fd6e0340>

In [11]: m1Enrollment
Out[11]: <hfc.fabric_ca.caservice.Enrollment at 0x7f7718541f10>

In [8]: secret
Out[8]: 'CIDEEZfJbbNX'

In [13]: cdb_ws
Out[13]: <hfc.fabric_network.couchdbwalletstore.CouchDBWalletStore at 0x7f77184ec100>

In [10]: cdb_ws.exists('Molly')
Out[10]: True
```

Couchdb database storing example user Molly’s enrollment certificate and Private Key.



```
{
  "_id": "Molly",
  "_rev": "1-57cdd51ff36b887448bf6a1daf4362ad",
  "EnrollmentCert": "-----BEGIN CERTIFICATE-----\nMIICATCCAaigAwIBAgIUF6P60mfNaRYbp6zjRyQ2uul/XSQwCgYIKoZIzj0EAwIw\nczELMAkGA1U
  "PrivateKey": "-----BEGIN PRIVATE KEY-----\nMIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgbzzrJeqCuFVS4MZO\ngwLG2EAKqptjjLl
}
```



Storing genomic data onto couchDB database

```
In [1]:
import couchdb
import json
```

```
In [6]:
# Start couchdb server at port 5984
server = couchdb.Server('http://127.0.0.1:5984/')

#Create a new empty database
db = server.create('sample_gennomic_data')
print('Server created: ',db)

#Sample json data
test_gene = {

    'name': {
        'g1':'MEC1',
        'g2':'RAD53',
        'g3':'MRC1'
    },
    'variant':{
        'v1':'CDC25',
        'v2':'ADC1',
        'v3':'PMR1'
    },
    'drug':{
        'd1':'cetrazine',
        'd2':'methamin',
        'd3':'donzaphine'
    }

}

print('gene_data',test_gene)

#Save data on the couchdb
db.save(test_gene)

```

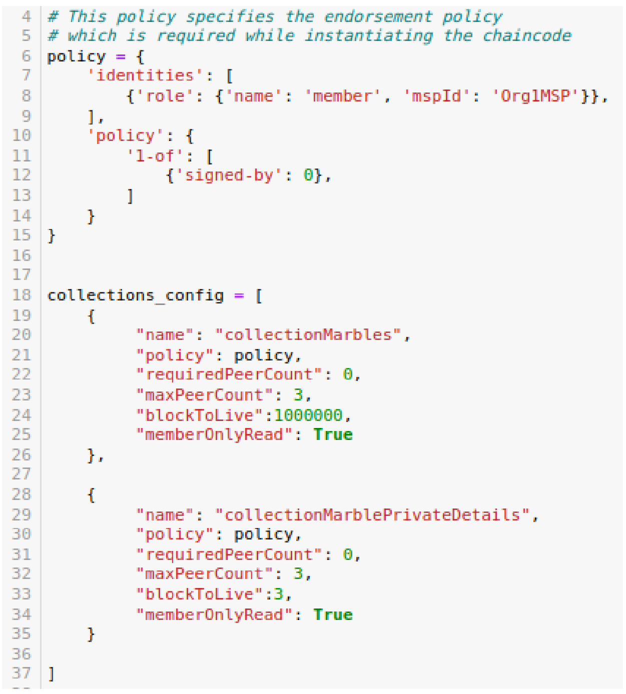
```
Server created:  <Database 'sample_gennomic_data'>
gene_data {'name': {'g1': 'MEC1', 'g2': 'RAD53', 'g3': 'MRC1'}, 'variant': {'v1': 'CDC25', 'v2': 'ADC1', 'v3': 'PMR
1'}, 'drug': {'d1': 'cetrazine', 'd2': 'methamin', 'd3': 'donzaphine'}}
```

```
Out[6]: ('ad60ffd3357e1832cf853ec904001fcd', '1-f1cdd633012464faf9dad87f9ec494aa')
```

```
In [7]:
# Write Longetivity data onto database

db2 = server.create('longetivity_data')

with open('data/longitivity_data.json') as f:
    data = json.load(f)
db2.save(data)
```

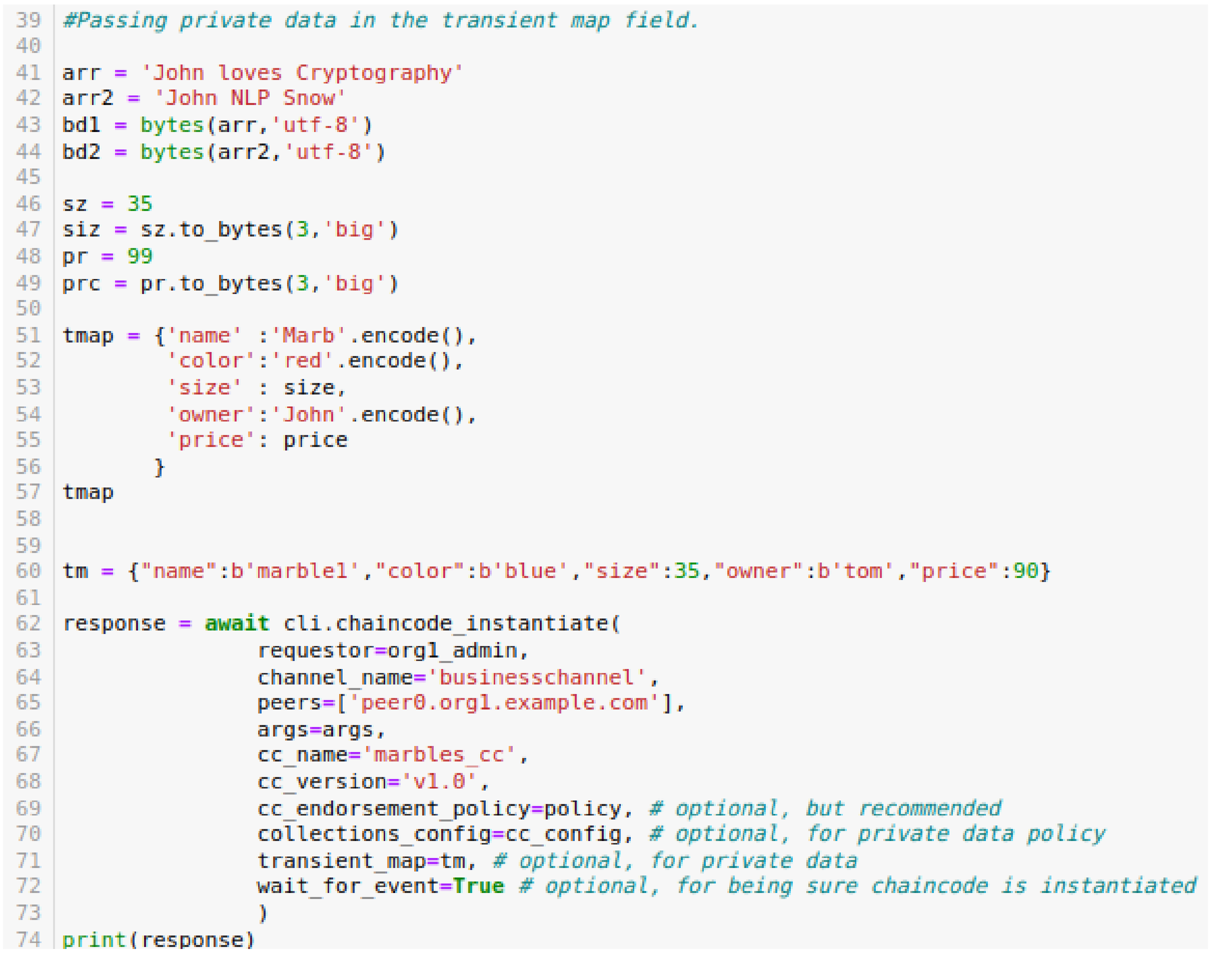
```
Out[7]: ('ad60ffd3357e1832cf853ec904004679', '1-a7df512e359a6d180cbd7dd212d59570')
```

Collection_config and policy defined properly, earlier it was incorrect



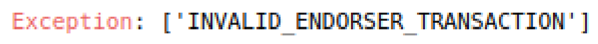
```
# This policy specifies the endorsement policy
# which is required while instantiating the chaincode
policy = {
    'identities': [
        {'role': {'name': 'member', 'mspId': 'Org1MSP'}},
    ],
    'policy': {
        '1-of': [
            {'signed-by': 0},
        ]
    }
}


collections_config = [
    {
         "name": "collectionMarbles",
         "policy": policy,
         "requiredPeerCount": 0,
         "maxPeerCount": 3,
         "blockToLive":1000000,
         "memberOnlyRead": True
    },

    {
         "name": "collectionMarblePrivateDetails",
         "policy": policy,
         "requiredPeerCount": 0,
         "maxPeerCount": 3,
         "blockToLive":3,
         "memberOnlyRead": True
    }

]
```

Transient map definition.



```python
#Passing private data in the transient map field.

arr = 'John loves Cryptography'
arr2 = 'John NLP Snow'
bd1 = bytes(arr,'utf-8')
bd2 = bytes(arr2,'utf-8')

sz = 35
siz = sz.to_bytes(3,'big')
pr = 99
prc = pr.to_bytes(3,'big')

tmap = {'name' :'Marb'.encode(),
        'color':'red'.encode(),
        'size' : size,
        'owner':'John'.encode(),
        'price': price
       }
tmap


tm = {"name":b'marble1',"color":b'blue',"size":35,"owner":b'tom',"price":90}

response = await cli.chaincode_instantiate(
               requestor=org1_admin,
               channel_name='businesschannel',
               peers=['peer0.org1.example.com'],
               args=args,
               cc_name='marbles_cc',
               cc_version='v1.0',
               cc_endorsement_policy=policy, # optional, but recommended
               collections_config=cc_config, # optional, for private data policy
               transient_map=tm, # optional, for private data
               wait_for_event=True # optional, for being sure chaincode is instantiated
               )
print(response)
```

Exception error in transient map
Currently I am facing issues while defining the transient map section, will debug it soon.



```
Exception: ['INVALID_ENDORSER_TRANSACTION']
```

Couch_data jupyter file available at
https://github.com/Ghanendra19213/IBC/blob/main/couchdb_data.ipynb

Summary.

As per the plan, I was able to implement couchdb data storage, next is to access couchdb from the chaincode using shim APIs. Once I am through with it, I will quickly define the fabric network and start with writing smart contracts in Go. Python SDK documentation is less, which makes it challenging simultaneously interesting to dig deeper taking inspiration from node js documentation. So, by the next three weeks, I will try to complete the project.

Thanks
Ghanendra