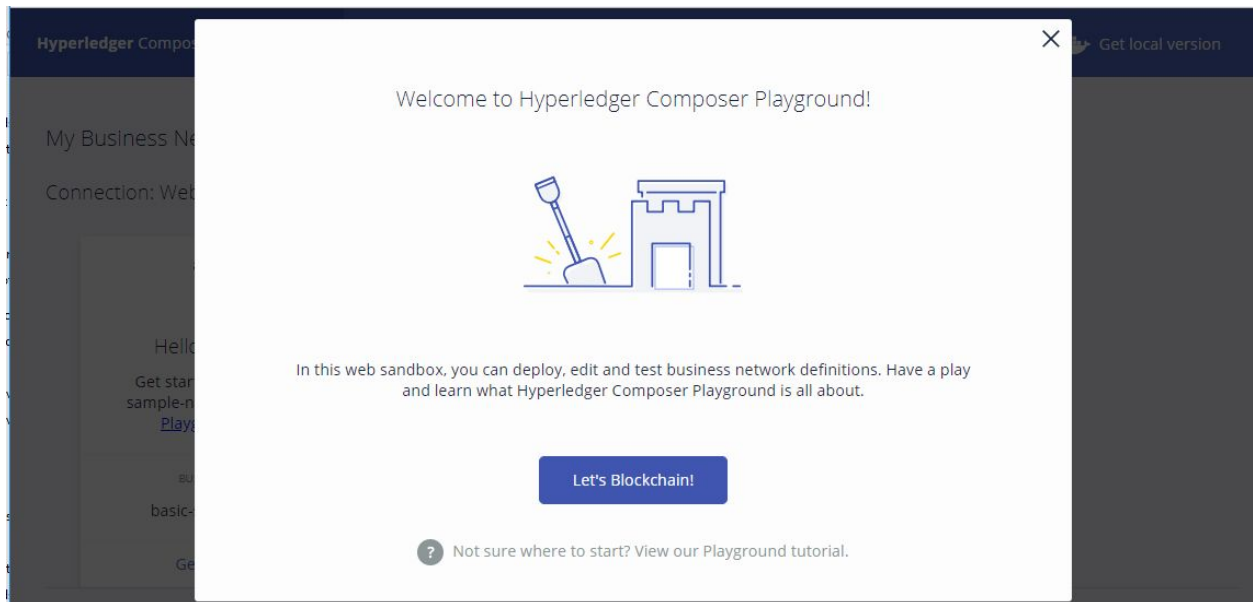


# Hyperledger Composer Playground Example

- Creating the business network

## Step 1. Open the Hyperledger Composer Playground

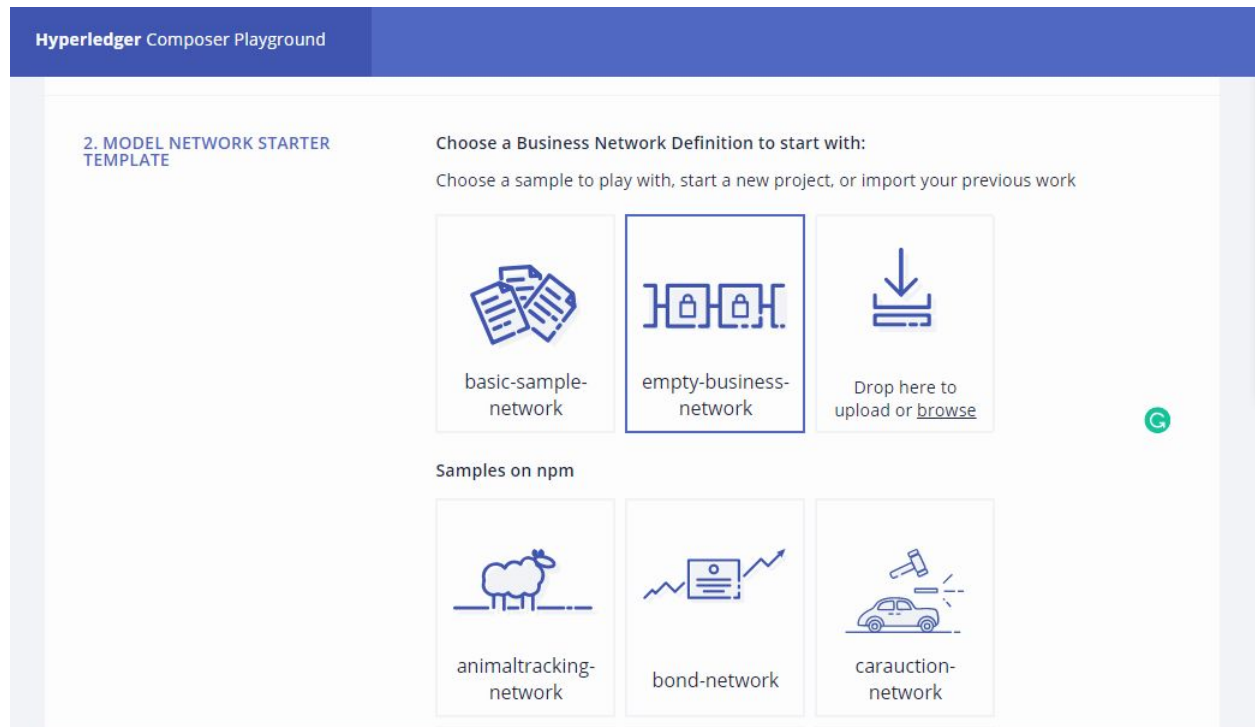


## Step 2. Create a new business network

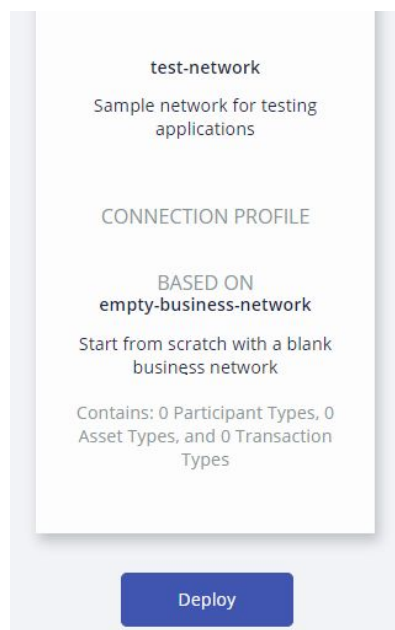
Step 2.1: First enter the basic information:

A screenshot of the "Deploy New Business Network" form in the Hyperledger Composer Playground. The form is titled "Deploy New Business Network" and has a back arrow and "My Wallet" label on the top left. On the top right, there is a question mark icon and the text "Not sure w". The form is divided into sections. The first section is "1. BASIC INFORMATION". It contains three rows of input fields. The first row is labeled "Give your new Business Network a name:" and has a text input field containing "test-network". The second row is labeled "Describe what your Business Network will be used for:" and has a text input field containing "Sample network for testing applications". The third row is labeled "Give the network admin card that will be created a name" and has a text input field containing "eg. admin@test-network". A green circular icon with a white 'G' is visible on the right side of the form.

Step 2.2: Now choose the empty-business network template:



Step 2.3: Now click on deploy to create the network:





### Step 3. Connect to the business network

**Hyperledger** Composer Playground

My Business Networks

A

admin@test-net




USER ID

admin

BUSINESS NETWORK

test-net

Connect now →



Deploy a new business network

## Step 4. Add a model file into your business network

```
/**
 * My business network
 */
namespace org.example.testnetwork
asset Commodity identified by tradingSymbol {
  o String tradingSymbol
  o String description
  o String mainExchange
  o Double quantity
  --> Trader owner
}
participant Trader identified by tradeId {
  o String tradeId
  o String firstName
  o String lastName
}
transaction Trade {
  --> Commodity commodity
  --> Trader newOwner
}
```

The screenshot shows the 'Web test-net' application interface. The top navigation bar includes 'Web test-net', 'Define', 'Test', and a user profile 'admin'. The left sidebar contains a 'FILES' section with links to 'About' (README.md, package.json), 'Model File' (models/model.cto), and 'Access Control' (permissions.acl). Below the files section are buttons for 'Add a file...' and 'Export'. The main area is titled 'Model File models/model.cto' and displays the following code in a dark-themed editor:

```
15 namespace org.example.testnetwork
16
17 asset Commodity identified by tradingSymbol {
18   o String tradingSymbol
19   o String description
20   o String mainExchange
21   o Double quantity
22   --> Trader owner
23 }
24 participant Trader identified by tradeId {
25   o String tradeId
26   o String firstName
27   o String lastName
28 }
29 transaction Trade {
30   --> Commodity commodity
31   --> Trader newOwner
32 }
```

At the bottom of the editor, a green checkmark icon is followed by the text 'Everything looks good!' and a smaller line of text: 'Any problems detected in your code would be reported here'.

## Step 5. Add a transaction script file

```
/**
 * Track the trade of a commodity from one trader to another
 * @param {org.example.testnetwork.Trade} trade - the trade
to be processed
 * @transaction
 */
async function tradeCommodity(trade) {
    trade.commodity.owner = trade.newOwner;
    let assetRegistry = await
getAssetRegistry('org.example.mynetwork.Commodity');
    await assetRegistry.update(trade.commodity);
}
```

The screenshot shows the 'Web test-net' interface. On the left, a sidebar lists files: 'About' (README.md, package.json), 'Model File' (models/model.cto), 'Script File' (lib/script.js), and 'Access Control' (permissions.acl). The 'Script File' is selected. The main area shows the script file 'lib/script.js' with the following code:

```
1
2 /**
3  * Track the trade of a commodity from one trader to another
4  * @param {org.example.testnetwork.Trade} trade - the trade to be processed
5  * @transaction
6  */
7  async function tradeCommodity(trade) {
8      trade.commodity.owner = trade.newOwner;
9      let assetRegistry = await getAssetRegistry('org.example.mynetwork.Commodity');
10     await assetRegistry.update(trade.commodity);
11 }
```

At the bottom, a status bar indicates 'Everything looks good!' and 'Any problems detected in your code would be reported here'.

## Step 6. Add a permissions file

Web test-net

DefineTest

admin

FILES

About

README.md, package.json

Model File

models/model.cto

Script File

lib/script.js

Access Control

permissions.acl

Add a file...

Export

UPDATE NETWORK

From: 0.0.1

To: 0.0.2-deploy.0

ACL File permissions.acl

```

12  * limitations under the License.
13  */
14
15  rule NetworkAdminUser {
16    description: "Grant business network administrators full access to user resources"
17    participant: "org.hyperledger.composer.system.NetworkAdmin"
18    operation: ALL
19    resource: "***"
20    action: ALLOW
21  }
22
23  rule NetworkAdminSystem {
24    description: "Grant business network administrators full access to system resources"
25    participant: "org.hyperledger.composer.system.NetworkAdmin"
26    operation: ALL
27    resource: "org.hyperledger.composer.system.*)"
28    action: ALLOW
29  }

```

Everything looks good!

Any problems detected in your code would be reported here

## Step 7. Deploy the created business network

Web test-net

DefineTest

FILES

About

README.md, package.json

Model File

models/model.cto

Script File

lib/script.js

Access Control

permissions.acl

Add a file...

Export

UPDATE NETWORK

From: 0.0.2-deploy.0

To: 0.0.2-deploy.1

Deploy changes

About File package.json

```

1  {
2    "name": "test-net",
3    "author": "author",
4    "description": "Sample business network",
5    "version": "0.0.2-deploy.1",
6    "devDependencies": {
7      "browserfs": "^1.2.0",
8      "chai": "^3.5.0",
9      "composer-admin": "latest",
10     "composer-cli": "latest",
11     "composer-client": "latest",
12     "composer-connector-embedded": "latest",
13     "eslint": "^3.6.1",
14     "istanbul": "^0.4.5",
15     "jsdoc": "^3.4.1",
16     "mkdirp": "^0.5.1",
17     "mocha": "^3.2.0",
18     "moment": "^2.19.3"

```

Everything looks good!

Any problems detected in your code would be reported here

Legal

GitHub

Playground v0.19.

- Test the created business network

## Step 1. Creating Participants

Participant 1:

```
{
  "$class": "org.example.testnetwork.Trader",
  "tradeId": "trader1",
  "firstName": "John",
  "lastName": "Doe"
}
```

Participant 2:

```
{
  "$class": "org.example.testnetwork.Trader",
  "tradeId": "trader2",
  "firstName": "Tyrion",
  "lastName": "Sinnister"
}
```

Web test-net

DefineTest

admin

PARTICIPANTS

Trader

ASSETS

Commodity

TRANSACTIONS

All Transactions

Submit Transaction

Participant registry for org.example.testnetwork.Trader

+ Create New Participant

ID	Data
2407	<div>{   "\$class": "org.example.testnetwork.Trader",   "tradeId": "2407",   "firstName": "John",   "lastName": "Doe" }</div>
5879	<div>{   "\$class": "org.example.testnetwork.Trader",   "tradeId": "5879",   "firstName": "Tyrion",   "lastName": "Sinnister" }</div>

## Step 2. Create Asset

Asset 1:

```
{
  "$class": "org.example.testnetwork.Commodity",
  "tradingSymbol": "XYZ",
  "description": "Test Token",
  "mainExchange": "token2",
  "quantity": 100,
  "owner": "resource:org.example.testnetwork.Trader#trader1"
}
```

Web test-net

Define Test

admin

PARTICIPANTS

Trader

ASSETS

Commodity

TRANSACTIONS

All Transactions

Submit Transaction

Asset registry for org.example.testnetwork.Commodity

+ Create New Asset

ID	Data
5660	<div>{   "\$class": "org.example.testnetwork.Commodity",   "tradingSymbol": "5660",   "description": "Test Token",   "mainExchange": "token2",   "quantity": 100, }</div> <div>Show All</div>
9412	<div>{   "\$class": "org.example.testnetwork.Commodity",   "tradingSymbol": "9412",   "description": "Test token",   "mainExchange": "token1",   "quantity": 10,   "owner": "resource:org.example.testnetwork.Trader#3589" }</div> <div>Collapse</div>



## Step 3. Transfer assets

```
{
  "$class": "org.example.samplenetwork.Trade",
  "commodity":
    "resource:org.example.samplenetwork.Commodity#GOT",
  "newOwner":
    "resource:org.example.samplenetwork.Trader#trader2"
}
```

The screenshot displays a web application interface for a sample network. The top navigation bar includes 'Web sample-network', 'Define', 'Test', and a user profile 'admin'. A left sidebar contains a menu with 'PARTICIPANTS' (Trader), 'ASSETS' (Commodity), and 'TRANSACTIONS' (All Transactions). A 'Submit Transaction' button is located at the bottom of the sidebar. The main content area features a 'Historian Record' window with a close button (X). This window has two tabs: 'Transaction' (selected) and 'Events (0)'. The 'Transaction' tab displays a JSON object in a dark-themed code editor:

```
1 {
2   "$class": "org.example.samplenetwork.Trade",
3   "commodity": "resource:org.example.samplenetwork.Commodity#GOT",
4   "newOwner": "resource:org.example.samplenetwork.Trader#trader2",
5   "transactionId": "f171f3c9-65a8-4b14-8ed5-078631b2c28b",
6   "timestamp": "2018-07-13T10:35:59.840Z"
7 }
```

On the right side of the interface, a list of transaction records is visible, each showing a partial ID (e.g., 'dmin') and a 'view record' link.