**Assignment 8: Setting up Hyperledger Fabric Network**

**Scenario:**

**Hopps Marker Car Auctions LLPhave now realizedthe capabilities of Hyperledger Fabric and wish to explore the platform further for more customization and features. They now wish to create a Business Network locally. Youhavebeen hired as a Blockchain developer to help them achieve this.**

**Here is what you need to do: Create a Local Business Network Containing the following**

1. **3 Participant Types - Member, Auctioneer, Owner**
2. **2 Asset Types - Vehicle, VehicleListing**
3. **2 Transaction Types - Offer, CloseBidding**

**After defining the business network, Create and Submit the "caracutionnetwork.bna"file for evaluation.**

**Solution:**

To achieve this assignment, I am installing Hyperledger composer first and then installing Hyperledger Fabric. Later, I am developing the Auction project using Visual studio code IDE and using the sample project of business network. After all files modified and added, I compiled this business network to **.bna** file and use this into the browser for testing.

First of all, installing prerequisites for Hyperledger Composer

```
curl -O https://hyperledger.github.io/composer/prereqs-ubuntu.sh
```

```
chmod u+x prereqs-ubuntu.sh
```

```
sudo ./prereqs-ubuntu.sh
```

```
edureka@edureka: ~
edureka@edureka:~$ curl -O https://hyperledger.github.io/composer/prereqs-ubuntu.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3519  100  3519    0     0   4227      0 --:--:-- --:--:-- --:--:--  4229
edureka@edureka:~$ chmod u+x prereqs-ubuntu.sh
edureka@edureka:~$ ./prereqs-ubuntu.sh
Installing Hyperledger Composer prereqs for Ubuntu xenial
# Updating package lists
[sudo] password for edureka:
gpg: keyring `/tmp/tmp6edyoa7v/secring.gpg' created
gpg: keyring `/tmp/tmp6edyoa7v/pubring.gpg' created
gpg: requesting key E1DF1F24 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmp6edyoa7v/trustdb.gpg: trustdb created
gpg: key E1DF1F24: public key "Launchpad PPA for Ubuntu Git Maintainers" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
Ign:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:3 http://ppa.launchpad.net/git-core/ppa/ubuntu xenial InRelease
Hit:4 http://dl.google.com/linux/chrome/deb stable Release
Hit:5 http://in.archive.ubuntu.com/ubuntu xenial InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:7 http://ppa.launchpad.net/ubuntu-desktop/ubuntu-make/ubuntu xenial InRelease
Hit:8 http://in.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:9 https://download.docker.com/linux/ubuntu xenial InRelease
Hit:10 https://deb.nodesource.com/node_8.x xenial InRelease
Reading package lists... Done
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/
:1
```

**Step 1: Install the CLI tools**

1. Essential CLI tools:

   ```
   npm install -g composer-cli
   ```

2. Utility for running a REST Server on your machine to expose your business networks as RESTful APIs:

   ```
   npm install -g composer-rest-server
   ```

3. Useful utility for generating application assets:

   ```
   npm install -g generator-hyperledger-composer
   ```

4. Yeoman is a tool for generating applications, which utilises generator-hyperledger-composer:

   ```
   npm install -g yo
   ```

**Step 2: Install Playground**

Installing playground, we can run business network code locally on our development machine which give us a UI for viewing and demonstrating our business networks.

Browser app for simple editing and testing Business Networks:

```
npm install -g composer-playground
```

**Step 3: Set up your IDE**

We can use the browser app to work on our Business Network code, but most users prefer to work in an IDE. Most of users favorite IDE is Visual Studio code, because a **Composer extension is available**.

Install VSCode from this URL: https://code.visualstudio.com/download

Open VSCode, go to Extensions, then search for and install the Hyperledger Composer extension from the Marketplace.

**Step 4: Install Hyperledger Fabric**

This step gives us a local Hyperledger Fabric runtime to deploy our business networks to.

1. In a directory of our choice (we will assume ~/fabric-tools), get the .zip file that contains the tools to install Hyperledger Fabric:

   ```
   mkdir ~/fabric-tools

   cd ~/fabric-tools

   curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.zip

   unzip fabric-dev-servers.zip
   ```

   A tar.gz is also available if you prefer: just replace the .zip file with fabric-dev-servers.tar.gz1 and the unzip command with a tar xvzf command in the above snippet.

2. Use the scripts you just downloaded and extracted to download a local Hyperledger Fabric runtime:

   ```
   cd ~/fabric-tools

   ./downloadFabric.sh
   ```

```
edureka@edureka: ~/fabric-tools
edureka@edureka:~/fabric-tools$ ./downloadFabric.sh
Development only script for Hyperledger Fabric control
Running 'downloadFabric.sh'
FABRIC_VERSION is unset, assuming hlfv1
FABRIC_START_TIMEOUT is unset, assuming 15 (seconds)

# Set ARCH
ARCH=`uname -m`
uname -m

# Grab the current directory.
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
 cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd
 dirname "${BASH_SOURCE[0]}"

# Pull and tag the latest Hyperledger Fabric base image.
docker pull hyperledger/fabric-peer:$ARCH-1.0.4
x86_64-1.0.4: Pulling from hyperledger/fabric-peer
d5c6f90da05d: Pull complete
1300883d87d5: Pull complete
c220aa3cfc1b: Pull complete
2e9398f099dc: Pull complete
dc27a084064f: Pull complete
87675a6d4030: Pull complete
93e601aafda8: Pull complete
278385815258: Pull complete
78f3c6b30e0c: Pull complete
621be214faa6: Pull complete
Digest: sha256:42bf413394938a48af26a4cc6ac6393d8c61acb51736a1fe6f45447850314778
Status: Downloaded newer image for hyperledger/fabric-peer:x86_64-1.0.4
docker pull hyperledger/fabric-ca:$ARCH-1.0.4
x86_64-1.0.4: Pulling from hyperledger/fabric-ca
aafe6b5e13de: Already exists
0a2b43a72660: Already exists
```

We have now installed everything required for the typical Developer Environment.

Now, lets create a **Business Network for Car Auction**

**Step 1: Create a Business Network Definition**

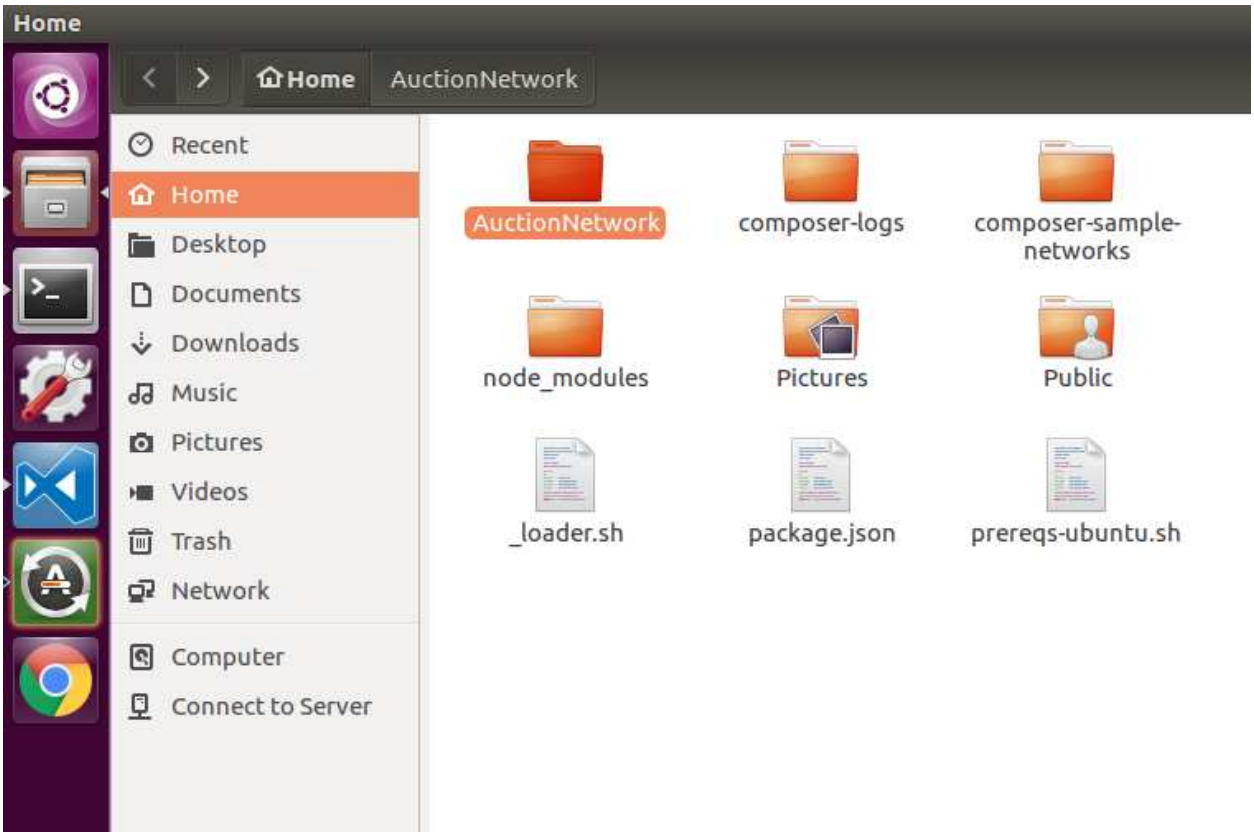
Lets begin by cloning a sample business network.

Open up a command prompt and execute the following command:

```
git clone https://github.com/hyperledger/composer-sample-networks.git
```

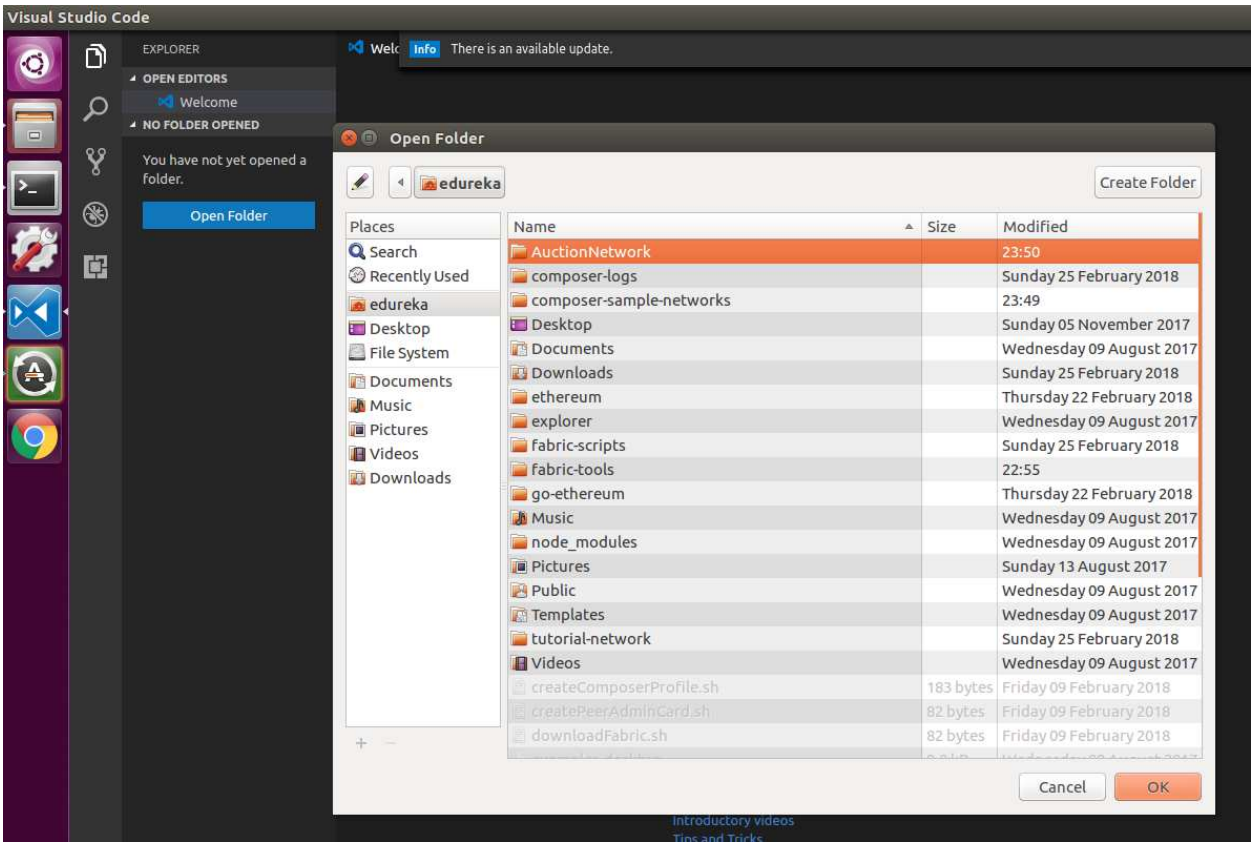
Make a copy of this directory in your project, called 'my-network' by using the following command:

```
cp -r ./composer-sample-networks/packages/basic-sample-network/
./AuctionNetwork
```




**Step 2: Opening Project File**

Open Visual Studio(VS) Code and click on Open Folder. Locate AuctionNetwork folder and click OK

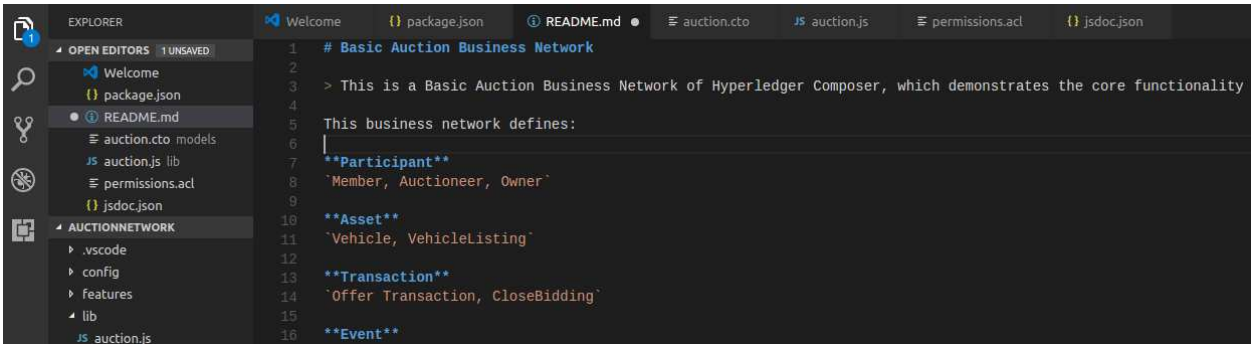



### Step 3: Update your package.json file

In the left panel, select package.json file under AuctionNetwork folder and changed the code with following.

**Step 4: Update README.md File**

Open README.md file and Update the first line as seen bellow:




**Step 5: Define Domain Model** (auction.cto file)

Delete the file models/sample.cto file and create auction.cto under models folder

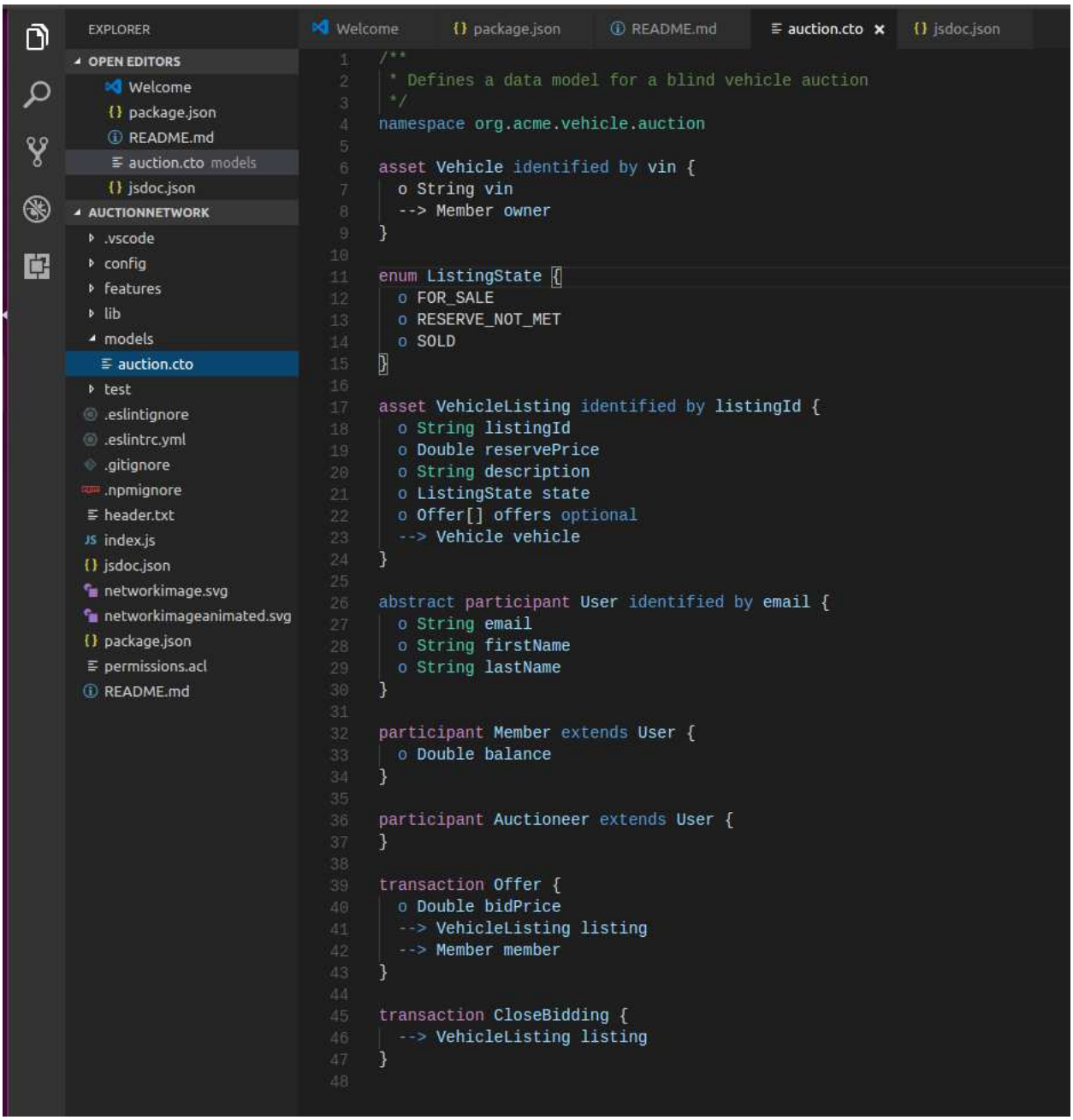
This is the domain model for the business network definition. It defines the structure (schema) for the assets, transaction and participants in the business network

Replace the existing code with the following code:

```
/**
 * Defines a data model for a blind vehicle auction
 */
namespace org.acme.vehicle.auction

asset Vehicle identified by vin {
  o String vin
  --> Member owner
}

enum ListingState {
  o FOR_SALE
  o RESERVE_NOT_MET
```

```
  o SOLD
}

asset VehicleListing identified by listingId {
  o String listingId
  o Double reservePrice
  o String description
  o ListingState state
  o Offer[] offers optional
  --> Vehicle vehicle
}

abstract participant User identified by email {
  o String email
  o String firstName
  o String lastName
}

participant Member extends User {
  o Double balance
}

participant Auctioneer extends User {
}

transaction Offer {
  o Double bidPrice
  --> VehicleListing listing
  --> Member member
}
```

```
transaction CloseBidding {
  --> VehicleListing listing
}
```



**Step 6: Write Transaction Processor Functions (Script File)**

1. Delete the file lib/sample.js in the left-hand pane

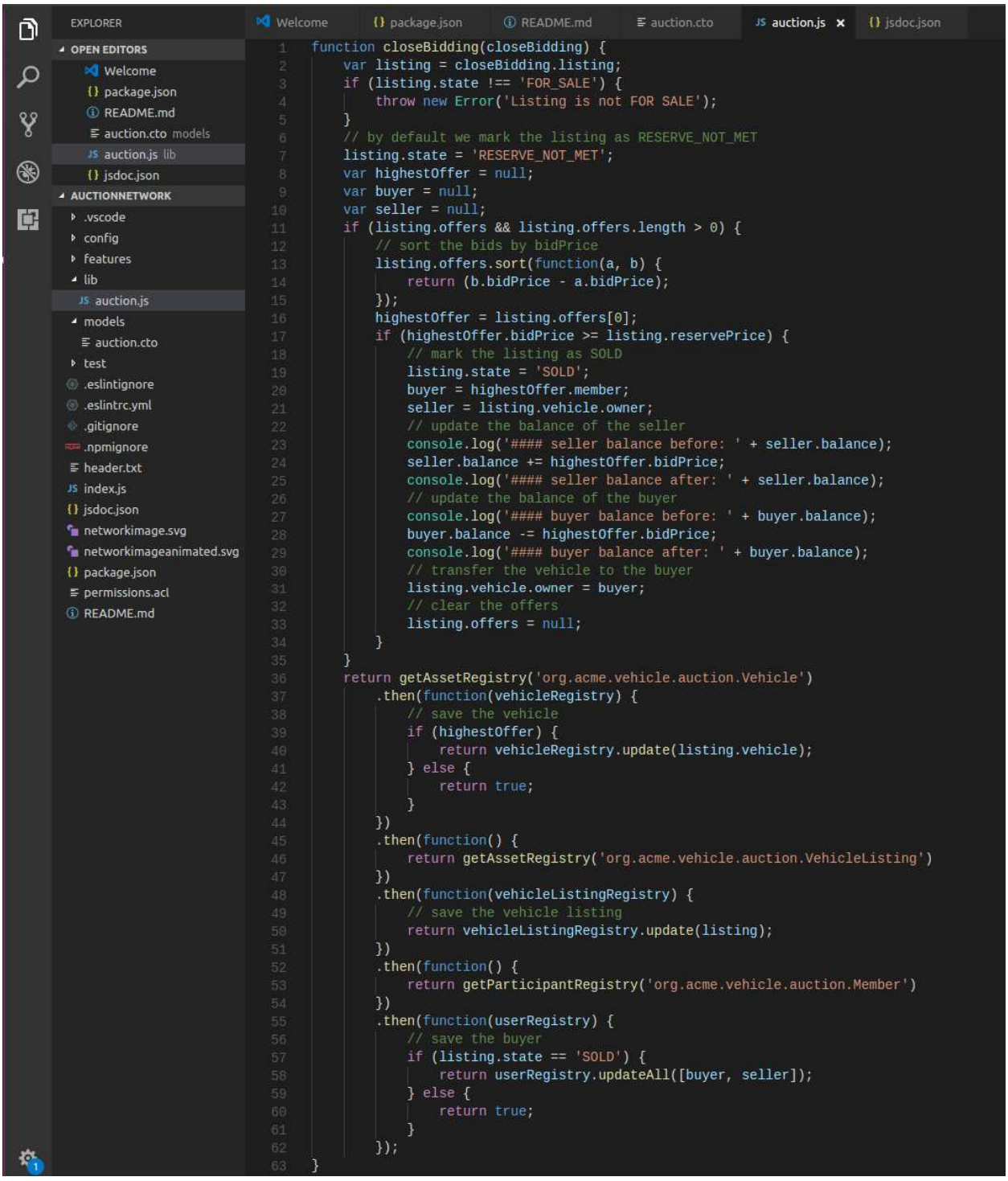
2. Create auction.js file with the following code:

```
function closeBidding(closeBidding) {
    var listing = closeBidding.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    // by default we mark the listing as RESERVE_NOT_MET
    listing.state = 'RESERVE_NOT_MET';
    var highestOffer = null;
    var buyer = null;
    var seller = null;
    if (listing.offers && listing.offers.length > 0) {
        // sort the bids by bidPrice
        listing.offers.sort(function(a, b) {
            return (b.bidPrice - a.bidPrice);
        });
        highestOffer = listing.offers[0];
        if (highestOffer.bidPrice >= listing.reservePrice) {
            // mark the listing as SOLD
            listing.state = 'SOLD';
            buyer = highestOffer.member;
            seller = listing.vehicle.owner;
            // update the balance of the seller
            console.log('#### seller balance before: ' +
seller.balance);
            seller.balance += highestOffer.bidPrice;
```

```
            console.log('#### seller balance after: ' +
seller.balance);
            // update the balance of the buyer
            console.log('#### buyer balance before: ' +
buyer.balance);
            buyer.balance -= highestOffer.bidPrice;
            console.log('#### buyer balance after: ' + buyer.balance);
            // transfer the vehicle to the buyer
            listing.vehicle.owner = buyer;
            // clear the offers
            listing.offers = null;
        }
    }
    return getAssetRegistry('org.acme.vehicle.auction.Vehicle')
        .then(function(vehicleRegistry) {
            // save the vehicle
            if (highestOffer) {
                return vehicleRegistry.update(listing.vehicle);
            } else {
                return true;
            }
        })
        .then(function() {
            return
getAssetRegistry('org.acme.vehicle.auction.VehicleListing')
        })
        .then(function(vehicleListingRegistry) {
            // save the vehicle listing
            return vehicleListingRegistry.update(listing);
        })
        .then(function() {
```

```
            return
getParticipantRegistry('org.acme.vehicle.auction.Member')
        })
        .then(function(userRegistry) {
            // save the buyer
            if (listing.state == 'SOLD') {
                return userRegistry.updateAll([buyer, seller]);
            } else {
                return true;
            }
        });
}

/**
 * Make an Offer for a VehicleListing
 * @param {org.acme.vehicle.auction.Offer} offer - the offer
 * @transaction
 */
function makeOffer(offer) {
    var listing = offer.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    if (listing.offers == null) {
        listing.offers = [];
    }
    listing.offers.push(offer);
    return getAssetRegistry('org.acme.vehicle.auction.VehicleListing')
        .then(function(vehicleListingRegistry) {
            // save the vehicle listing
```

```
            return vehicleListingRegistry.update(listing);
        });
}
```



**Step 7: Update your Access Control Rules**

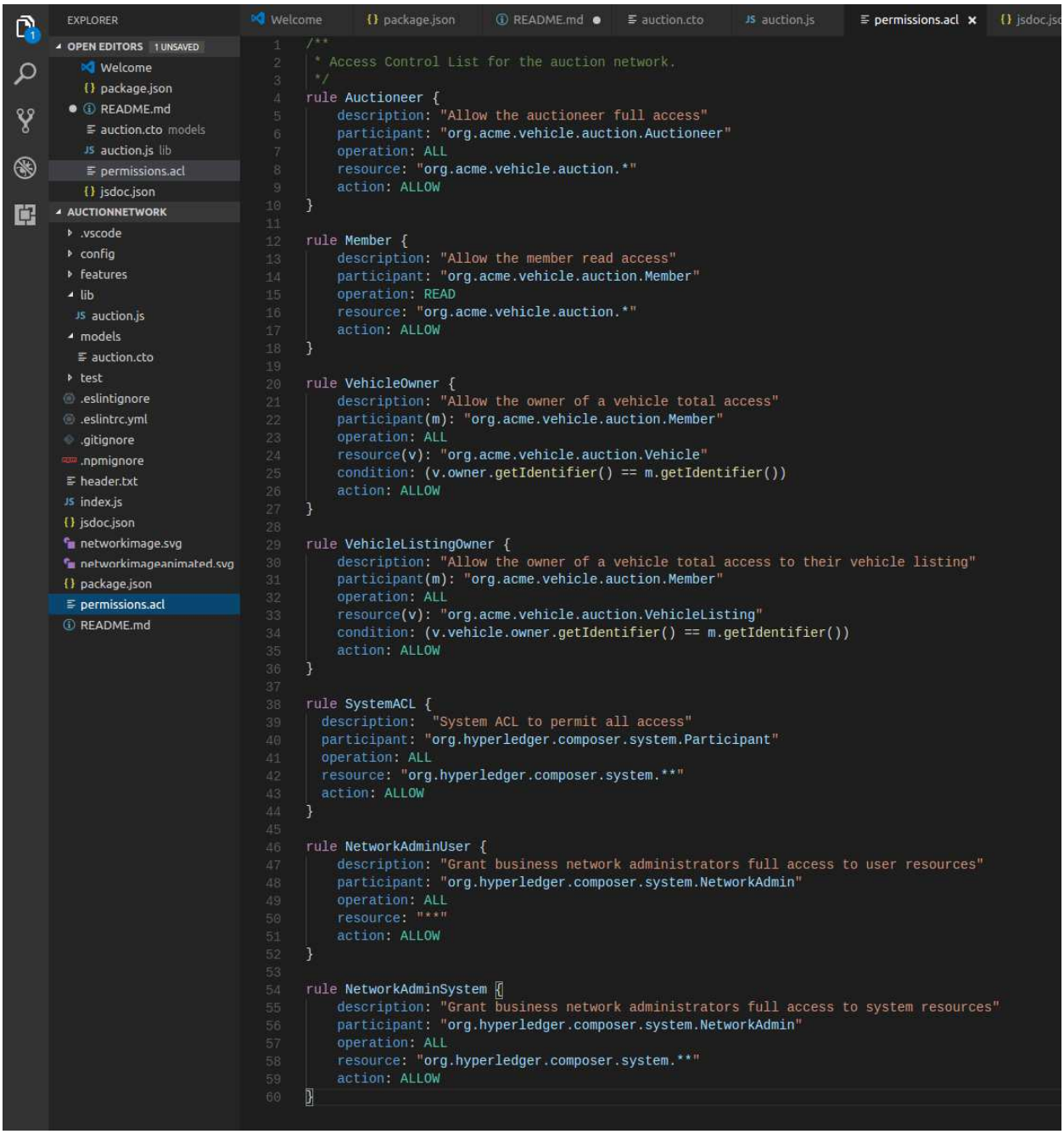
The file permissions.acl defines the access control rules for the business network definition.

Replace the entire contents of permissions.acl with the rule below:

```
/**
 * Access Control List for the auction network.
 */
rule Auctioneer {
    description: "Allow the auctioneer full access"
    participant: "org.acme.vehicle.auction.Auctioneer"
    operation: ALL
    resource: "org.acme.vehicle.auction.*"
    action: ALLOW
}

rule Member {
    description: "Allow the member read access"
    participant: "org.acme.vehicle.auction.Member"
    operation: READ
    resource: "org.acme.vehicle.auction.*"
    action: ALLOW
}

rule VehicleOwner {
    description: "Allow the owner of a vehicle total access"
    participant(m): "org.acme.vehicle.auction.Member"
    operation: ALL
    resource(v): "org.acme.vehicle.auction.Vehicle"
    condition: (v.owner.getIdentifier() == m.getIdentifier())
    action: ALLOW
}
```

```
rule VehicleListingOwner {
    description: "Allow the owner of a vehicle total access to their
vehicle listing"
    participant(m): "org.acme.vehicle.auction.Member"
    operation: ALL
    resource(v): "org.acme.vehicle.auction.VehicleListing"
    condition: (v.vehicle.owner.getIdentifier() == m.getIdentifier())
    action: ALLOW
}

rule SystemACL {
  description:  "System ACL to permit all access"
  participant: "org.hyperledger.composer.system.Participant"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}

rule NetworkAdminUser {
    description: "Grant business network administrators full access to
user resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "**"
    action: ALLOW
}

rule NetworkAdminSystem {
    description: "Grant business network administrators full access to
system resources"
```

```
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}
```



**Step 8: Generate the Auction Business Network Archive**

Open a new Terminal and execute the following command:

```
cd AuctionNetwork

npm install
```

```
  birthtime: 2018-03-20T19:07:35.567Z }
  SOLINK_MODULE(target) Release/obj.target/pkcs11.node
  COPY Release/pkcs11.node
make: Leaving directory '/home/edureka/AuctionNetwork/node_modules/pkcs11js/build'

> grpc@1.6.6 install /home/edureka/AuctionNetwork/node_modules/grpc
> node-pre-gyp install --fallback-to-build --library=static_library

[grpc] Success: "/home/edureka/AuctionNetwork/node_modules/grpc/src/node/extension_binary/node-v57-linux-x64/grpc_node.node"

> cucumber-expressions@3.0.0 postinstall /home/edureka/AuctionNetwork/node_modules/cucumber-expressions
> node scripts/postinstall.js


> protobufjs@6.6.3 postinstall /home/edureka/AuctionNetwork/node_modules/protobufjs
> node scripts/postinstall

npm WARN prepublish-on-install As of npm@5, `prepublish` scripts are deprecated.
npm WARN prepublish-on-install Use `prepare` for build steps and `prepublishOnly` for upload-only.
npm WARN prepublish-on-install See the deprecation note in `npm help scripts` for more information.

> auction@0.1.14 prepublish /home/edureka/AuctionNetwork
> mkdirp ./dist && composer archive create --sourceType dir --sourceName . -a ./dist/auction.bna

Creating Business Network Archive


Looking for package.json of Business Network Definition
        Input directory: /home/edureka/AuctionNetwork

Found:
        Description: My Auction Network
        Name: auction
        Identifier: auction@0.1.14

Written Business Network Definition Archive file to
        Output file: ./dist/auction.bna

Command succeeded

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.3 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.1.3: wanted {"os":"darwin","arch":"any"} (c

added 1627 packages in 166.995s
edureka@edureka:~/AuctionNetwork$
```

**Step 9: Creating a BNA file**

Ensure we are at the top-level project folder (AuctionNetwork).

Generate the BNA file using the following command:

```
mkdir dist
```

```
composer archive create -a dist/my-network.bna --sourceType dir --
sourceName
```

We can also use the command `npm run prepublish` achieves the same thing.

```
edureka@edureka:~/AuctionNetwork$ npm run prepublish

> auction@0.1.14 prepublish /home/edureka/AuctionNetwork
> mkdirp ./dist && composer archive create --sourceType dir --sourceName . -a ./dist/auction.bna

Creating Business Network Archive


Looking for package.json of Business Network Definition
        Input directory: /home/edureka/AuctionNetwork

Found:
        Description: My Auction Network
        Name: auction
        Identifier: auction@0.1.14

Written Business Network Definition Archive file to
        Output file: ./dist/auction.bna

Command succeeded

edureka@edureka:~/AuctionNetwork$
```

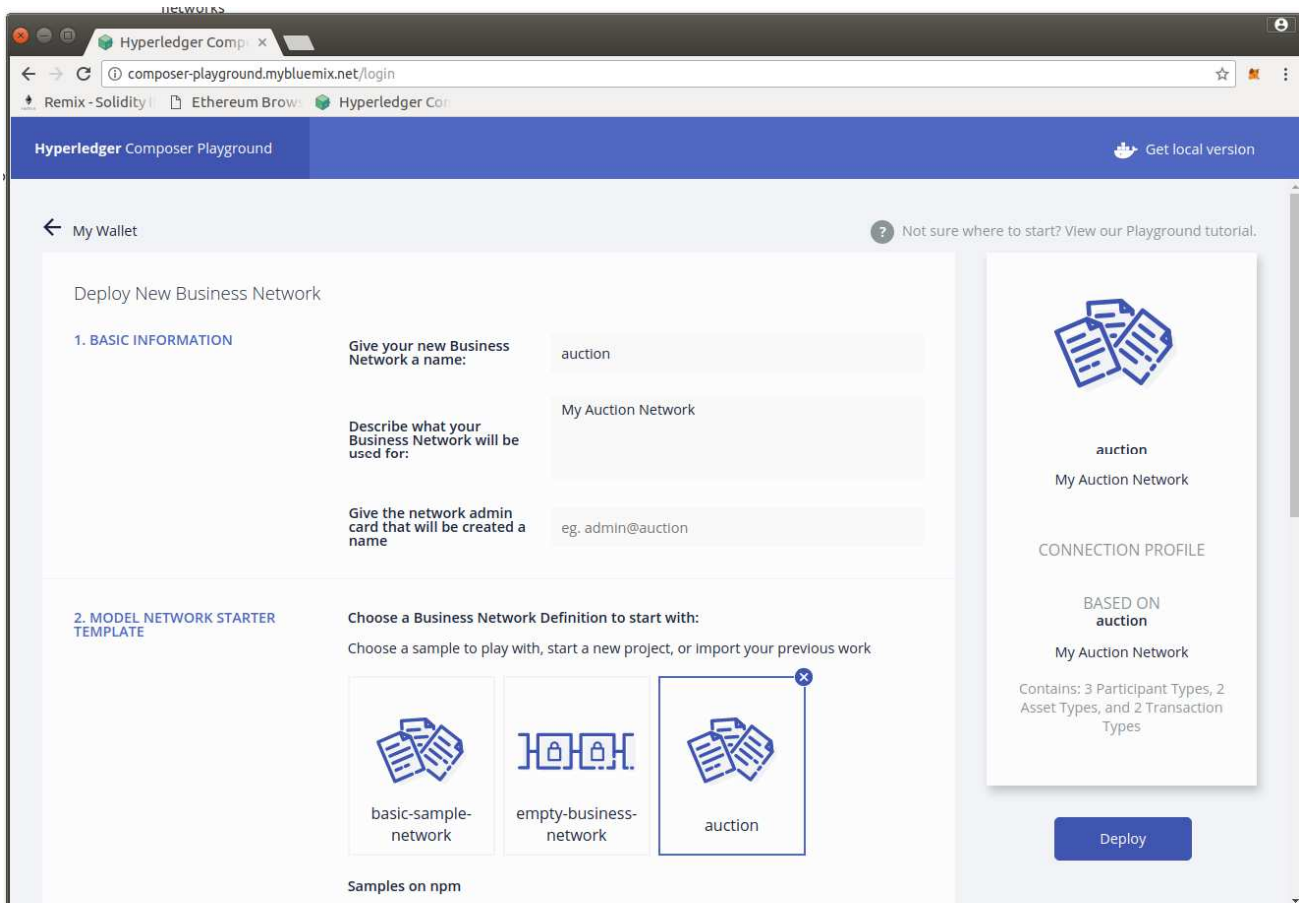
**Step 10: Deploying Business Network to Composer**

In Chrome browser, navigate to the online [Bluemix Composer Playground](Bluemix Composer Playground).

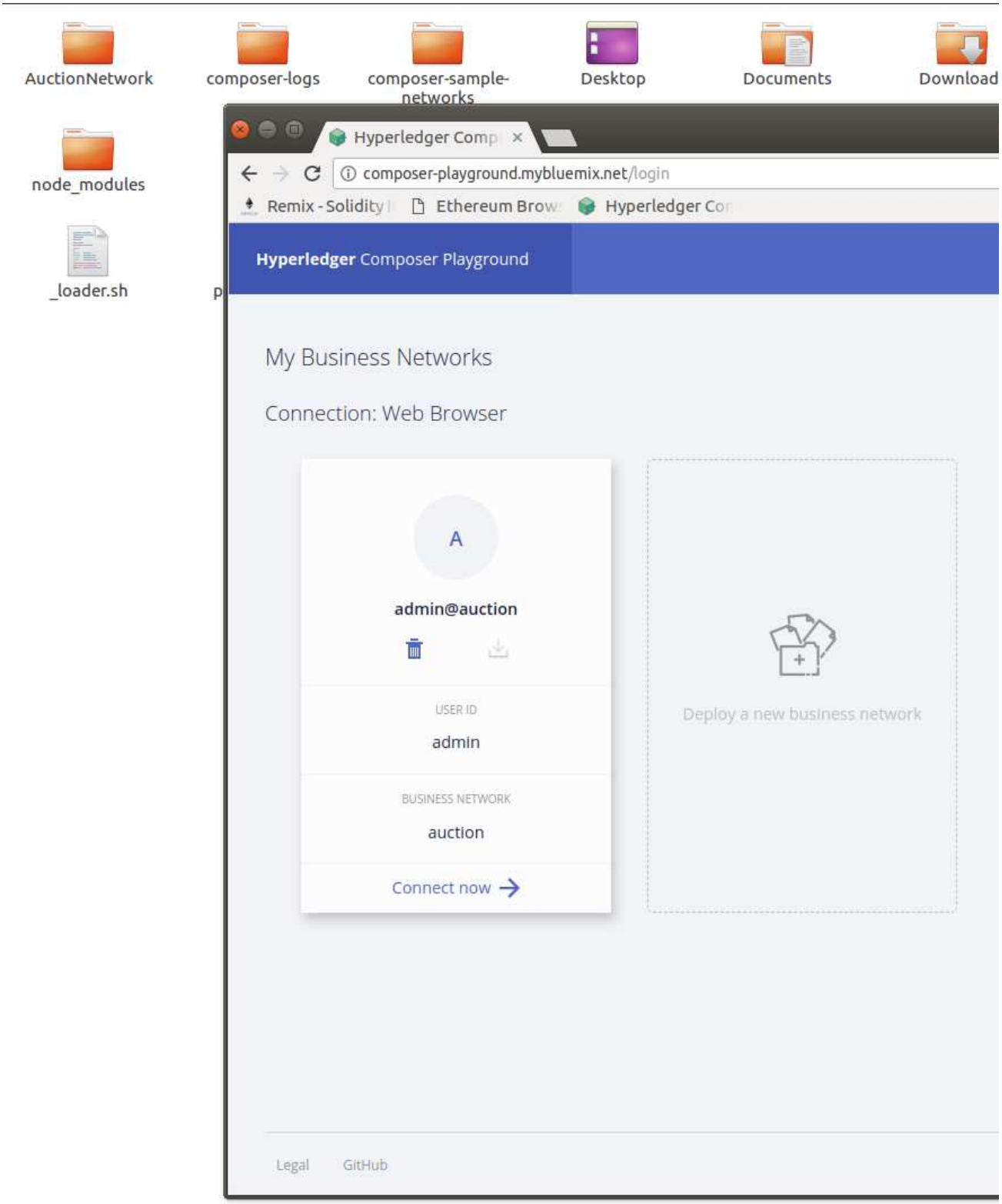
After browsing above link, Click **Let's Blockchain** button

Click on **Deploy a new business network**

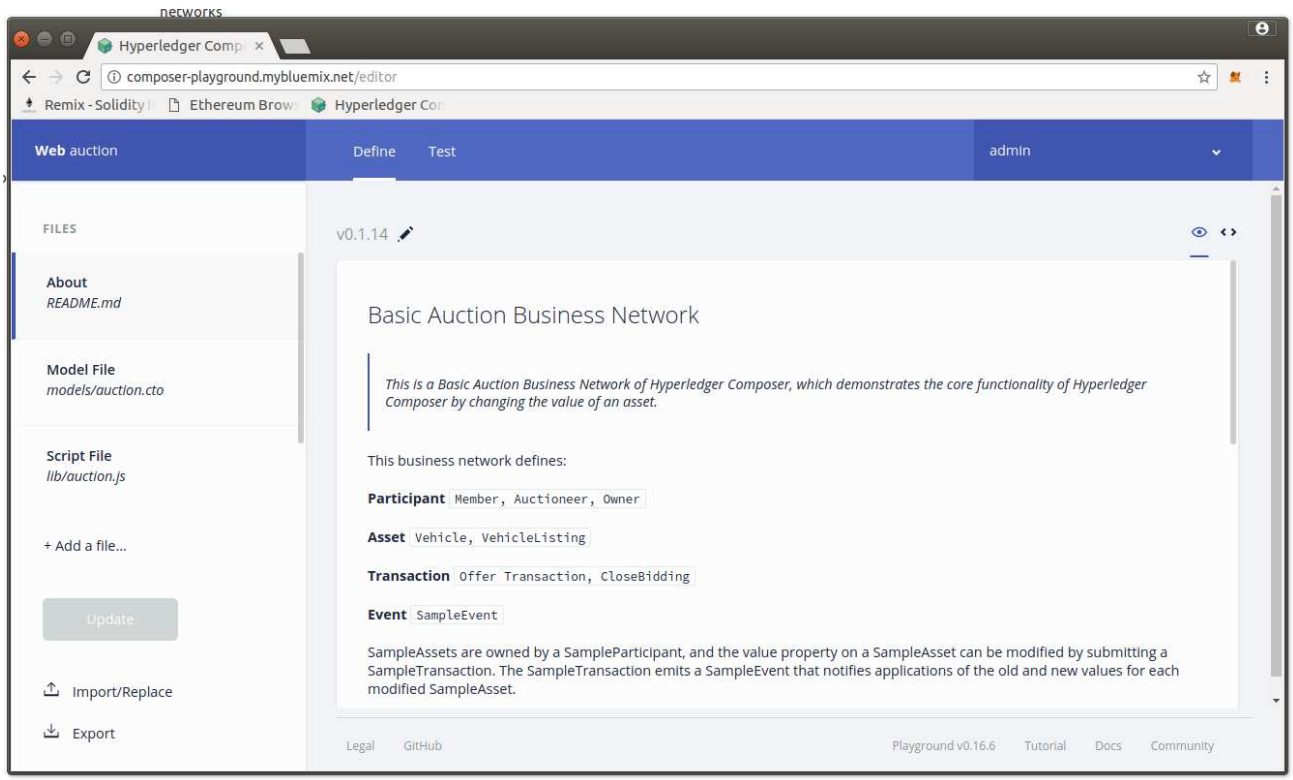
Click on **Drop here to upload or Browse** to deploy `auction.bna` file.




Now, Click on **Deploy**




Click on **Connect now**.




We are connected now on our Auction business Network.