Hyperledger Mentorship Project Presentation

August 2021

Hyperledger Iroha + Cactus - Integration

- **Introduction**
  - **Name**: Han Xu
  - **Location:** Urbana, IL
  - **University:** University of Illinois at Urbana-Champaign
  - **Mentor(s):** Peter Somogyvari, Grzegorz Bazior
  - **Hyperledger Project**: Hyperledger Iroha + Cactus - Integration

Hyperledger Iroha + Cactus - Integration

- **Project Description**:
- Cactus is a blockchain decentralized integration tool that allows users to securely integrate different blockchains. It has a pluggable architecture which makes easy to integrate various blockchain by creating plugins. Hence, Cactus can transfer both assets and data between multiple blockchains.
- Iroha (version 1.x) is great with asset management, and has functionality to store data, which makes Cactus and Iroha a perfect fit!
- Technologies used: Typescript, Node.js, Express, Docker

Hyperledger Iroha + Cactus - Integration

- **Project Objectives:**
  - Obj 1: A documented Iroha connector plugin based on Iroha 1.x for the Cactus project
  - Obj 2: Documented example of integration between multiple (two and more) Iroha's networks with Cactus
  - Obj 3: Documented example of integration between Fabric and Iroha (Fabric plugin for Cactus is already implemented) integration

Hyperledger Iroha + Cactus - Integration

- **Project Deliverables:**
  - Deliverable 1: A documented Iroha connector plugin for Iroha and Cactus integration
  - Deliverable 2: A modified Iroha all-in-one (AIO) dockerfile (and thus docker image)
  - Deliverable 3: One documented example of integration between two Iroha networks with Cactus

Hyperledger Iroha + Cactus - Integration

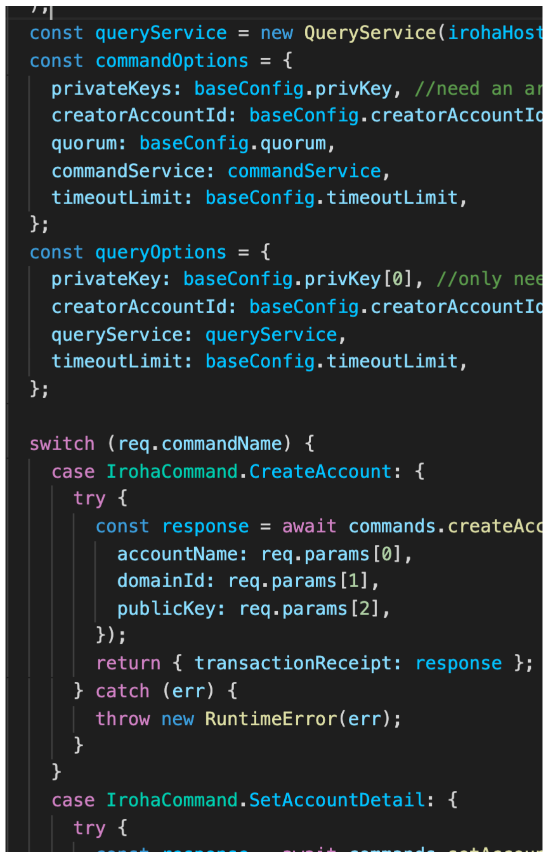
- **Project Execution & Accomplishments**:
- Accomplished: the Iroha connector plugin for Cactus project (Obj1) + example of integration between multiple (two and more) Iroha's networks with Cactus (Obj2)
- Not accomplished: examples of Iroha & Fabric integration (Obj3)
- Most proud: was able to support most of Iroha's commands and queries, as well as validate them.
- Most challenging: understanding Cactus's full architecture (i.e., how Cactus works, and how it connects to Iroha)

Hyperledger Iroha + Cactus - Integration

- **Recommendations for future work:**
- 1. Implement gRPC TLS communication protocol for Iroha ledger.
- 2. Parameters should be more generic in the future so that parameter changes can be done dynamically.
- 3. Currently, utilized a third-party open-source library, “iroha-helper-ts”. But in the future, build our own “iroha-helper-ts” library (outputs tx status and tx hash) based on the Iroha Javascript library, so that it could be upgraded to the latest Javascript library and optimized.
- For a more detailed list, see: https://wiki.hyperledger.org/display/INTERN/Project+Plan+-+HL+Iroha+and+HL+Cactus+Integration

Hyperledger Iroha + Cactus - Integration

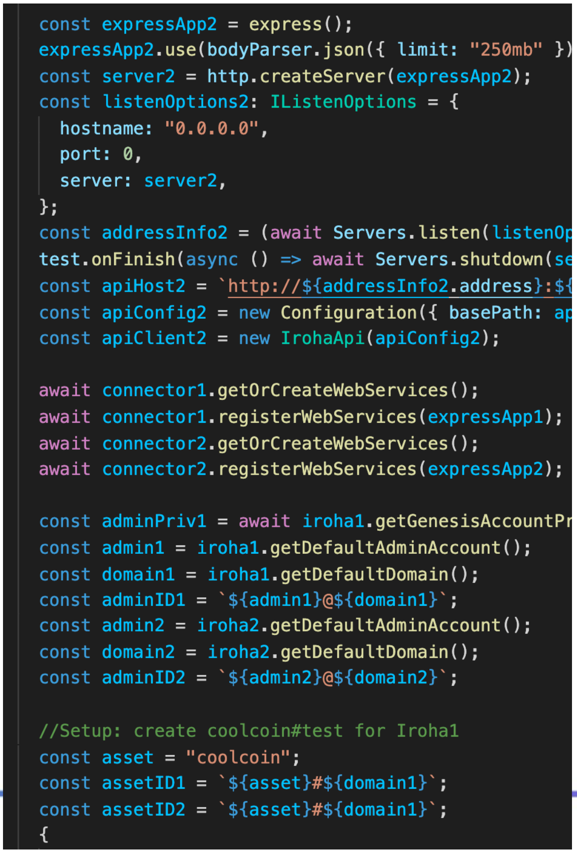
- **Project Output or Results:**
- 1st PR: https://github.com/hyperledger/cactus/pull/1169 (Obj1 + Obj2)
- 2nd PR: https://github.com/hyperledger/cactus/pull/1183 (Deliverable 2)



```
const queryService = new QueryService(irohaHost
const commandOptions = {
  privateKeys: baseConfig.privKey, //need an ar
  creatorAccountId: baseConfig.creatorAccountIc
  quorum: baseConfig.quorum,
  commandService: commandService,
  timeoutLimit: baseConfig.timeoutLimit,
};
const queryOptions = {
  privateKey: baseConfig.privKey[0], //only nee
  creatorAccountId: baseConfig.creatorAccountIc
  queryService: queryService,
  timeoutLimit: baseConfig.timeoutLimit,
};

switch (req.commandName) {
  case IrohaCommand.CreateAccount: {
    try {
      const response = await commands.createAcc
        accountName: req.params[0],
        domainId: req.params[1],
        publicKey: req.params[2],
      });
      return { transactionReceipt: response };
    } catch (err) {
      throw new RuntimeError(err);
    }
  }
  case IrohaCommand.SetAccountDetail: {
    try {
```

Deal with Iroha commands/queries



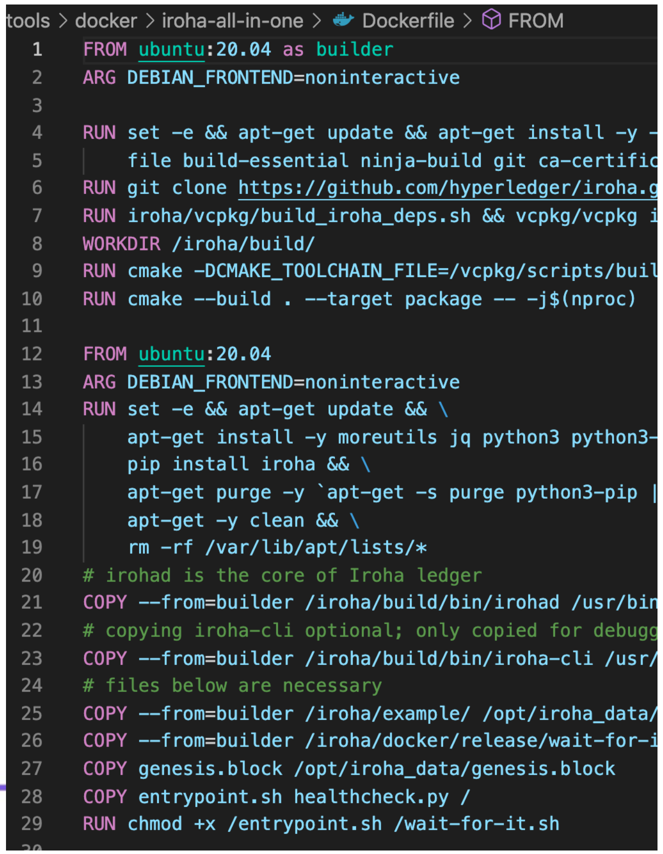
```
const expressApp2 = express();
expressApp2.use(bodyParser.json({ limit: "250mb" })
const server2 = http.createServer(expressApp2);
const listenOptions2: IListenOptions = {
  hostname: "0.0.0.0",
  port: 0,
  server: server2,
};
const addressInfo2 = (await Servers.listen(listenOp
test.onFinish(async () => await Servers.shutdown(se
const apiHost2 = `http://${addressInfo2.address}:${
const apiConfig2 = new Configuration({ basePath: ap
const apiClient2 = new IrohaApi(apiConfig2);

await connector1.getOrCreateWebServices();
await connector1.registerWebServices(expressApp1);
await connector2.getOrCreateWebServices();
await connector2.registerWebServices(expressApp2);

const adminPriv1 = await iroha1.getGenesisAccountPr
const admin1 = iroha1.getDefaultAdminAccount();
const domain1 = iroha1.getDefaultDomain();
const adminID1 = `${admin1}@${domain1}`;
const admin2 = iroha2.getDefaultAdminAccount();
const domain2 = iroha2.getDefaultDomain();
const adminID2 = `${admin2}@${domain2}`;

//Setup: create coolcoin#test for Iroha1
const asset = "coolcoin";
const assetID1 = `${asset}#${domain1}`;
const assetID2 = `${asset}#${domain1}`;
{
```

Iroha Cactus integration example (iroha node 1 transfers to node2)



tools > docker > iroha-all-in-one > Dockerfile > FROM

```
FROM ubuntu:20.04 as builder
ARG DEBIAN_FRONTEND=noninteractive

RUN set -e && apt-get update && apt-get install -y -
    file build-essential ninja-build git ca-certific
RUN git clone https://github.com/hyperledger/iroha.g
RUN iroha/vcpkg/build_iroha_deps.sh && vcpkg/vcpkg i
WORKDIR /iroha/build/
RUN cmake -DCMAKE_TOOLCHAIN_FILE=/vcpkg/scripts/buil
RUN cmake --build . --target package -- -j$(nproc)

FROM ubuntu:20.04
ARG DEBIAN_FRONTEND=noninteractive
RUN set -e && apt-get update && \
    apt-get install -y moreutils jq python3 python3-
    pip install iroha && \
    apt-get purge -y `apt-get -s purge python3-pip |
    apt-get -y clean && \
    rm -rf /var/lib/apt/lists/*
# irohad is the core of Iroha ledger
COPY --from=builder /iroha/build/bin/irohad /usr/bin
# copying iroha-cli optional; only copied for debugg
COPY --from=builder /iroha/build/bin/iroha-cli /usr/
# files below are necessary
COPY --from=builder /iroha/example/ /opt/iroha_data/
COPY --from=builder /iroha/docker/release/wait-for-i
COPY genesis.block /opt/iroha_data/genesis.block
COPY entrypoint.sh healthcheck.py /
RUN chmod +x /entrypoint.sh /wait-for-it.sh
```

Part of the Iroha AIO Dockerfile

Hyperledger Iroha + Cactus - Integration

- **Insights Gained:**
- Learnt about the workflow of open source project is developed

- **Advice:**
- Have a solid plan in the beginning so that you can follow
- Actively reach out to mentors to seek advice and feedback
- Learning from the open-source community is also very helpful

Hyperledger Iroha + Cactus - Integration

- Big thank to my mentors(Peter & Greg) and members from the community
- Also thank LFX for hosting and sponsoring this event

Hyperledger Iroha + Cactus - Integration

- Questions

THANK YOU!