Hyperledger Mentorship Project Presentation

November 2021

Kiv Chen

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Introduction**
  - **Name**: Kiv Chen
  - **Location:** Liverpool, United Kingdom
  - **University:** University of Liverpool
  - **Mentor(s):** Dixing Xu, Baohua Yang, Guillaume Cisco, Wang Dong
  - **Hyperledger Project**: Support Decentralized Governance for Smart Contracts in Fabric Python SDK

Hyperledger Fabric Python SDK

HYPERLEDGER
FABRIC

Project Description: With the introduction of Fabric v2.x, a more decentralized way of **chaincode management** is implemented. There are several improvements over the previous **lifecycle** and it requires several changes on the sdk. This project aims to support **decentralized governance** for smart contracts in fabric python sdk and add features such as private data sharing/verifying and external chaincode launcher. The projects will provide a user-friendly and easy-to-use tool for fabric developers and operators.

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Project Objectives**:
  - Obj 1: Chaincode lifecycle management on sdk-py
  - Obj 2: Align with new features of fabric 2.x
  - Obj 3: Documentation on using fabric 2.x

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Project Deliverables:**
  - Deliverable 1: Development of fabric 2.2+ Lifecycle 2.0 full support
  - Deliverable 2: Documentation for fabric 2.2+ features

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Project Execution & Accomplishments**:
  - Get familiar with Fabric & Fabric SDK
  - Fulfill and extend the decentralized support for Fabric 2.x
  - Revise some previous APIs according to mentor's suggestions

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 1 Setup**
  - Setup needed attributes …
    - Name/Version
    - Sequence
    - Endorsement Policy
    - Validation configuration
  - … that influence the Fabric “citizens”
    - Client and Peers
    - Channels
    - Chaincode

```python
async def chaincode_definition_operation(self, requestor, peers, channel_name, cc_version, package_id=None,
                                         signature_policy=None, channel_config_policy=None, init_required=False,
                                         sequence=1, collections_config=None, endorsement_plugin="",
                                         validation_plugin="", wait_for_event=False,
                                         wait_for_event_timeout=DEFAULT_WAIT_FOR_EVENT_TIMEOUT):
    target_peers = self._client.get_target_peers(peers)

    tx_context = create_tx_context(requestor, requestor.cryptoSuite, TXProposalRequest())

    application_policy = policy_pb2.ApplicationPolicy()
    if signature_policy:
        application_policy.signature_policy.CopyFrom(build_policy(s2d().parse(signature_policy), returnProto=True))
    elif channel_config_policy:
        application_policy.channel_config_policy_reference = proto_str(channel_config_policy)

    # package_if needed only for approval operation and so can be used to differentiate between operations
    args = lp.ApproveChaincodeDefinitionForMyOrgArgs() if package_id else lp.CommitChaincodeDefinitionArgs()
    args.name = proto_str(self._name)
    args.version = proto_str(cc_version)
    args.sequence = sequence
    args.endorsement_plugin = endorsement_plugin
    args.validation_plugin = validation_plugin
    args.validation_parameter = application_policy.SerializeToString()
    method = LC_COMMIT
    if package_id:
        method = LC_APPROVE_FOR_MY_ORG
        args.source.local_package.package_id = proto_str(package_id)
    args.init_required = init_required
    if collections_config is not None:
        args.collections.collections = build_collection_config_proto(collections_config)

    responses, proposal, header = utils.send_proposal(tx_context, target_peers, args, method, LIFECYCLE_CC,
                                                      channel_name)
    res = await asyncio.gather(*responses)
    self.parse_proposal_res(res)
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 2 Package**
  - Build a tar file from the source code files and metadata files
  - Available to be sent to other organizations

```python
def package(self, source_path, label, dest_path=None, cc_type=CC_TYPE_GOLANG):
    """
    Package chaincode

    :param source_path: Path to the chaincode
    :param label: The package label contains a human-readable description of the package
    :param dest_path: Path with file name where package would be stored
    :param cc_type: Language the chaincode is written in (default "golang")
    :return: bytes of the packaged chaincode
    """
    metadata = {
        "path": source_path,
        "type": cc_type,
        "label": label
    }
    tar_bytes = lifecycle_package(package_chaincode(source_path, cc_type), metadata)
    if dest_path:
        with open(dest_path, "wb") as file:
            file.write(tar_bytes)
    return tar_bytes
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 3 Install**
  - Send the packaged code to a peer
  - Save the hash value returned

```python
async def install(self, requestor, peers, packaged_cc=None):
    """
    Install chaincode to given peers by requestor role

    :param requestor: User role who issue the request
    :param peers: List of  peer name and/or Peer to install
    :param packaged_cc: packaged chaincode
    :return: A dict representation of `InstallChaincodeResult`
    """
    target_peers = self._client.get_target_peers(peers)

    tx_context = create_tx_context(requestor, requestor.cryptoSuite, TXProposalRequest())

    install_args = lp.InstallChaincodeArgs()
    install_args.chaincode_install_package = packaged_cc

    responses, proposal, header = utils.send_proposal(tx_context, target_peers, install_args, LC_INSTALL,
                                                      LIFECYCLE_CC)
    res = await asyncio.gather(*responses)
    return self.parse_proposal_res(res, lp.InstallChaincodeResult)
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 4 Approve for Organization**
  - Send a “approve chaincode definition for organization” chaincode lifecycle transaction to one peer in our organization
  - Commit the transaction (send to orderer)

```python
async def approve_for_my_org(self, requestor, peers, channel, cc_version, package_id, signature_policy=None,
                             channel_config_policy=None, init_required=False, sequence=1, collections_config=None,
                             endorsement_plugin="", validation_plugin="", wait_for_event=True,
                             wait_for_event_timeout=DEFAULT_WAIT_FOR_EVENT_TIMEOUT):
    """
    Approve chaincode definition for current org

    :param requestor: User role who issue the request
    :param peers: List of  peer name and/or Peer to install
    :param channel: channel name
    :param cc_version: chaincode version
    :param package_id: The identifier of the chaincode install package
    :param signature_policy: The endorsement policy specified as a signature policy
    :param channel_config_policy: The endorsement policy specified as a channel config policy reference
    :param init_required: Whether the chaincode requires invoking 'init'
    :param sequence: The sequence number of the chaincode definition for the channel
    :param collections_config: collection configuration
    :param validation_plugin: The name of the validation plugin to be used for this chaincode
    :param endorsement_plugin: The name of the endorsement plugin to be used for this chaincode
    :param wait_for_event: Whether to wait for the event from each peer's deliver filtered service signifying
     that the transaction has been committed successfully (default true)
    :param wait_for_event_timeout: Time to wait for the event from each peer
    """
    return await self.chaincode_definition_operation(requestor, peers, channel, cc_version, package_id=package_id,
                                                     signature_policy=signature_policy,
                                                     channel_config_policy=channel_config_policy,
                                                     init_required=init_required, sequence=sequence,
                                                     collections_config=collections_config,
                                                     validation_plugin=validation_plugin,
                                                     endorsement_plugin=endorsement_plugin,
                                                     wait_for_event=wait_for_event,
                                                     wait_for_event_timeout=wait_for_event_timeout)
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 5 Commit**
  - Send a “commit definition chaincode for channel” chaincode lifecycle transaction to enough organizations
  - Commit the transaction (send to orderer)

```python
async def commit_definition(self, requestor, peers, channel, cc_version, signature_policy=None,
                            channel_config_policy=None, init_required=False, sequence=1, collections_config=None,
                            endorsement_plugin="", validation_plugin="", wait_for_event=True,
                            wait_for_event_timeout=DEFAULT_WAIT_FOR_EVENT_TIMEOUT):
    """
    Commit the chaincode definition on the channel.

    :param requestor: User role who issue the request
    :param peers: List of  peer name and/or Peer to install
    :param channel: channel name
    :param cc_version: chaincode version
    :param signature_policy: The endorsement policy specified as a signature policy
    :param channel_config_policy: The endorsement policy specified as a channel config policy reference
    :param init_required: Whether the chaincode requires invoking 'init'
    :param sequence: The sequence number of the chaincode definition for the channel
    :param collections_config: collection configuration
    :param validation_plugin: The name of the validation plugin to be used for this chaincode
    :param endorsement_plugin: The name of the endorsement plugin to be used for this chaincode
    :param wait_for_event: Whether to wait for the event from each peer's deliver filtered service signifying
     that the transaction has been committed successfully (default true)
    :param wait_for_event_timeout: Time to wait for the event from each peer
    """
    return await self.chaincode_definition_operation(requestor, peers, channel, cc_version, package_id=None,
                                                     signature_policy=signature_policy,
                                                     channel_config_policy=channel_config_policy,
                                                     init_required=init_required, sequence=sequence,
                                                     collections_config=collections_config,
                                                     validation_plugin=validation_plugin,
                                                     endorsement_plugin=endorsement_plugin,
                                                     wait_for_event=wait_for_event,
                                                     wait_for_event_timeout=wait_for_event_timeout)
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Chaincode Lifecycle Step 6 Init**
  - Invoke the chaincode

```
async def initialize_chaincode(self):
    """
   Test initialising chaincode
    """
    logger.info("E2E: Chaincode initialisation start")
    chaincode = Chaincode(self.client, CC_NAME)
    org = "org1.example.com"
    org_admin = self.client.get_user(org, "Admin")
    args = ['a', '200', 'b', '300']
    res = await chaincode.invoke(org_admin, self.channel_name, ['peer0.' + org, 'peer1.' + org], args, fcn="Init",
                                 is_init=True, wait_for_event=True)
    self.assertEqual("", res, res)
    time.sleep(2)
    logger.info("E2E: Chaincode initialisation done")
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

```
// Copyright the Hyperledger Fabric contributors. All rights reserved.
//
// SPDX-License-Identifier: Apache-2.0

syntax = "proto3";

option java_package = "org.hyperledger.fabric.protos.peer.lifecycle";
option go_package = "github.com/hyperledger/fabric-protos-go/peer/lifecycle";

package lifecycle;

import "hfc/protos/peer/collection.proto";

// InstallChaincodeArgs is the message used as the argument to
// '_lifecycle.InstallChaincode'.
message InstallChaincodeArgs {
    bytes chaincode_install_package = 1; // This should be a marshaled lifecycle.Chainc
}

// InstallChaincodeArgs is the message returned by
// '_lifecycle.InstallChaincode'.
message InstallChaincodeResult {
    string package_id = 1;
    string label = 2;
}

// QueryInstalledChaincodeArgs is the message used as arguments
// '_lifecycle.QueryInstalledChaincode'
message QueryInstalledChaincodeArgs {
    string package_id = 1;
}

// QueryInstalledChaincodeResult is the message returned by
// '_lifecycle.QueryInstalledChaincode'
message QueryInstalledChaincodeResult {
    string package_id = 1;
    string label = 2;
    map<string, References> references = 3;

    message References {
        repeated Chaincode chaincodes = 1;
    }

    message Chaincode {
        string name = 1;
        string version = 2;
    }
}
```

```
// GetInstalledChaincodePackageArgs is the message used as the argument to
// '_lifecycle.GetInstalledChaincodePackage'.
message GetInstalledChaincodePackageArgs {
    string package_id = 1;
}

// GetInstalledChaincodePackageResult is the message returned by
// '_lifecycle.GetInstalledChaincodePackage'.
message GetInstalledChaincodePackageResult {
    bytes chaincode_install_package = 1;
}

// QueryInstalledChaincodesArgs currently is an empty argument to
// '_lifecycle.QueryInstalledChaincodes'.   In the future, it may be
// extended to have parameters.
message QueryInstalledChaincodesArgs {
}

// QueryInstalledChaincodesResult is the message returned by
// '_lifecycle.QueryInstalledChaincodes'.  It returns a list of installed
// chaincodes, including a map of channel name to chaincode name and version
// pairs of chaincode definitions that reference this chaincode package.
message QueryInstalledChaincodesResult {
    message InstalledChaincode {
        string package_id = 1;
        string label = 2;
        map<string, References> references = 3;
    }

    message References {
        repeated Chaincode chaincodes = 1;
    }

    message Chaincode {
        string name = 1;
        string version = 2;
    }

    repeated InstalledChaincode installed_chaincodes = 1;
}
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

```
========================================================================== 129 passed in 440.48s (0:07:20) ==========================================================================
_____________________________________________________________________________ summary _____________________________________________________________________________
  pylint: commands succeeded
  congratulations :)
# set +o pipefail
>>> Tox test: flake8 ...
# set -o pipefail
# bin_path=.tox/flake8/bin
# export PYTHON=in_path/python
using tox.ini: /home/dex/hyperledger/fabric-sdk-py/tox.ini (pid 42589)
using tox-3.24.4 from /home/dex/hyperledger/fabric-sdk-py/venv/lib/python3.8/site-packages/tox/__init__.py (pid 42589)
flake8 cannot reuse: no previous config /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/.tox-config1
flake8 create: /home/dex/hyperledger/fabric-sdk-py/.tox/flake8
[42596] /home/dex/hyperledger/fabric-sdk-py/.tox$ /home/dex/hyperledger/fabric-sdk-py/venv/bin/python -m virtualenv --no-download --python /home/dex/hyperledger/fabric-sdk-py/venv/bin/python flake8 >flake8/log/flake8-0.log
flake8 installdeps: -r/home/dex/hyperledger/fabric-sdk-py/requirements.txt, -r/home/dex/hyperledger/fabric-sdk-py/requirements-test.txt
[42603] /home/dex/hyperledger/fabric-sdk-py$ /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/bin/python -m pip install -r/home/dex/hyperledger/fabric-sdk-py/requirements.txt -r/home/dex/hyperledger/fabric-sdk-py/requirements-test.txt >.tox/flake8/log/flake8-1.log
flake8 develop-inst: /home/dex/hyperledger/fabric-sdk-py
write config to /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/.tox-config1 as 'ac6aed841d27b6e91a9d78643e0ed9c20d10fac867d1c490bd0fb037ebb2f61d /home/dex/hyperledger/fabric-sdk-py/venv/bin/python\n3.24.4 0 1 0\n0000000000000000000000000000000 -r/home/dex/hyperledger/fabric-sdk-py/requirements.txt\n0000000000000000000000000000000 -r/home/dex/hyperledger/fabric-sdk-py/requirements-test.txt'
[42649] /home/dex/hyperledger/fabric-sdk-py$ /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/bin/python -m pip install --exists-action w -e . >.tox/flake8/log/flake8-2.log
[42662] /home/dex/hyperledger/fabric-sdk-py$ /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/bin/python -m pip freeze >.tox/flake8/log/flake8-3.log
flake8 installed: aiogrpc==1.8,attrs==21.2.0,autopep8==1.6.0,bcrypt==3.2.0,cached-property==1.5.2,certifi==2021.10.8,cffi==1.15.0,charset-normalizer==2.0.7,CouchDB==1.2,coverage==6.1.2,cryptography==35.0.0,docker==3.7.3,docker-compose==1.25.4,docker-pycreds==0.4.0,dockerpty==0.4.1,docopt==0.6.2,-e git+https://github.com/hyperledger/fabric-sdk-py.git@25209f61518873da68d28313582607c29b5bae7d#egg=fabric_sdk_py,flake8==4.0.1,grpcio==1.41.1,grpcio-tools==1.41.1,hkdf==0.0.3,idna==3.3,iniconfig==1.1.1,jsonschema==3.2.0,lark-parser==0.7.1,mccabe==0.6.1,mock==4.0.3,packaging==21.2,paramiko==2.8.0,pluggy==1.0.0,protobuf==3.19.1,py==1.11.0,pycodestyle==2.8.0,pycparser==2.21,pycryptodomex==3.11.0,pyflakes==2.4.0,PyNaCl==1.4.0,pyparsing==2.4.7,pyrsistent==0.18.0,pysha3==1.0b1,pytest==6.2.5,pytest-cov==3.0.0,PyYAML==5.4.1,requests==2.26.0,Rx==3.2.0,six==1.16.0,texttable==1.6.4,toml==0.10.2,tomli==1.2.2,urllib3==1.26.7,websocket-client==0.59.0
flake8 run-test-pre: PYTHONHASHSEED='4178508382'
flake8 run-test: commands[0] | flake8 hfc test test --exclude=protos
[42668] /home/dex/hyperledger/fabric-sdk-py$ /home/dex/hyperledger/fabric-sdk-py/.tox/flake8/bin/flake8 hfc test test --exclude=protos
_____________________________________________________________________________ summary _____________________________________________________________________________
  flake8: commands succeeded
  congratulations :)
# set +o pipefail
>>> Tox test: py36 ...
# set -o pipefail
# bin_path=.tox/py36/bin
# export PYTHON=in_path/python
using tox.ini: /home/dex/hyperledger/fabric-sdk-py/tox.ini (pid 42683)
using tox-3.24.4 from /home/dex/hyperledger/fabric-sdk-py/venv/lib/python3.8/site-packages/tox/__init__.py (pid 42683)
py36 cannot reuse: no previous config /home/dex/hyperledger/fabric-sdk-py/.tox/py36/.tox-config1
py36 create: /home/dex/hyperledger/fabric-sdk-py/.tox/py36
SKIPPED: InterpreterNotFound: python3.6
_____________________________________________________________________________ summary _____________________________________________________________________________
SKIPPED:  py36: InterpreterNotFound: python3.6
  congratulations :)
# set +o pipefail
make[2]: Leaving directory '/home/dex/hyperledger/fabric-sdk-py'
make[1]: Leaving directory '/home/dex/hyperledger/fabric-sdk-py'
```

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Recommendations for future work:**
  - Add more chaincode examples
  - More detailed documentation
  - Track and support latest features of Fabric

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Project Output or Results**:
  - Code Available at:

    https://github.com/hyperledger/fabric-sdk-py
  - Project Link:

    https://wiki.hyperledger.org/display/INTERN/Support+Decentralized+Governance+for+Smart+Contracts+in+Fabric+Python+SDK

Hyperledger Fabric Python SDK

HYPERLEDGER FABRIC

- **Insights Gained**:
  - Communication
    - Managing feedbacks, deliveries and expectations
  - Programming Skills
    - The thing about and not about the written code
- **Take Aways**:
  - Documentation is important!
  - The open-source workflow and paradigm


THANK YOU!