

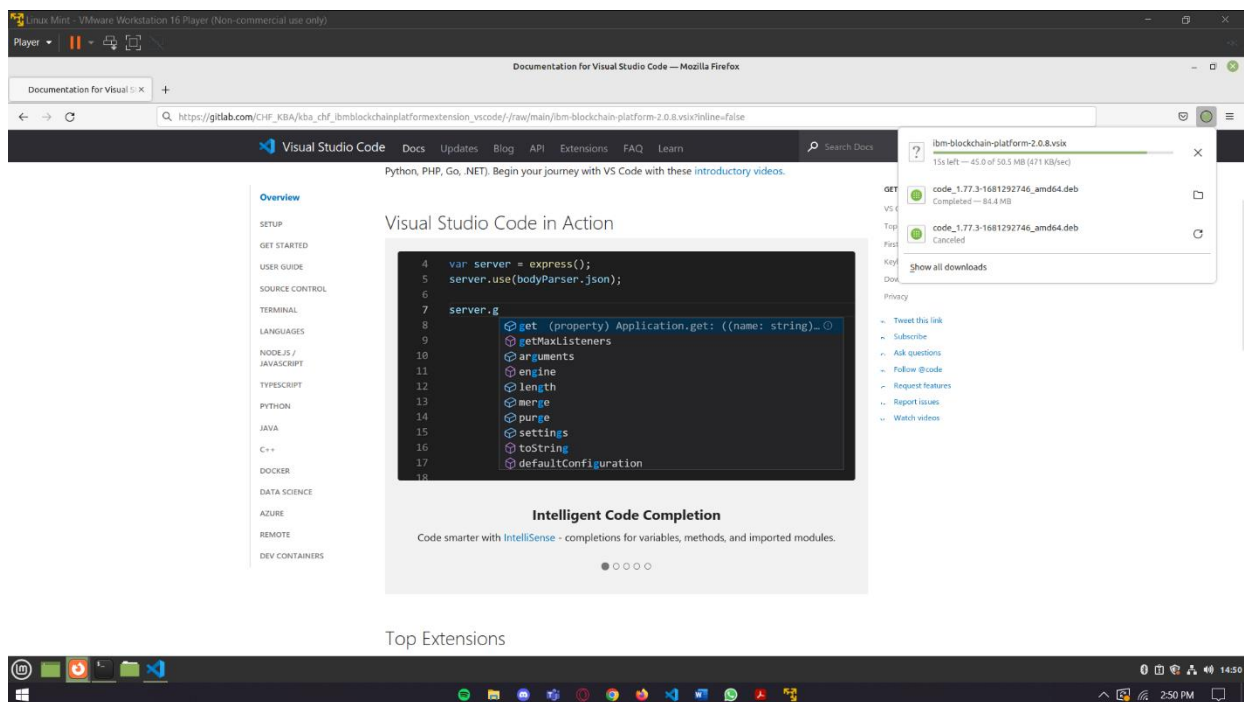
NAME: ANKIT KUMAR JHA  
REG.NO.20BPS1050

## EXERCISE 5

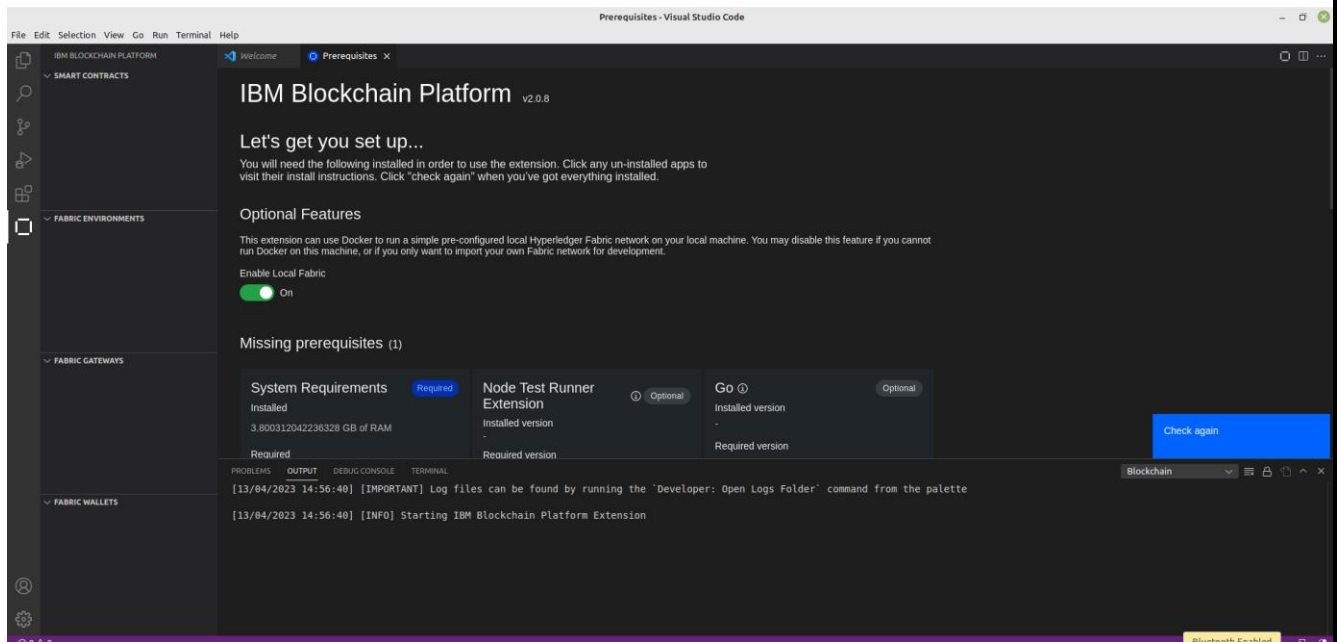
### Hyperledger fabric using IBM Blockchain platform

#### Downloading Prerequisites

- Visual Studio Code
- IBM Blockchain Platform extension
- 



#### Installing Extension



## Configuration

```
$ export MICROFAB_CONFIG='{
```

```
> "port": 8080,
> "endorsing_organizations":[
> {
>   "name": "ProducersOrg"
> },
> {
>   "name": "SellersOrg"
> },
> ],
> "channels":[
> {
>   "name": "mango-channel",
>   "endorsing_organizations":[
>     "ProducersOrg",
>     "SellersOrg"
>   ]
> }
> ]
> }
```

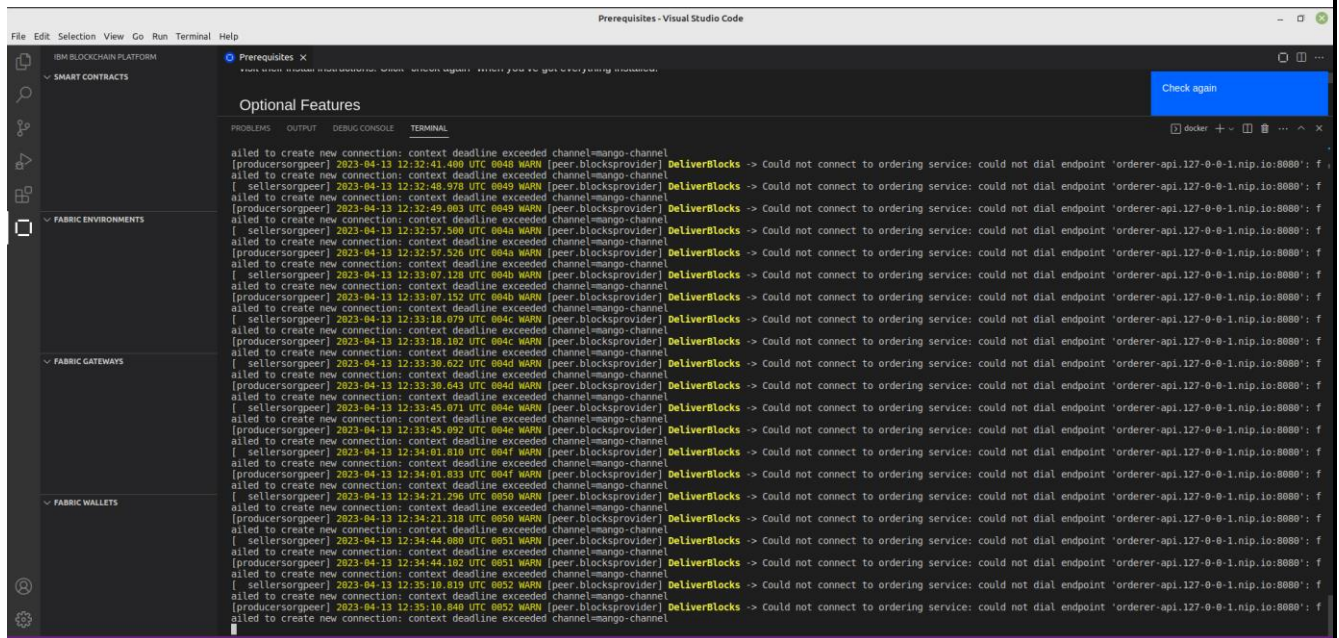
## Running the Network

```
$ docker run -e MICROFAB_CONFIG -p 8080:8080 10mccomm/10p-microfab
```

```

Unable to find image 'ibmcom/ibp-microfab:latest' locally
latest: Pulling from ibmcom/ibp-microfab
d6e3788a121c: Pull complete
b336a633f9e2: Pull complete
b25deee7e50f: Pull complete
9041dff50abb: Downloading [=====>] 30.7MB/196MB
c50c12d7db31: Download complete
5186d5863148: Download complete
531dcea8d07e: Download complete
690939f6038f: Download complete
f24a19b34ee1: Downloading [=>] 12.94MB/363.1MB
576db8757925: Downloading 1.619MB
734dce7f474b: Waiting
eb98c0bd8a72: Waiting
65606c3065d6: Waiting
2785a9abab40: Waiting
78f82eb67606: Waiting

```



Smart Contract project

Package.json:

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of the 'KBA-mango-contract' project. The main editor area shows the 'package.json' file with the following content:

```
1  {}
2  {
3    "name": "KBA-mango-contract",
4    "version": "0.0.1",
5    "description": "My Smart Contract",
6    "main": "index.js",
7    "engines": {
8      "node": ">=8",
9      "npm": ">=5"
10   },
11   "scripts": {
12     "lint": "eslint .",
13     "pretest": "npm run lint",
14     "test": "nyc mocha --recursive",
15     "start": "fabric-chaincode-node start"
16   },
17   "engineStrict": true,
18   "author": "John Doe",
19   "license": "Apache-2.0",
20   "dependencies": {
21     "fabric-contract-api": "^2.4.1",
22     "fabric-shim": "^2.4.1"
23   },
24   "devDependencies": {
25     "chai": "^4.2.0",
26     "chai-as-promised": "^7.1.1",
27     "eslint": "^8.7.0",
28     "mocha": "^9.2.0",
29     "nyc": "^15.0.0",
30     "sinon": "^9.0.1",
31     "sinon-chai": "^3.5.0",
32     "winston": "^3.2.1"
33   },
34   "nyc": {
35     "exclude": [
36       ".eslintrc.js",
37       "coverage/**",
38       "test/**"
39     ],
40     "reporter": [
41       "text-summary",
42       "html"
```

## Index.js

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of the 'KBA-mango-contract' project. The main editor area shows the 'index.js' file with the following content:

```
1  /
2  * SPDX-License-Identifier: Apache-2.0
3  */
4  'use strict';
5
6  const MangoContract = require('../lib/mango-contract');
7
8  module.exports.MangoContract = MangoContract;
9  module.exports.contracts = [ MangoContract ];
10
11
```

## Mango-contract.js

```
1  /*
2   * SPDX-License-Identifier: Apache-2.0
3   */
4
5  'use strict';
6
7  const { Contract } = require('fabric-contract-api');
8
9  class MangoContract extends Contract {
10
11    async mangoExists(ctx, mangoId) {
12      const buffer = await ctx.stub.getState(mangoId);
13      return (!!buffer && buffer.length > 0);
14    }
15
16    async createMango(ctx, mangoId, value) {
17      const exists = await this.mangoExists(ctx, mangoId);
18      if (exists) {
19        throw new Error('The mango ${mangoId} already exists');
20      }
21      const asset = { value };
22      const buffer = Buffer.from(JSON.stringify(asset));
23      await ctx.stub.putState(mangoId, buffer);
24    }
25
26    async readMango(ctx, mangoId) {
27      const exists = await this.mangoExists(ctx, mangoId);
28      if (!exists) {
29        throw new Error('The mango ${mangoId} does not exist');
30      }
31      const buffer = await ctx.stub.getState(mangoId);
32      const asset = JSON.parse(buffer.toString());
33      return asset;
34    }
35
36    async updateMango(ctx, mangoId, newValue) {
37      const exists = await this.mangoExists(ctx, mangoId);
38      if (!exists) {
39        throw new Error('The mango ${mangoId} does not exist');
40      }
41      const asset = { value: newValue };
42      const buffer = Buffer.from(JSON.stringify(asset));
```

## Adding More Functionalities

### Mango-contract.js

```
38    const buffer = await ctx.stub.getState(mangoId);
39    const asset = JSON.parse(buffer.toString());
40    return asset;
41  }
42
43  async updateMango(ctx, mangoId, newValue) {
44    const exists = await this.mangoExists(ctx, mangoId);
45    if (!exists) {
46      throw new Error('The mango ${mangoId} does not exist');
47    }
48    const asset = {
49      BatchNumber: batchNumber,
50      Producer: producer,
51      OwnedBy: owner,
52      Quantity: quantity,
53      Price: price,
54    };
55    const buffer = Buffer.from(JSON.stringify(asset));
56    await ctx.stub.putState(mangoId, buffer);
57  }
58
59  async deleteMango(ctx, mangoId) {
60    const exists = await this.mangoExists(ctx, mangoId);
61    if (!exists) {
62      throw new Error('The mango ${mangoId} does not exist');
63    }
64    await ctx.stub.deleteState(mangoId);
65  }
66
67  async sellMangos(ctx, mangoId, ownerName) {
68    const exists = await this.mangoExists(ctx, mangoId);
69    if (!exists) {
70      throw new Error('The apple ${mangoId} does not exist');
71    }
72    const asset = { currentOwner: ownerName };
73    const buffer = Buffer.from(JSON.stringify(asset));
74    await ctx.stub.putState(mangoId, buffer);
75  }
76
77  module.exports = MangoContract;
```

```

/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Contract } = require('fabric-contract-api');

class MangoContract extends Contract {

  async mangoExists(ctx, mangoId) {
    const buffer = await ctx.stub.getState(mangoId);
    return (!!buffer && buffer.length > 0);
  }

  async createMango(ctx, mangoId, value) {
    const exists = await this.mangoExists(ctx, mangoId);
    if (exists) {
      throw new Error(`The mango ${mangoId} already exists`);
    }
    const asset = {
      ID: mangoId,
      BatchNumber: batchNumber,
      Producer: producer,
      OwnedBy: producer,
      Quantity: quantity,
      Price: price,
    };
    const buffer = Buffer.from(JSON.stringify(asset));
    await ctx.stub.putState(mangoId, buffer);
  }

  async readMango(ctx, mangoId) {
    const exists = await this.mangoExists(ctx, mangoId);
    if (!exists) {
      throw new Error(`The mango ${mangoId} does not exist`);
    }
    const buffer = await ctx.stub.getState(mangoId);
    const asset = JSON.parse(buffer.toString());
    return asset;
  }

  async updateMango(ctx, mangoId, newValue) {
    const exists = await this.mangoExists(ctx, mangoId);
    if (!exists) {
      throw new Error(`The mango ${mangoId} does not exist`);
    }
    const asset = {
      BatchNumber: batchNumber,

```

```

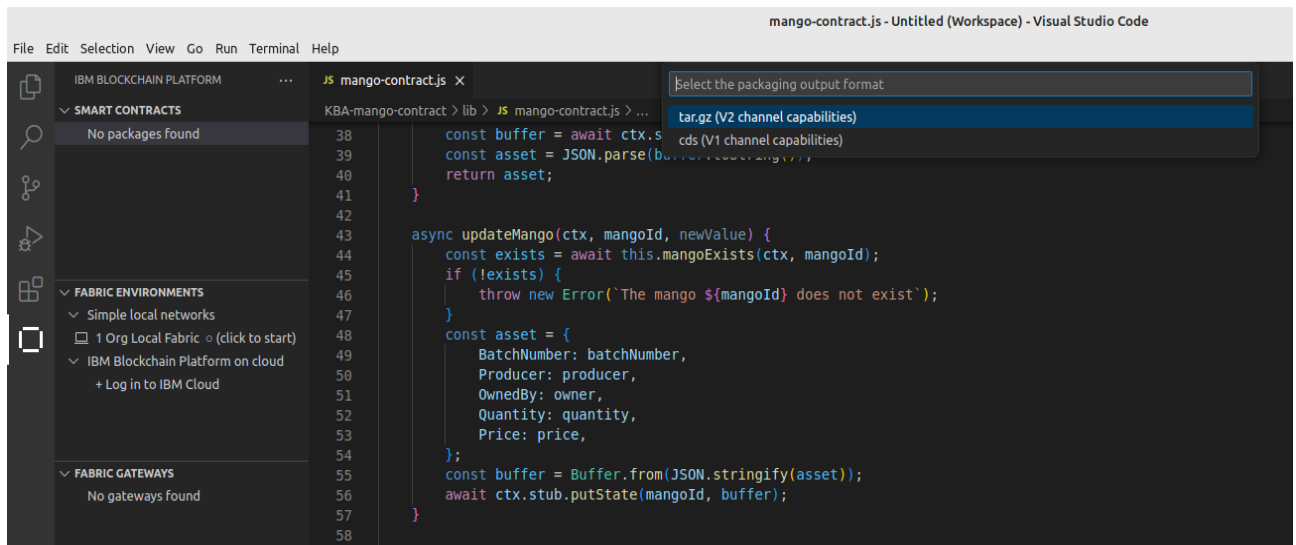
        Producer: producer,
        OwnedBy: owner,
        Quantity: quantity,
        Price: price,
    };
    const buffer = Buffer.from(JSON.stringify(asset));
    await ctx.stub.putState(mangoId, buffer);
}

async deleteMango(ctx, mangoId) {
    const exists = await this.mangoExists(ctx, mangoId);
    if (!exists) {
        throw new Error(`The mango ${mangoId} does not exist`);
    }
    await ctx.stub.deleteState(mangoId);
}

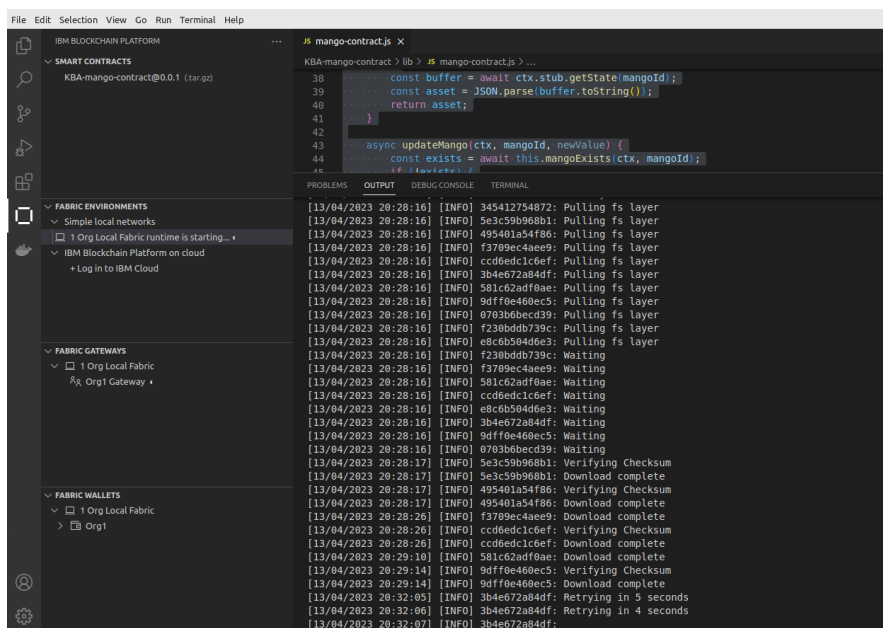
async sellMangos(ctx, mangoId, ownerName) {
    const exists = await this.mangoExists(ctx, mangoId);
    if (!exists) {
        throw new Error(`The apple ${mangoId} does not exist`);
    }
    const asset = { currentOwner: ownerName };
    const buffer = Buffer.from(JSON.stringify(asset));
    await ctx.stub.putState(mangoId, buffer);
}
}

module.exports = MangoContract;

```

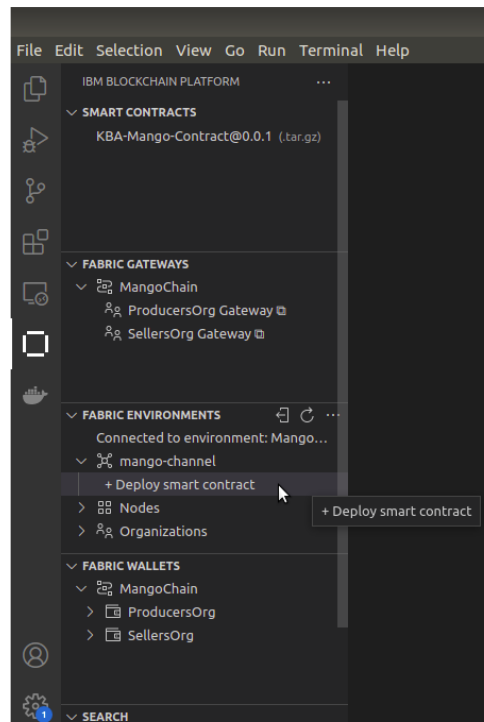


## Fabric Environments

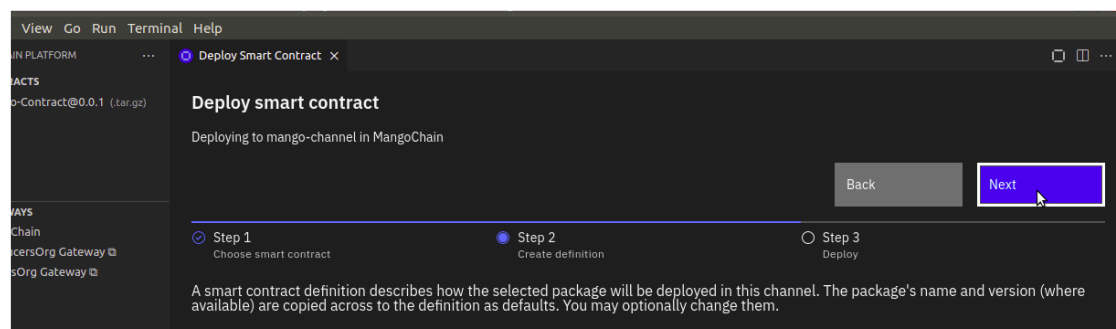


## Deploy Smart Contract



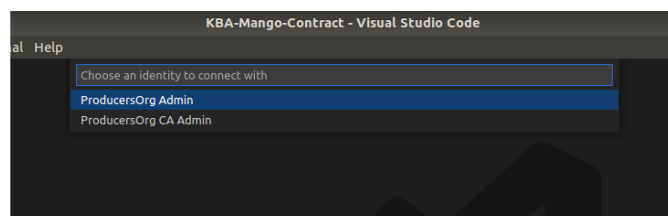


## Create Smart Contract Definition

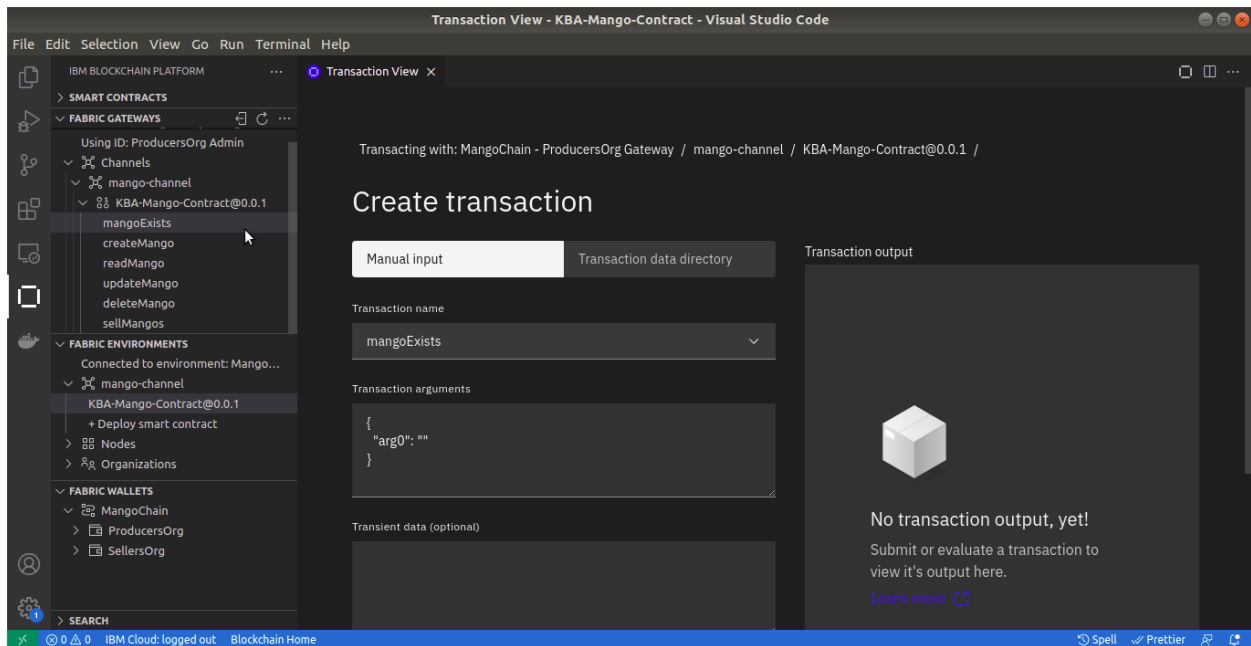


## Making Transactions

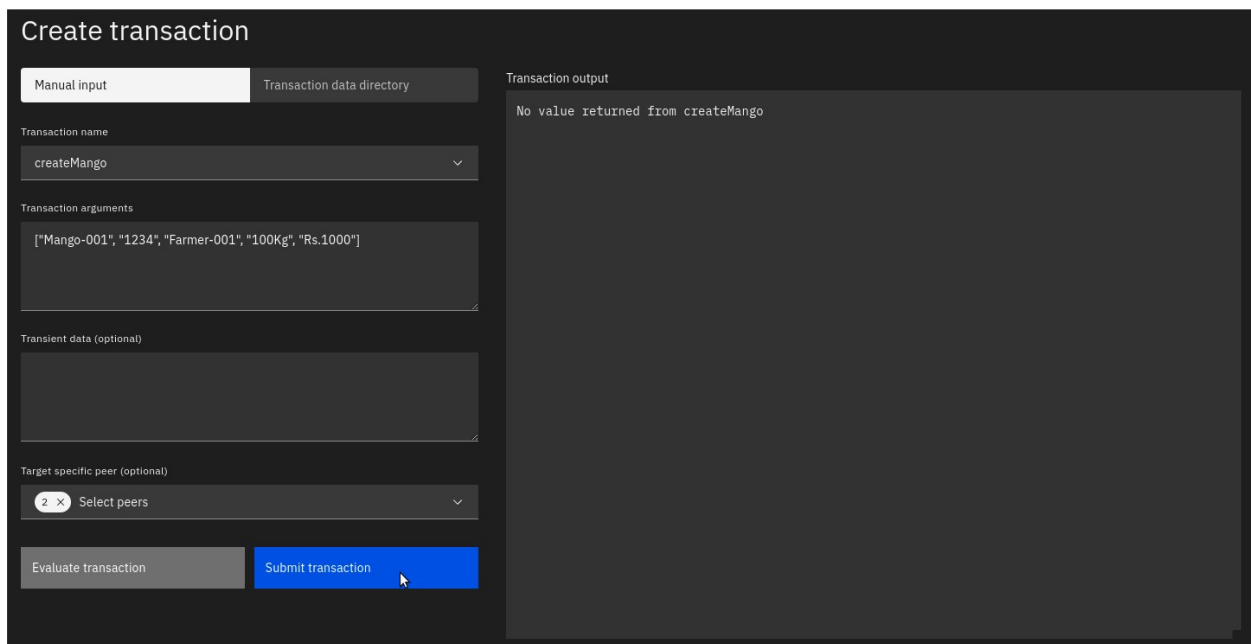
### Fabric Gateway



### mangoExists



createMango



readMango

## Create transaction

Manual input

Transaction data directory

Transaction name

readMango

Transaction arguments

["Mango-001"]

Transient data (optional)

Target specific peer (optional)

2 X Select peers

Evaluate transaction

Submit transaction

Transaction output

```
Returned value from readMango: {"BatchNumber": "1234", "ID": "Mango-001", "OwnedBy": "Farmer-001", "Price": "Rs.1000", "Producer": "Farmer-001", "Quantity": "100Kg"}
```