Multiple user identity and authentication

Reference links:

1. <https://hyperledger.github.io/composer/tutorials/developer-guide.html>
2. <https://hyperledger.github.io/composer/integrating/enabling-multiuser.html>
3. <https://hyperledger.github.io/composer/integrating/enabling-rest-authentication.html>

**Step 0:**

**Install required components:**

npm install -g composer-cli

npm install -g generator-hyperledger-composer

npm install -g composer-rest-server

**Go to fabric-tools folder and in terminal type**

./stopFabric.sh

./teardownFabric.sh

./startFabric.sh

./createFabric.sh

**Get a sample example from git**

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

cp -r ./composer-sample-networks/packages/basic-sample-network/ ./my-network

**Update your package.json file in the project folder my-network**

"name": "my-network",

"version": "0.1.6",

"description": "My Commodity Trading network",

"networkImage": "https://hyperledger.github.io/composer-sample-networks/packages/basic-sample-network/networkimage.svg",

"networkImageanimated": "https://hyperledger.github.io/composer-sample-networks/packages/basic-sample-network/networkimageanimated.svg",

"scripts": {

"prepublish": "mkdirp ./dist ; composer archive create --sourceType dir --sourceName . -a ./dist/my-network.bna",

"pretest": "npm run lint",

"lint": "eslint .",

"postlint": "npm run licchk",

"licchk": "license-check",

"postlicchk": "npm run doc",

"doc": "jsdoc --pedantic --recurse -c jsdoc.json",

"test-inner": "mocha -t 0 --recursive && cucumber-js",

"test-cover": "nyc npm run test-inner",

"test": "npm run test-inner"

},

## **Define your Domain (**models/sample.cto**)**

/\*\*

\* My commodity trading network

\*/

namespace org.acme.mynetwork

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

}

**Write Transaction Processor Functions(**lib/sample.js**)**

/\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

/\*\*

\* Track the trade of a commodity from one trader to another

\* @param {org.acme.mynetwork.Trade} trade - the trade to be processed

\* @transaction

\*/

function tradeCommodity(trade) {

trade.commodity.owner = trade.newOwner;

return getAssetRegistry('org.acme.mynetwork.Commodity')

.then(function (assetRegistry) {

return assetRegistry.update(trade.commodity);

});

}

**Update your Access Control Rules(**permissions.acl**)**

/\*\*

\* Access control rules for mynetwork

\*/

rule Default {

description: "Allow all participants access to all resources"

participant: "ANY"

operation: ALL

resource: "org.acme.mynetwork.\*"

action: ALLOW

}

rule SystemACL {

description: "System ACL to permit all access"

participant: "ANY"

operation: ALL

resource: "org.hyperledger.composer.system.\*\*"

action: ALLOW

}

## **Generate the Business Network Archive**

Switch back to the terminal and type:

cd my-network

npm install

## **Deploy to the running Hyperledger Fabric**

Switch to the terminal, change directory to the dist folder containing the my-network.bna file and type:

cd dist

composer network deploy -a my-network.bna -p hlfv1 -i PeerAdmin -s randomString -A admin -S

**Adding participants**

**Step 1:**

composer participant add -p hlfv1 -n 'my-network' -i admin -s adminpw -d '{"$class":"org.acme.mynetwork.Trader","tradeId":"bob1","firstName":" bob","lastName":"Dylan","company":"Capgemini"}'

composer participant add -p hlfv1 -n 'my-network' -i admin -s adminpw -d '{"$class":"org.acme.mynetwork.Trader","tradeId":"alice1","firstName":" alice","lastName":"vikander","company":"IGate"}'

**Step 2:**

composer identity issue -p hlfv1 -n 'my-network' -i admin -s adminpw -u bob1 -a "resource:org.acme.mynetwork.Trader#bob1"

userID = bob1

userSecret = HLEDFPOgPXAk

composer identity issue -p hlfv1 -n 'my-network' -i admin -s adminpw -u alice1 -a "resource:org.acme.mynetwork.Trader#alice1"

userID = alice1

userSecret = DRisDifWFJWx﻿

**Step 3: (Refer link 2 and 3 )**

Create a github profile and link it to admin

Github

Client ID

﻿ ﻿﻿d8cc59a17a92649025ee

Client Secret

﻿ ﻿﻿7601b61d4bc97733470db915e1abb01a9da45693

﻿

export COMPOSER\_PROVIDERS='{

"github": {

"provider": "github",

"module": "passport-github",

"clientID": "d8cc59a17a92649025ee",

"clientSecret": "﻿7601b61d4bc97733470db915e1abb01a9da45693",

"authPath": "/auth/github",

"callbackURL": "/auth/github/callback",

"successRedirect": "/",

"failureRedirect": "/"

}

}'

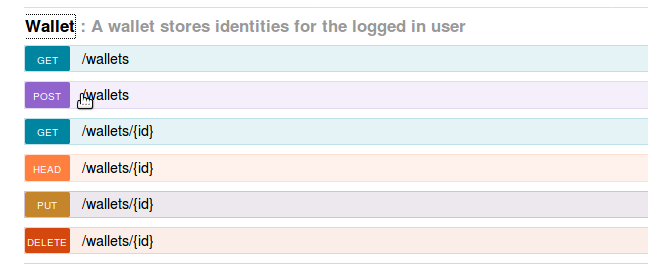
**Step 4:**

Now run the rest server in multiuser mode

﻿

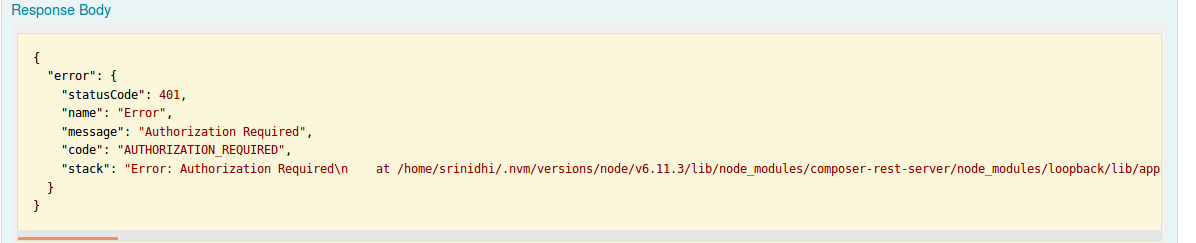
composer-rest-server -p hlfv1 -n test-network -i admin -s adminpw -a true –m true

It will loopback and add a wallet endpoint to the rest collections



***Now getting the /wallets operation will be rejected and asks for authorization (which indicates successful authentication has been done )***

﻿

**

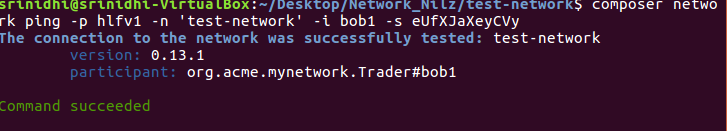
**N.B. [**

Check if the identity is added and issued to blockchain or not

*composer network ping -p hlfv1 -n 'test-network' -i admin -s adminpw*

*composer network ping -p hlfv1 -n 'test-network' -i bob1 -s* iVxxvgslkiIw

*composer network ping -p hlfv1 -n 'test-network' -i alice1 -s* AfnYpEcqYBHM

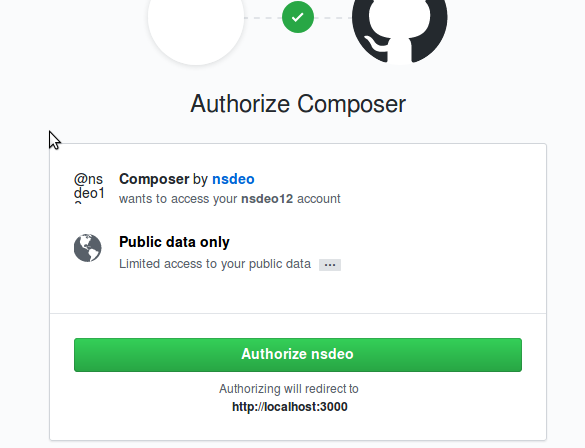
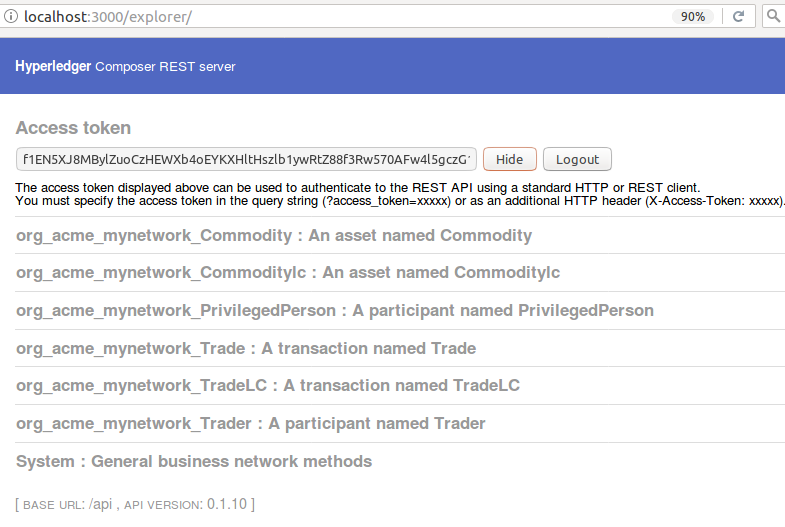


If not add the user using the ***step 2*** above.**]**

**Step 5:**

**Now go to** [**http://localhost:3000/auth/github**](http://localhost:3000/auth/github)

**It will show the github screen asking for authorization.so authorize it.**

****

**Step 6:**

**The access token can be seen as below**



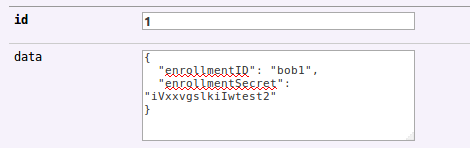
Now go to the wallet and check for the stored credential using ﻿

http://localhost:3000/api/wallets

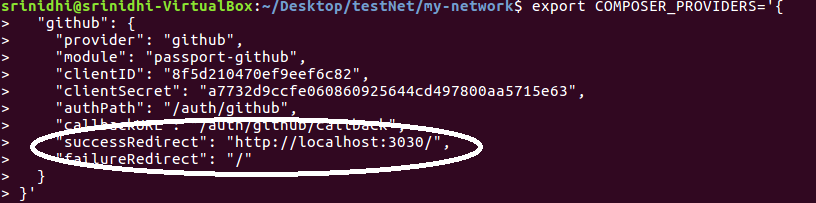


A new identity can be added using the following URL too.

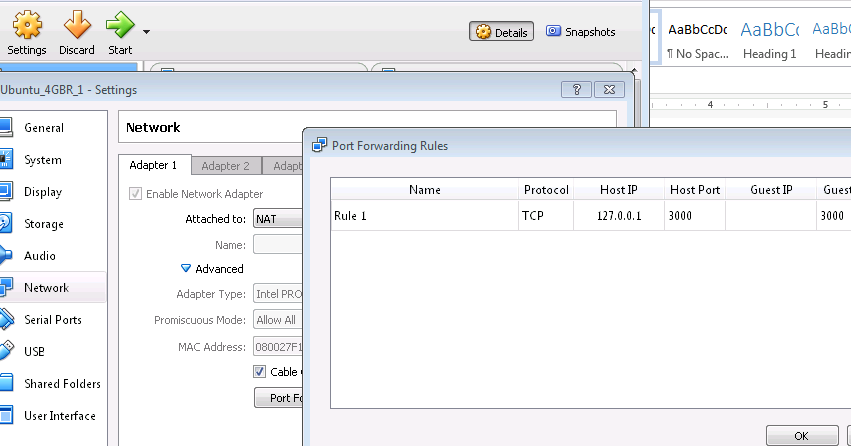
****



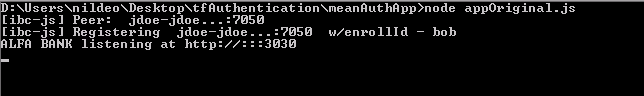
**Accessing from another app running in 3030**

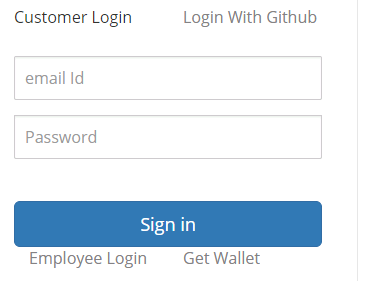
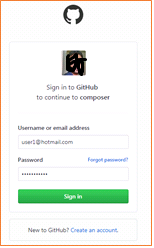
Make sure the **COMPOSER\_PROVIDERS** should look like the following in Ubuntu (with **successRedirect** field)

Change the setting as below in the VM. Assuming composer running in 3000 port in Ubuntu and other app running in 3030 in host (windows)



Start the application in console.



Run the app in browser

It will redirect to **Login with Github** page

After signing in, it should redirect back to the application running on 3030 port.

# Some theories

## Authorization Process

* In order to make an [HTTP](https://developer.mozilla.org/en/HTTP) request to the server with JavaScript, one need an instance of an object with the necessary functionality.
* This is where XMLHttpRequest comes in. The [XMLHttpRequest](https://xhr.spec.whatwg.org/#xmlhttprequest) object is an API for [fetching](https://fetch.spec.whatwg.org/#concept-fetch) resources.
* Its predecessor is ActiveX from Microsoft. For using it in IE6 it should be explicitly mentioned in the code.

// Old compatibility code, no longer needed.

if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...

httpRequest = new XMLHttpRequest();

} else if (window.ActiveXObject) { // IE 6 and older

httpRequest = new ActiveXObject("Microsoft.XMLHTTP");

}

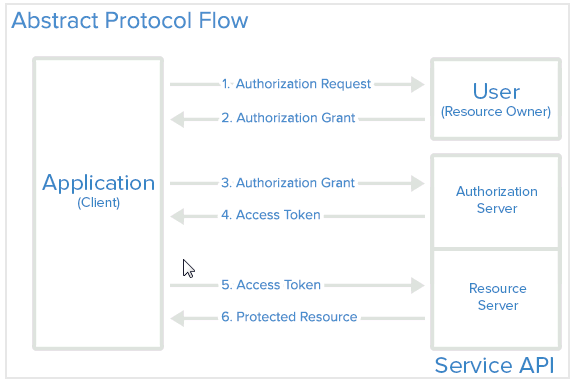
### As the Third-Party Application: "Client" trying to access content from another application that application should authorize it .It can be done using its own authorizing mechanism or OAuth.

### The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service. It’s a way to publish and interact with protected data.

## OAuth Roles

OAuth defines four roles:

* Resource Owner
* Client
* Resource Server
* Authorization Server

Client: It is the application that wants to access the resource.

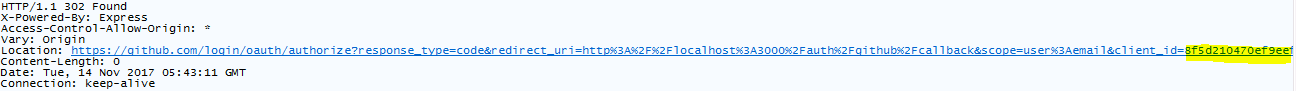
Resource Owner: Who authorizes an application to access their account?

### Resource / Authorization Server: *API:* The resource server hosts the protected information of the user and authorization server verifies the identity and issues *access token* to the application.

## Using passport strategies for authentication

### Passport is the authentication middleware for nodejs. Passport uses the concept of strategies to authenticate requests.

### Strategies can range from verifying username and password credentials, delegated authentication using [OAuth](http://oauth.net/) (for example, via [Facebook](http://www.facebook.com/) or [Twitter](http://twitter.com/)), or federated authentication using [OpenID](http://openid.net/).



# References

https://www.digitalocean.com/community/users/manicas. (n.d.). *An Introduction to OAuth 2*. Retrieved from Digital Ocean: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2