HYPERLEDGER CALIPER

AUTOMOBILE NETWORK

Introduction:

Hyperledger caliper is used to test the performance of the blockchain and evaluate it based on several metrics like transactions per second, successful and failed transaction, latency etc. In this caliper test we are testing hyperledger caliper to asses it's performance. We have deployed a car-contract smart contract which contains the functionalities of car dealership like creating and selling the car.

Steps to be performed:

Step1:

Run the Automobile -network and deploy the chaincode KBA-Automobile.

Step2:

Create a folder called caliper-workspace in the Automobile-network directory.

Step3:

create three separate folders inside the caliper-workspace directory named

‘Network’, ‘Workload’, ‘Benchmark’.

Step4:

Inside this network folder is where we will define the network specifications of the fabric network where we will specify which organization are we going to use to test the blockchain and their private and public key.

Now let's create a file called **networkconfig.yaml** inside Network directory inside caliper-workspace directory. This is how the networkconfig.yaml looks like.

```
name: Caliper test
version: "2.0.0"


caliper:
  blockchain: fabric


channels:
- channelName: autochannel
  contracts:
   - id: KBA-Automobile


organizations:
- mspid: ManufacturerMSP
  identities:
    certificates:
    - name: 'User1'
      clientPrivateKey:
        path: 
'../Automobile-network/organizations/peerOrganizations
/manufacturer.auto.com/users/
User1@manufacturer.auto.com/msp/keystore/
ab817efa109c38dced849fb26cf8048d746b80f693caf3cc26699a
71bf36fcc5_sk'
      clientSignedCert:
        path: 
'../Automobile-network/organizations/peerOrganizations
/manufacturer.auto.com/users/
User1@manufacturer.auto.com/msp/signcerts/cert.pem'
  peers:
      - endpoint: localhost:7051
        grpcOptions:
          ssl-target-name-override: 
peer0.manufacturer.auto.com
          grpc.keepalive_time_ms: 600000
        tlsCACerts:
          path: 
'../Automobile-network/organizations/peerOrganizations
/manufacturer.auto.com/peers/
peer0.manufacturer.auto.com/tls/tlscacerts/tls-
localhost-7054-ca-manufacturer.pem'
```

Note: Replace the path with the location in your local system.

Note: Every time the network is restarted there will be a change in the file name

Step 5:

Create a file called **carbenchmark.yaml** in the Benchmark directory. In the benchmark file we have to specify the caliper operations of what exactly you want the no of workers to perform the operation how many times you want to execute the transaction , the duration of the transaction etc. This is an example of carbenchmark.yaml file

```
test:
   name: basic-contract-benchmark
   description: test benchmark
   workers:
     number: 2
   rounds:
     - label: readcar
       description: Read car benchmark
       txDuration: 30
       rateControl:
         type: fixed-load
         opts:
           transactionLoad: 2
       workload:
         module:  Workload/readcar.js
         arguments:
           assets: 10
           contractId: KBA-Automobile
```

Step 6:

Now create a file named **readcar.js** in the workload directory. Workload file defines the operations it needs to perform for the chaincode and the SUT(system under test). Here we will specify the modules to performed in caliper assessment. Below is the code for readcar.js

```
'use strict';
const { WorkloadModuleBase } =
require('@hyperledger/caliper-core');


class MyWorkload extends WorkloadModuleBase {
  constructor() {
      super();
  }


  async initializeWorkloadModule(workerIndex,
totalWorkers, roundIndex, roundArguments, sutAdapter,
sutContext) {
       await
super.initializeWorkloadModule(workerIndex,
totalWorkers, roundIndex, roundArguments, sutAdapter,
sutContext);
       for (let i=0; i<this.roundArguments.assets; i+
+) {
           const carId = `${this.workerIndex}_${i}`;
           console.log(`Worker ${this.workerIndex}:
Creating car ${carId}`);
           const request = {
               contractId:
```

```
this.roundArguments.contractId,
               contractFunction: 'CreateCar',
               invokerIdentity: 'User1',
               contractArguments: [carId,'SUV','XUV
700','Blue','03/05/23','Mahindra'],
               readOnly: false
           };


         await this.sutAdapter.sendRequests(request);
       }
   }


   async submitTransaction() {
       const randomId = `${this.workerIndex}_$
{Math.floor(Math.random()*this.roundArguments.assets)}
`;
       const myArgs = {
           contractId: this.roundArguments.contractId,
           contractFunction: 'ReadCar',
           invokerIdentity: 'User1',
           contractArguments:  [randomId],
           readOnly: true
       };


       await this.sutAdapter.sendRequests(myArgs);
   }


   async cleanupWorkloadModule() {
       for (let i=0; i<this.roundArguments.assets; i+
+) {
```

```
           const carId = `${this.workerIndex}_${i}`;
           console.log(`Worker ${this.workerIndex}:
Deleting car ${carId}`);
           const request = {
               contractId:
this.roundArguments.contractId,
               contractFunction: 'DeleteCar',
               invokerIdentity: 'User1',
               contractArguments: [carId],
               readOnly: false
           };
            await
this.sutAdapter.sendRequests(request);
       }
   }
}


function createWorkloadModule() {
   return new MyWorkload();
}
 module.exports.createWorkloadModule =
createWorkloadModule;
```

Step 7:

After saving these files go to caliper-workspace directory and install the caliper cli with the following command

```
npm install --only=prod @hyperledger/caliper-cli@0.5.0
```

Specify the sut here it is fabric 2.4

```
npx caliper bind --caliper-bind-sut fabric:2.4
```

Step 8:

```
npx caliper launch manager --caliper-workspace ./ --caliper-networkconfig Network/networkconfig.yaml --caliper-benchconfig Benchmark/carbenchmark.yaml --caliper-flow-only-test --caliper-fabric-gateway-enabled
```

KBA