

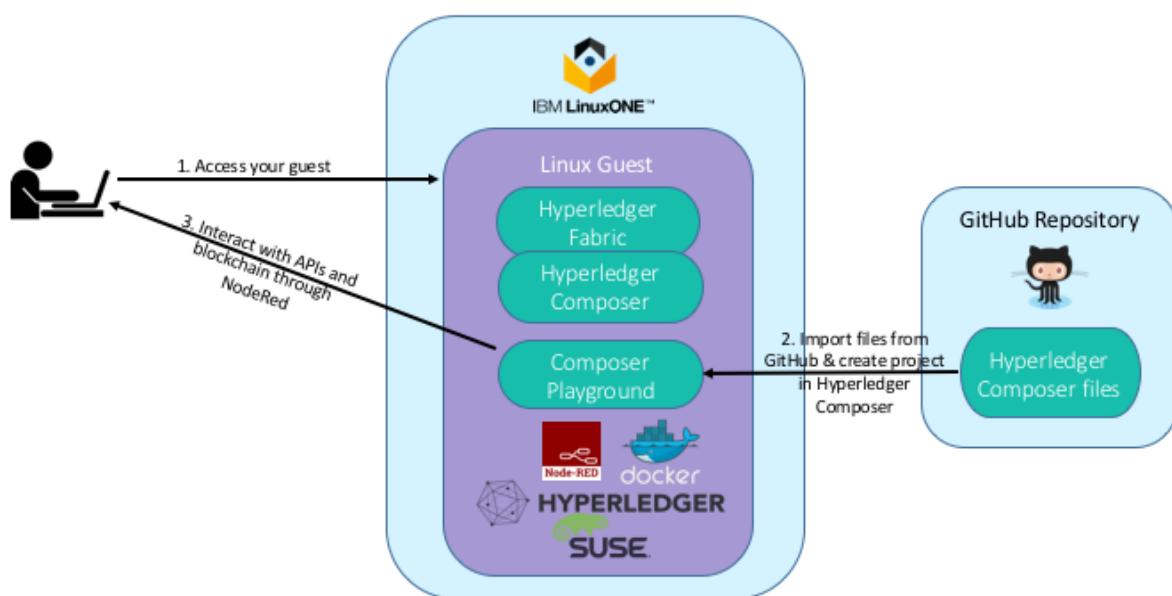
Hyperledger Fabric and Hyperledger Composer on LinuxONE

Note: To run this lab after TechU in Orlando, please use the link below.

<https://github.com/IBM/hyperledger-fabric-on-linux-one>

That guide will allow you to run the lab on a SLES guest on LinuxONE community cloud. The guide includes links to everything and information on how to get a LinuxONE Community Cloud guest.

Architecture



This lab will guide you through the following process.

1. Access your LinuxONE guest.
2. Setup and verify of your blockchain environment.
3. Create a blockchain project in Hyperledger Composer.
4. Interact with blockchain and third party APIs through Composer Rest Server and NodeRed.

Application Overview

This blockchain lab is intended to give you a basic understanding of how a developer would interact with Hyperledger Fabric using Hyperledger Composer. In this workshop you will use a browser based UI to modify chaincode, test your code and deploy your changes. You will also learn how tooling can take the code and generate API to allow for application integration through a RESTful interface.

This lab will be broken into three parts:

Part 1 — Setup your LinuxONE guest

1. [Access your guest.](#)
2. [Run a setup script.](#)
3. [Verify the installation of Hyperledger Fabric and Hyperledger Composer](#)

Part 2 — Creating a blockchain application and generating API

5. [Importing the components of your blockchain application](#)
6. [Creating your blockchain application](#)
7. [Test application code](#)
8. [Deploy application to Hyperledger Fabric](#)
9. [Generating API from your blockchain application](#)

Part 3 — Utilizing blockchain API through NodeRED

10. [Importing your flow into NodeRED](#)
11. [Interacting with blockchain through a dashboard](#)

Workshop Instructions

Scenario Overview

For this lab, we will simulate a thermostat and a temperature gauge to provide us temperature data. In a real world scenario, this could be a temperature sensor in your house or in an office building. The sensor could be connected to a real thermostat like Nest or other smart home devices via API. To keep family members, housemates, friends or children from excessively running air conditioning or heat, they must first find out if they have permission to adjust the thermostat by running a transaction defined in a smart contract running on Hyperledger Fabric. The contract will check the value recorded in the ledger for the temperature gauge to determine if their thermostat adjustment is environmentally friendly. Secondly, it will add integration to Weather.com to check current temperatures and adjust the thermostat to ideal settings based on the terms of the smart contract.

Part 1 - Setup your LinuxONE guest

In this section of the lab, you will use PuTTY to connect to your LinuxONE guest, setup your blockchain environment and verify everything is running. Your guest is a basic Ubuntu 16.04.3 server running on an IBM z13 server residing in the Washington Systems Center in Herndon, Virginia. You can verify your Ubuntu version the command, `uname -a`, once you're logged into your guest.

Access your LinuxONE guest

1. At your workstation, you should be given the IP address of a Linux guest you will be working with. Keep this information available throughout the lab.
2. This lab document assumes your lab instructors have already started your VPN for you. If you cannot get the next step to work, please ask a lab instructor for assistance.
3. **Open** PuTTY and **select** the preconfigured session showing the IP address of your guest to connect to your LinuxONE guest. Once you're connected you should see something like the image below.

```
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Wed Mar  7 18:57:11 2018 from 192.168.215.44
```

Setup your blockchain environment

4. To save time in this lab, the pre-requisites for this lab are already installed. Use the following commands to verify installation.
 - Docker: `docker -v`
 - Docker Compose: `docker-compose -v`
 - Hyperledger Composer: `composer -v`
 - Hyperledger Fabric: `docker images`

```
bcuser@ubuntu16043:~$ docker -v
Docker version 17.06.2-ce, build cec0b72
bcuser@ubuntu16043:~$ docker-compose -v
docker-compose version 1.19.0, build 9e633ef
bcuser@ubuntu16043:~$ composer -v
v0.17.6
bcuser@ubuntu16043:~$ docker images
REPOSITORY          TAG           IMAGE ID      CREATED       SIZE
hyperledger/fabric-ca   s390x-1.1.0-alpha  f6930b769b2e  5 weeks ago   297MB
hyperledger/fabric-orderer  s390x-1.1.0-alpha  3ea7bb0520ae  5 weeks ago   227MB
hyperledger/fabric-peer    s390x-1.1.0-alpha  6180e7247c51  5 weeks ago   233MB
hyperledger/fabric-ccenv   s390x-1.1.0-alpha  6303ad1d7c33  5 weeks ago   1.32GB
hyperledger/fabric-couchdb  s390x-0.4.5     b4c1f99eddbc  6 weeks ago   1.7GB
bcuser@ubuntu16043:~$
```

5. To be ready to build your first blockchain application, you'll need to get everything running. This has been scripted for you. In your terminal, enter `ls` to see what is in your home directory. You should find a script called `SHARESacramentoBlockchainScript.sh`.

```
bcuser@ubuntu16043:~$ ls
CHANGELOG.md  README.md          bin  fabric-tools  lib      share
LICENSE       SHARESacramentoBlockchainScript.sh  etc  include      playground
```

6. Take a look at the script to see the commands that will be run to start up your blockchain environment by issuing `cat SHARESacramentoBlockchainScript.sh`.

```
echo -e "*** Start the Hyperledger Fabric network ***"
echo -e ""
cd ~/fabric-tools
./startFabric.sh                                         Start the Docker containers of Hyperledger Fabric
./createPeerAdminCard.sh                                  Create the Admin IDCard needed for Hyperledger Composer
nohup composer-playground >/home/bcuser/playground/playground.stdout 2>/home/bcuser/playground/playground.stderr & disown
#sudo bash -c "iptables-save > /etc/linuxzone/iptables.save"                                Start up the Hyperledger Composer Playground

#Start NodeRed
echo -e "*** Starting NodeRed. ***"
echo -e ""
nohup node-red >/home/bcuser/playground/nodered.stdout 2>/home/bcuser/playground/nodered.stderr & disown
```

7. Enter `./SHARESacramentoBlockchainScript.sh` to run the script.

```
FABRIC_VERSION is set to 'hlfv1'
FABRIC_START_TIMEOUT is unset, assuming 15 (seconds)
```

```
Using composer-cli at v0.17.6
```

```
Successfully created business network card file to
  Output file: /tmp/PeerAdmin@hlfv1.card
```

```
Command succeeded
```

```
Successfully imported business network card
  Card file: /tmp/PeerAdmin@hlfv1.card
  Card name: PeerAdmin@hlfv1
```

```
Command succeeded
```

```
The following Business Network Cards are available:
```

```
Connection Profile: hlfv1
```

Card Name	UserId	Business Network
PeerAdmin@hlfv1	PeerAdmin	

```
Issue composer card list --name <Card Name> to get details a specific card
```

```
Command succeeded
```

```
Hyperledger Composer PeerAdmin card has been imported, host of fabric specified as 'localhost'
*** Starting NodeRed. ***
```

```
bcuser@ubuntu16043:~$
```

Verify the installation of Hyperledger Fabric and Hyperledger Composer

8. To see if your blockchain network is up and running, use the command `docker ps -a`. You should see 4 containers with image names like the ones shown below.

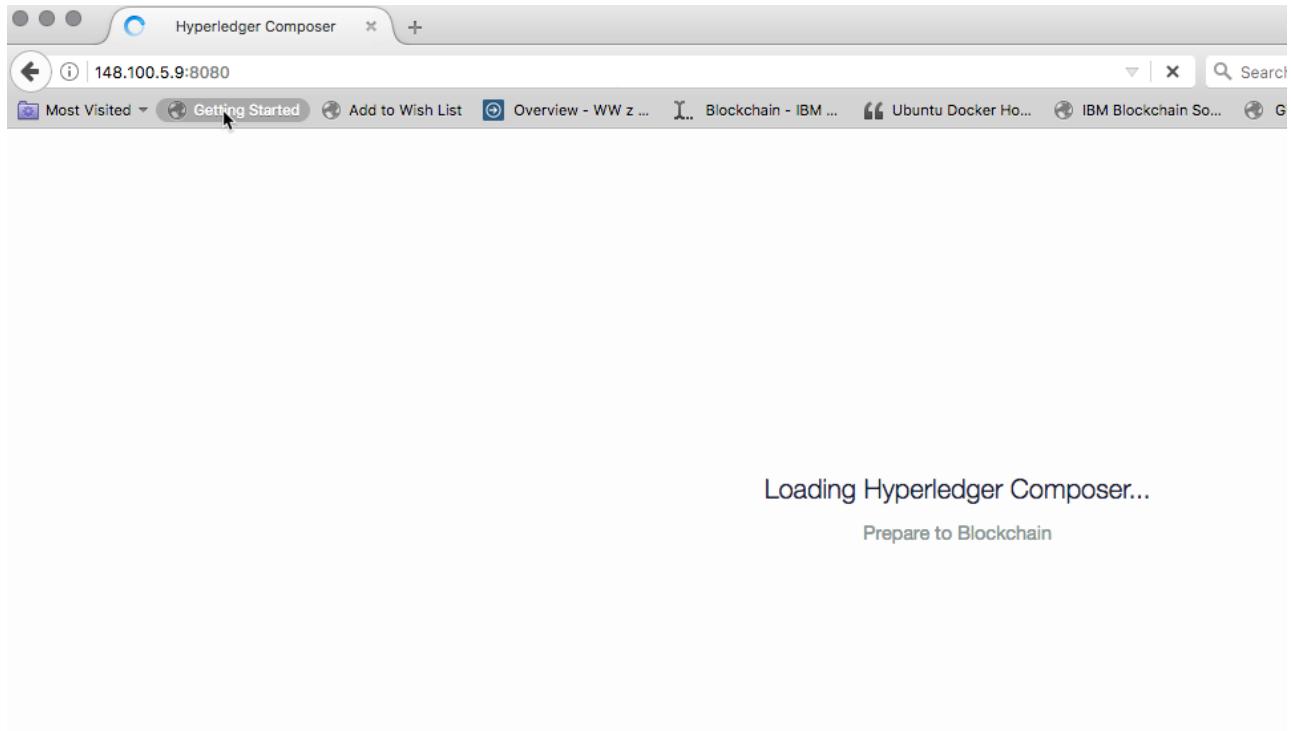
```
bcuser@ubuntu16043:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
aee06b16e8d3        hyperledger/fabric-peer:s390x-1.1.0-alpha   "peer node start"   45 minutes ago   Up 45 minutes    0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp   peer0.org1.example.com
863b9f91dbbd        hyperledger/fabric-ca:s390x-1.1.0-alpha    "sh -c 'fabric-ca-..." 45 minutes ago   Up 45 minutes    0.0.0.0:7054->7054/tcp                           ca.org1.example.com
ca9dc0a09518        hyperledger/fabric-orderer:s390x-1.1.0-alpha  "orderer"          45 minutes ago   Up 45 minutes    0.0.0.0:7050->7050/tcp                           orderer.example.com
b3ff81a569b2        hyperledger/fabric-couchdb:s390x-0.4.5       "tini -- /docke..."  45 minutes ago   Up 45 minutes    4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp   couchdb
bcuser@ubuntu16043:~$
```

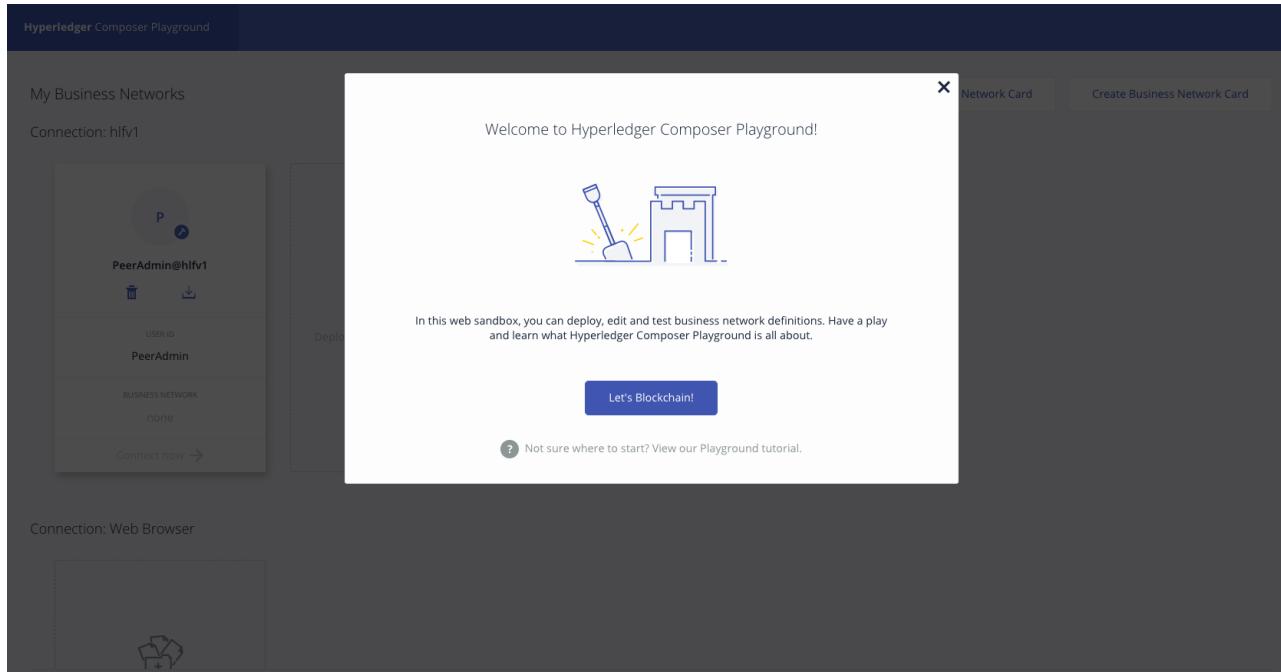
9. Verify Composer Playground is running by looking for its process using the command, `ps -ef | grep playground`.

```
bcuser@ubuntu16043:~$ ps -ef | grep playground
bcuser      5443      1  0 15:06 pts/0    00:00:00 node /home/bcuser/bin/composer-playground
bcuser      5549  4657  0 15:53 pts/0    00:00:00 grep --color=auto playground
bcuser@ubuntu16043:~$
```

10. Open a browser and enter `xxx.xxx.x.x:8080` into the address bar where the x's correspond to your Linux guest's IP address.

- **Note:** It is recommended to use Chrome as your browser for Hyperledger Composer Playground. It is also recommended that you open the Playground in a Incognito Window. This allows you to quickly clear cache and history if you start noticing odd behaviors.
- **Note:** If you use Firefox, you cannot use it in Private mode.
- You should see the following:





11. Congratulations! Part 1 is now complete! Lets get to work on the fun part. 😊

Part 2 — Creating a blockchain application and generating API

Importing the components of your blockchain application

1. On your workstation desktop, you will find a folder named `code`. This folder contains the artifacts you will need for the following steps.
 2. Go back to your browser that has Composer Playground open. If you've closed it, you can open it in your browser by entering `xxx.xxx.x.x:8080` into the address bar where the x's correspond to your Linux guest's IP address.
- **Note:** You will need to view the browser in Full Screen (fully expanded) mode to be able to access everything and prevent issues with inability to scroll on certain screens.

Hyperledger Composer Playground

My Business Networks

Connection: hlfv1

PeerAdmin@hlfv1

User ID: PeerAdmin

BUSINESS NETWORK: none

Connect now →

Welcome to Hyperledger Composer Playground!

In this web sandbox, you can deploy, edit and test business network definitions. Have a play and learn what Hyperledger Composer Playground is all about.

Let's Blockchain!

Not sure where to start? View our Playground tutorial.

Connection: Web Browser

This screenshot shows the Hyperledger Composer Playground interface. On the left, there's a sidebar titled 'My Business Networks' with a connection 'hlfv1'. It lists a peer named 'PeerAdmin@hlfv1', the user ID 'PeerAdmin', and indicates there are 'none' business networks. A 'Connect now' button is present. The main area is a large white box titled 'Welcome to Hyperledger Composer Playground!'. It features a cartoon illustration of a shovel digging into the ground next to a small wall. Below the illustration, text encourages users to deploy, edit, and test business network definitions. A blue button labeled 'Let's Blockchain!' is at the bottom. A note at the bottom right suggests viewing the 'Playground tutorial'. At the very bottom of the main area, there's a question mark icon followed by the text 'Not sure where to start? View our Playground tutorial.' On the far right, there are buttons for 'Network Card' and 'Create Business Network Card'.

3. Select Deploy a new business network under Connection: Web Browser.

My Business Networks

Connection: hlfv1

PeerAdmin@hlfv1

User ID: PeerAdmin

BUSINESS NETWORK: none

Connect now →

Deploy a new business network

Connection: Web Browser

This screenshot shows the same interface as the previous one, but with a red box and arrow highlighting the 'Deploy a new business network' button in the 'Web Browser' section. This step is likely indicating where the user should click to begin creating a new business network.

4. Complete the BASIC INFORMATION.

- Give your new Business Network a name: **blockchain-journey**
- Describe what your Business Network will be used for: **Creating my first blockchain network.**

Hyperledger Composer Playground

← My Wallet

Not sure where to start? View our Playground tutorial.

Deploy New Business Network

1. BASIC INFORMATION

Give your new Business Network a name: **blockchain-journey**

Describe what your Business Network will be used for: **Creating my first blockchain network.**

Give the network admin card that will be created a name **PeerAdmin@blockchain-journey**

blockchain-journey
Creating my first blockchain network.

CONNECTION PROFILE

5. Scroll until you can see *Choose a Business Network Definition to start with:* and select **empty-business-network** and Deploy.

Hyperledger Composer Playground

2. MODEL NETWORK STARTER TEMPLATE

Choose a Business Network Definition to start with:

Choose a sample to play with, start a new project, or import your previous work

empty-business-network 1

Deploy 2

blockchain-journey
Creating my first blockchain network.

CONNECTION PROFILE

BASED ON
empty-business-network

Start from scratch with a blank business network

Contains: 1 Participant Types, 8 Asset Types, and 15 Transaction Types

6. o From *My Wallet* select **Connect now** to go into your business network.

Connection: Web Browser

A

PeerAdmin@blockchain-jour... 1

USER ID
admin

BUSINESS NETWORK
blockchain-journey

Connect now → 2

Deploy a new business network

7. Select **Add a File**.

FILES

v0.0.1

About
README.md

Access Control
permissions.acl

+ Add a file...

Update

This is the readme file for the Business Network Definition created in Playground

8. From the *Add a file* pop-up dialog, select **browse**.

Add a file

Upload a file from your computer...

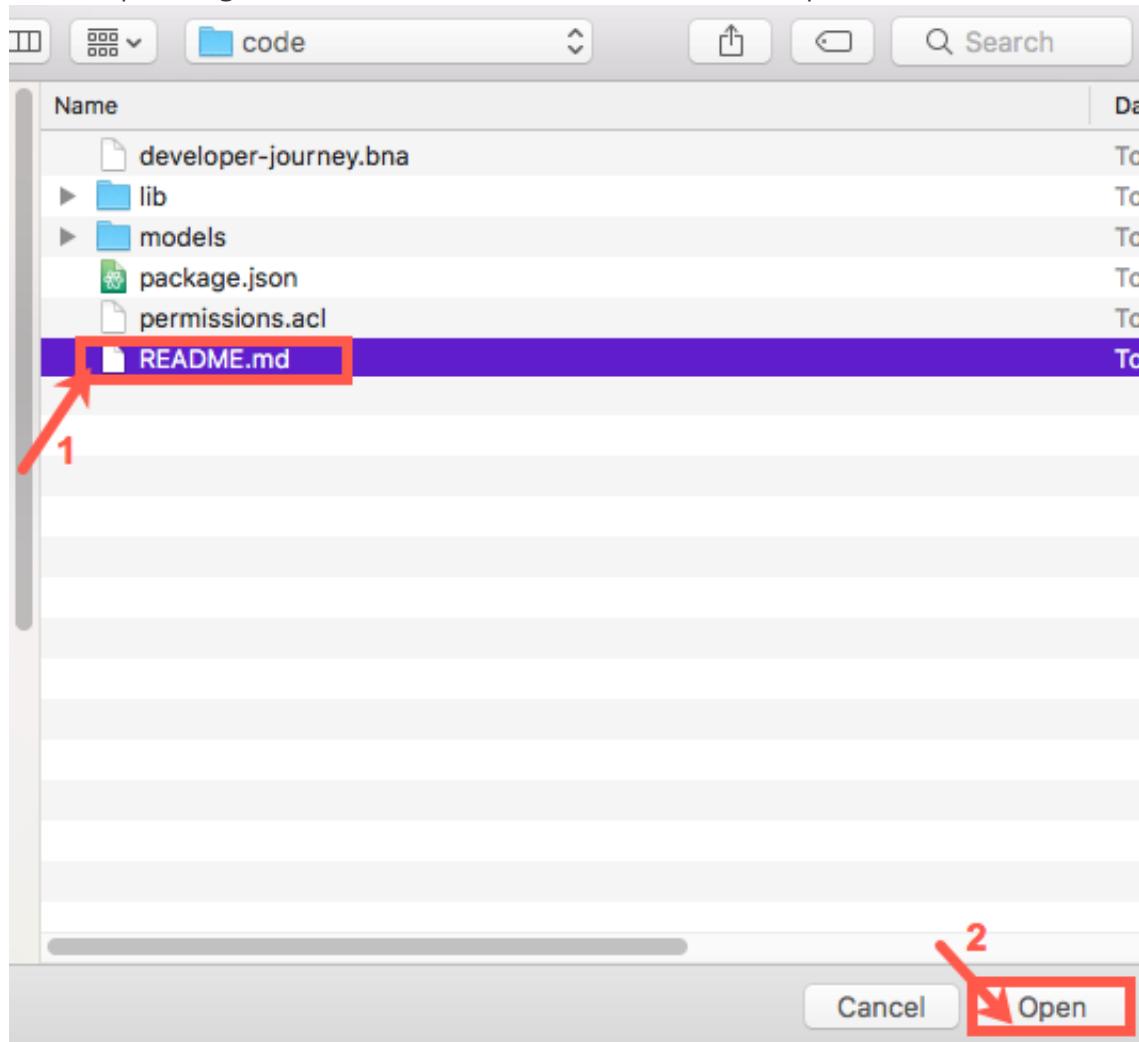
↓ Drop here to upload or browse

Model File (.cto)
Define Assets, Participants and Transactions using Hyperledger Composer modelling langauge.

Script File (.js)
Define the logic of transaction executions using JavaScript.

Cancel

9. In the file explorer window, navigate to where you downloaded the files. Refer to step 5 if you need help finding this location. **Select README.md** and **Click Open**.



10. Select **Add**.

Add a file

Upload a file from your computer...

Drop here to upload
Max file size 5mb

.md README.md Remove File

Add

11. On the *Current file will be replaced* dialog, select **Replace**.

⚠ Current file will be replaced

Your current README file will be replaced with the new one that you are uploading.

Please ensure that you have saved a copy of your README file to disc.

Cancel Replace

12. Let's keep adding the files to the Composer Playground. **Repeat steps 11-15 to add the following files:**

- `org.acme.sample.cto` — This is located in the models folder. In this exercise you'll use this file to create a model for your asset and transactions. You could also create participants in this file.

This is similar to creating a Java class and defining what you would need in the class.

- *logic.js* — This is located in the lib folder. This is a JavaScript file that becomes the brains of your application. In this file is code, your smart contract, that defines how a transaction can happen. This is similar to Java methods.
- **Add last:** *permissions.acl* — This is where you would limit permissions for participants in a blockchain network.

13. Your files are all now loaded into Composer Playground. **Click Update** on the left side of the browser.

The screenshot shows the Composer Playground interface. The top navigation bar has tabs for 'Define' and 'Test', with 'Define' being the active tab. On the left, there's a sidebar with sections for 'FILES' (containing 'node' and 'README.md'), 'Model File' (containing 'models/org.acme.sample.cto'), and 'Script File' (containing 'lib/logic.js'). Below the sidebar, there's a button labeled '+ Add a file...'. At the bottom left of the main area, there's a red box highlighting the 'Update' button. The main area displays the 'logic.js' script content:

```
1
2
3  /**
4   * Sample transaction processor function.
5   */
6
7
8
9  function onSetSensorTemp(setSensorTemp) {
10    setSensorTemp.gauge.sensorTemp = setSensorTemp.newSensorValue;
11    return getAssetRegistry('org.acme.sample.Sensor')
12      .then(function (assetRegistry) {
13        return assetRegistry.update(setSensorTemp.gauge);
14      });
15 }
16
17 function onChangeThermostatTemp(changeThermostat) {
18   var diff = Math.abs(changeThermostat.thermostat.sensorTemp -
19     changeThermostat.newThermostatValue);
20   if (diff <= 3) {
```

At the bottom right of the editor, there's a message: 'Everything looks good!'

Creating your blockchain application

14. Click on **Model File**.

```

1 /**
2  * business network definition.
3 */
4 namespace org.acme.sample
5
6 asset Sensor identified by teamID {
7   //create your Team asset model
8   o String teamID
9 }
10
11 transaction SetSensorTemp {
12   //create your SetSensorTemp transaction model
13 }
14
15 transaction ChangeThermostatTemp {
16   //create your ChangeThermostatTemp transaction model
17 }
18
19 transaction CompareWeather{
20   //create your CompareWeather transaction model
21 }
22
23
24
25
26

```

Everything looks good!
Any problems detected in your code would be reported here

15. Click in the **editor** on the right to begin writing your models.

- NOTE: **DO NOT** modify the namespace during the lab.

```

1 /**
2  * business network definition.
3 */
4 namespace org.acme.sample
5
6 asset Sensor identified by teamID {
7   //create your Team asset model
8   o String teamID ←
9 }
10
11 transaction SetSensorTemp {
12   //create your SetSensorTemp transaction model
13 }
14
15 transaction ChangeThermostatTemp {
16   //create your ChangeThermostatTemp transaction model
17 }
18
19 transaction CompareWeather{
20   //create your CompareWeather transaction model
21 }
22
23
24
25
26

```

Everything looks good!
Any problems detected in your code would be reported here

16. On a new line, give your asset `sensor` the following attributes.

- Note: a small "o" is used as a bullet in the model.
- `o String teamID` — this will be the value that is assigned to your team. (already there!)
- `o String teamName` — this could be anything! Come up with something clever!
- `o Double sensorTemp` — temperature from the Raspberry Pi will be stored here.
- `o Double thermostatTemp` — you will create a temperature for the thermostat.
- `o String recommendation` — this will be populated based on the `CompareWeather` transaction.
- **Click Update** to save changes.

Web blockchain-journey

Define Test

FILES

About
README.md

Model File
models/org.acme.sample.cto

Script File
lib/logic.js

+ Add a file...

Update 2

Model File models/org.acme.sample.cto Everything looks good!

```

1  /**
2  * business network definition.
3  */
4  namespace org.acme.sample
5
6  asset Sensor identified by teamID {
7      //create your Team asset model
8      o String teamID
9      o String teamName
10     o Double sensorTemp
11     o Double thermostatTemp
12     o String recommendation
13 }
14
15 transaction SetSensorTemp {
16     //create your SetSensorTemp transaction model
17 }
18
19
20 transaction ChangeThermostatTemp {

```

17. Now create your first transaction model for `SetSensorTemp`. Enter the following attributes:

- `--> Sensor gauge` — The transaction will need to put data into the `Sensor` asset. This passes a reference to the asset so we can work with the asset in the logic for the transaction.
- `o Double newSensorValue` — This is the variable that will be set by the temperature passed into the transaction from the NodeRed Sensor for picking up temperature.
- Click `*Deploy` to save changes.

Web blockchain-journey

Define Test admin

FILES

About
README.md

Model File
models/org.acme.sample.cto

Script File
lib/logic.js

+ Add a file...

Update 2

Model File models/org.acme.sample.cto Everything looks good!

```

12     o String recommendation
13 }
14
15 transaction SetSensorTemp {
16     //create your SetSensorTemp transaction model
17     --> Sensor gauge
18     o Double newSensorValue
19 }
20
21 transaction ChangeThermostatTemp {
22     //create your ChangeThermostatTemp transaction model
23 }
24
25 transaction CompareWeather{
26     //create your CompareWeather transaction model
27 }
28
29 }
30
31

```

18. Build your `ChangeThermostatTemp` transaction model. Add the following:

- `--> Sensor thermostat` — The transaction will need to put data into the `sensor` asset for the thermostat. This passes a reference to the asset so we can work with the asset in the logic for the transaction.
- `o Double newThermostatValue` — This allows for a new, proposed value to be sent into the transaction. In the logic tab, we will use this value to compare to what the gauge says and decide if the thermostat value should be adjusted.
- **Click *Update*** to save changes.

```
12     o String recommendation
13   }
14
15   transaction SetSensorTemp {
16     //create your SetSensorTemp transaction model
17     --> Sensor gauge
18     o Double newSensorValue
19   }
20
21   transaction ChangeThermostatTemp {
22     //create your ChangeThermostatTemp transaction model
23     --> Sensor thermostat
24     o Double newThermostatValue
25   }
26
27   transaction CompareWeather{
28     //create your CompareWeather transaction model
29
30   }
31
```

Model File `models/org.acme.sample.cto`

FILES

- About
- `README.md`

Model File
`models/org.acme.sample.cto`

Script File
`lib/logic.js`

+ Add a file...

Update

Import/Replace

Define Test admin

Model File `models/org.acme.sample.cto`

12 o String recommendation
13 }
14
15 transaction SetSensorTemp {
16 //create your SetSensorTemp transaction model
17 --> Sensor gauge
18 o Double newSensorValue
19 }
20
21 transaction ChangeThermostatTemp {
22 //create your ChangeThermostatTemp transaction model
23 --> Sensor thermostat
24 o Double newThermostatValue
25 }
26
27 transaction CompareWeather{
28 //create your CompareWeather transaction model
29
30 }
31

Everything looks good!

Any problems detected in your code would be reported here

19. Enter the following values to build your `compareWeather` transaction model:

- `--> Sensor recommend` — The transaction will need to put data into the `sensor` asset. This passes a reference to the asset so we can work with the asset in the logic for the transaction.
- `o Double outsideTemp` — Looking at the [WeatherUnderground.com API](#) for Conditions, you can see all of the possible data that the call could return. Based on the data, it was decided to take the actual outside temperature and the feels like temperature to give a recommendation on thermostat settings. This variable stores the value passed into it via NodeRed from Weather.com for the outside temperature. The model on the API page shows up whether the data is returned in Celsius or Fahrenheit and its variable type. In this exercise we will use Celsius.
- `o Double feelsLike` — the variable to store the `feels_like` value from Weather.com.
- **Click *Update*** to save changes.

Web blockchain-journey

Define Test admin

Model File models/org.acme.sample.cto

FILES

- About
- README.md
- Model File** models/org.acme.sample.cto
- Script File** lib/logic.js
- + Add a file...

Update 

 Import/Replace

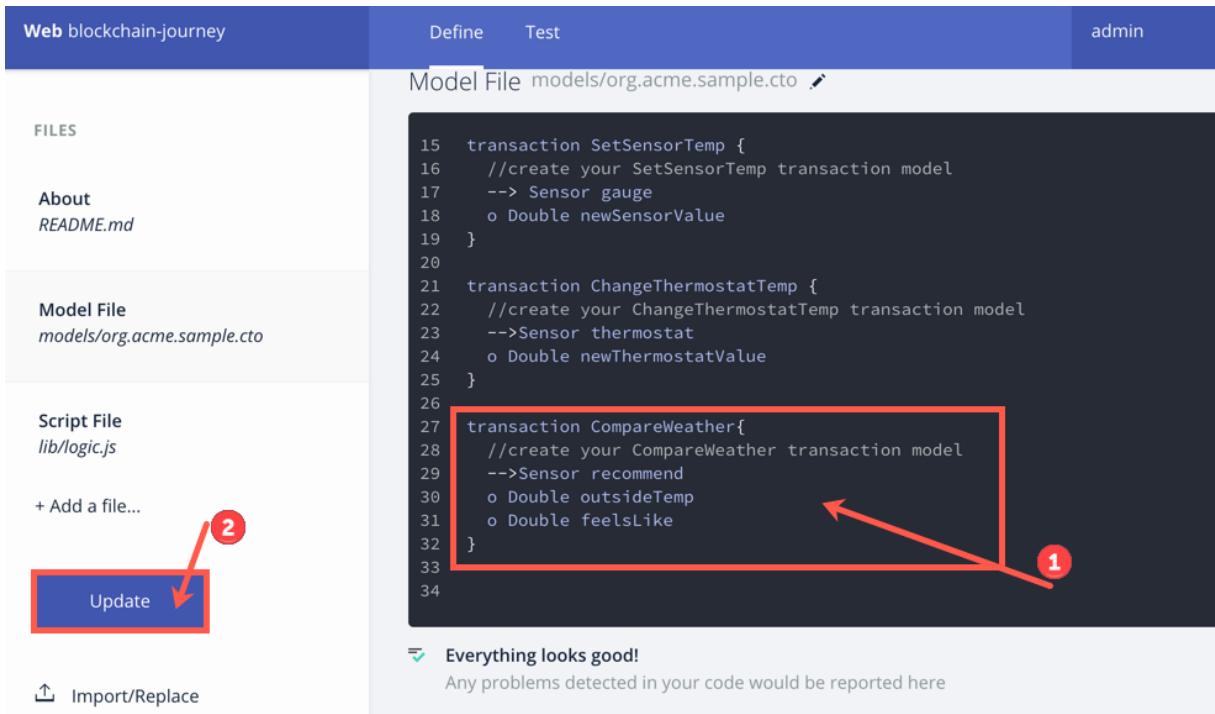
```

15  transaction SetSensorTemp {
16      //create your SetSensorTemp transaction model
17      --> Sensor gauge
18      o Double newSensorValue
19  }
20
21  transaction ChangeThermostatTemp {
22      //create your ChangeThermostatTemp transaction model
23      --> Sensor thermostat
24      o Double newThermostatValue
25  }
26
27  transaction CompareWeather{
28      //create your CompareWeather transaction model
29      --> Sensor recommend
30      o Double outsideTemp
31      o Double feelsLike
32  }
33
34

```

 1

 **Everything looks good!**
Any problems detected in your code would be reported here



20. Click on the **Script File** tab.

FILES

- About
- README.md
- Model File** models/org.acme.sample.cto
- Script File** lib/logic.js 
- Access Control permissions.acl
- + Add a file...

Deploy

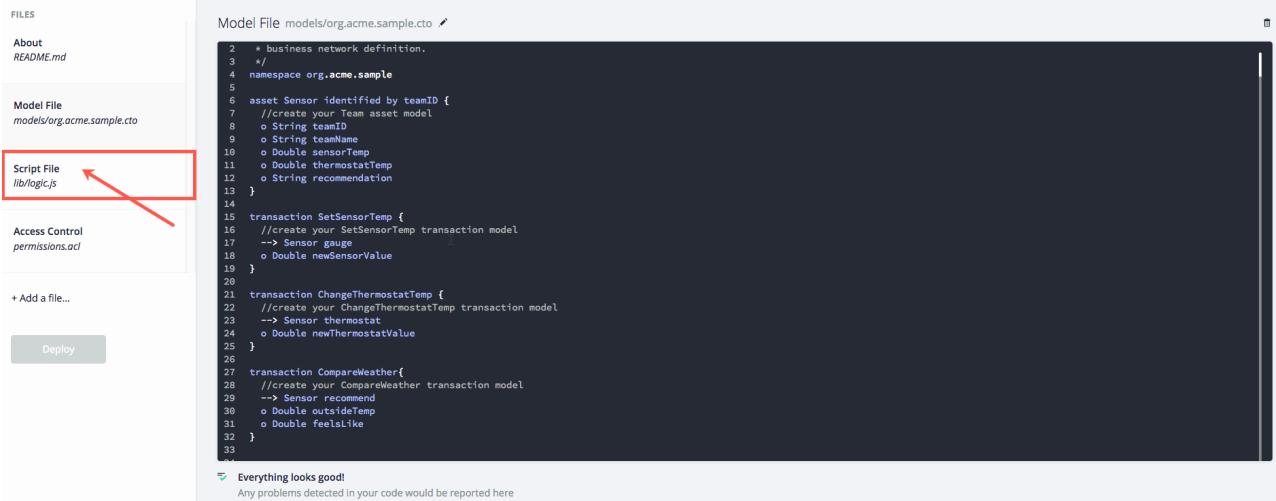
 2

```

2  * business network definition.
3  */
4  namespace org.acme.sample
5
6  asset Sensor identified by teamID {
7      //create your team asset model
8      o String teamID
9      o String teamName
10     o Double sensorTemp
11     o Double thermostatTemp
12     o String recommendation
13 }
14
15  transaction SetSensorTemp {
16      //create your SetSensorTemp transaction model
17      --> Sensor gauge
18      o Double newSensorValue
19  }
20
21  transaction ChangeThermostatTemp {
22      //create your ChangeThermostatTemp transaction model
23      --> Sensor thermostat
24      o Double newThermostatValue
25  }
26
27  transaction CompareWeather{
28      //create your CompareWeather transaction model
29      --> Sensor recommend
30      o Double outsideTemp
31      o Double feelsLike
32  }
33

```

 **Everything looks good!**
Any problems detected in your code would be reported here



21. Review the code in the editor. Verify that your variable names match the variable names here. Capitalization does matter! If names don't match, you'll have errors.

- Any guesses what the code is doing for each transaction?

```

1
2
3  /**
4   * Sample transaction processor function.
5  */
6
7
8
9  function onSetSensorTemp(setSensorTemp) {
10    setSensorTemp.gauge.sensorTemp = setSensorTemp.newSensorValue;
11    return getAssetRegistry('org.acme.sample.Sensor')
12      .then(function (assetRegistry) {
13        return assetRegistry.update(setSensorTemp.gauge);
14      });
15  }
16
17  function onChangeThermostatTemp(changeThermostat) {
18    var diff = Math.abs(changeThermostat.thermostat.sensorTemp - changeThermostat.newThermostatValue);
19    if (diff < 3) {
20      changeThermostat.thermostat.thermostatTemp = changeThermostat.newThermostatValue;
21      return getAssetRegistry('org.acme.sample.Sensor')
22        .then(function (assetRegistry) {
23          return assetRegistry.update(changeThermostat.thermostat);
24        });
25    } else {
26      //reject transaction
27      throw new Error("You do not have permission to change the temperature.");
28    }
29  }
30
31  function onCompareWeather(compareWeather) {

```

Everything looks good!
Any problems detected in your code would be reported here

Test application code

- Click on the **Test** tab at the top to try out your code.

Hyperledger Composer Playground

Define Test

FILES

About README.md

Model File models/org.acme.sample.cto

Script File lib/logic.js

```

1
2
3  /**
4   * Sample transaction processor f
5  */
6
7

```

- Notice that in this particular case because we have no participants, the **Test** tab has opened to the **Asset** menu on the left. You must have an asset to be able to run any of the transactions.

- Click **Create New Asset**.

Hyperledger Composer Playground

Define Test

admin

PARTICIPANTS

ASSETS

Sensor

TRANSACTIONS

All Transactions

Submit Transaction

Asset registry for org.acme.sample.Sensor

ID Data

+ Create New Asset

This registry is empty!

To create resources in this registry click create new at the top of this page

- Create an example asset to test your code by filling in the following information:

- "teamID": "teamID:**xxx**" where ** xxx ** is any team number you'd like.

- `"teamName": ""` — this could be any name you'd like. Be clever! :bowtie:
- `"sensorTemp": **0**` — Set **0** to any value. When you work with NodeRed, temperatures will be in Celsius.
- `"thermostatTemp": **0**` — Set **0** to any value. This is initializing your thermostat so pick a value you want to work with.
- `"recommendation": ""` — Leave this as is.
- *Make a note somewhere* of the values you used for `sensorTemp` and `thermostatTemp`.

Create New Asset X

In registry: **org.acme.sample.Sensor**

JSON Data Preview

```

1  {
2    "$class": "org.acme.sample.Sensor",
3    "teamID": "01",
4    "teamName": "Linux",           ←
5    "sensorTemp": 17,
6    "thermostatTemp": 20,
7    "recommendation": "Id reprehenderit in culpa amet."
8  }

```

Optional Properties

Just need quick test data? [Generate Random Data](#)

[Cancel](#) [Create New](#)

25. Click **Create New**.

Create New Asset

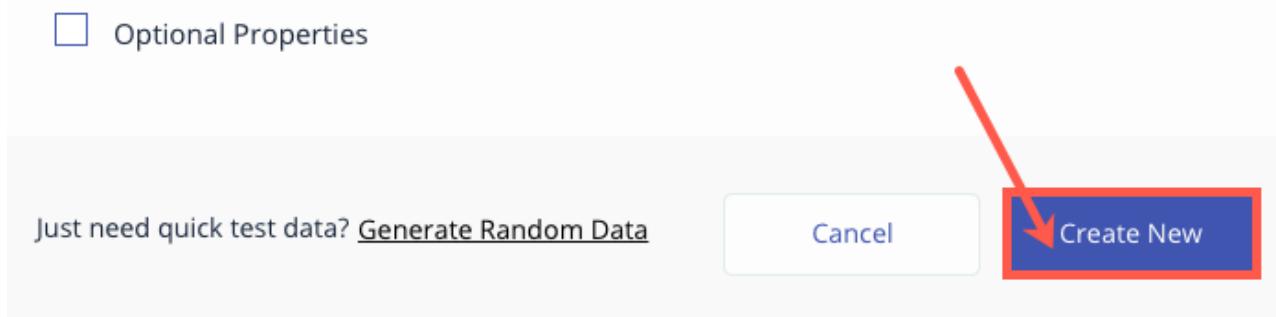
In registry: **org.acme.sample.Sensor**

JSON Data Preview

```

1  {
2    "$class": "org.acme.sample.Sensor",
3    "teamID": "01",
4    "teamName": "Linux",
5    "sensorTemp": 17,
6    "thermostatTemp": 20,
7    "recommendation": "Id reprehenderit in culpa amet."
8  }

```



26. Once your **Team** asset is created it should show in the registry as shown below.

PARTICIPANTS		Asset registry for org.acme.sample.Sensor		+ Create New Asset
ASSETS		ID	Data	
Sensor		01	<pre>{ "\$class": "org.acme.sample.Sensor", "teamID": "01", "teamName": "Linux", "sensorTemp": 17, "thermostatTemp": 20, "recommendation": "Id reprehenderit in culpa amet." }</pre>	
TRANSACTIONS				Collapse
All Transactions				

[Submit Transaction](#)

27. You're ready to run your first transaction. **Click** on *Submit Transaction*.

The screenshot shows the Asset registry interface for the org.acme.sample.Sensor class. On the left, there are tabs for PARTICIPANTS, ASSETS (which is selected), and TRANSACTIONS. Under ASSETS, 'Sensor' is listed. Under TRANSACTIONS, 'All Transactions' is selected. On the right, the 'Asset registry for org.acme.sample.Sensor' page displays a table with columns 'ID' and 'Data'. One row is shown with ID '01' and the following JSON data:

```
{
  "sclass": "org.acme.sample.Sensor",
  "teamName": "Linux",
  "sensorTemp": 17,
  "thermostatTemp": 20,
  "recommendation": "Id reprehenderit in culpa amet."
}
```

A red box highlights the 'Submit Transaction' button at the bottom-left of the interface. A red arrow points from this button towards the JSON data table.

28. The **Submit Transaction** dialog will open a new window.

- Make sure that the **Transaction Type** is set to `SetSensorTemp`.
- Modify the JSON data `"gauge": "resource:org.acme.sample.Sensor#xxxx"` — enter your team's identifier in place of the value where **xxxx** is in the sample JSON data.
- Modify the JSON data `"newSensorValue": 0` — enter a value your sensor could have.
- Click **Submit**.

Submit Transaction

Transaction Type

SetSensorTemp

▼

1

JSON Data Preview

```
1  {
2    "$class": "org.acme.sample.SetSensorTemp".
3    "gauge": "resource:org.acme.sample.Sensor#01",
4    "newSensorValue": 18
5 }
```

2

Optional Properties

Just need quick test data? [Generate Random Data](#)

Cancel

Submit

29. If you submitted the transaction with your correct team ID, then you should have a transaction showing under *All Transactions*. **Click view record** to see the data you entered in the prior step. Congratulations! You've now completed a transaction. 

PARTICIPANTS	Date, Time	Entry Type	Participant	
ASSETS				
Sensor	2017-12-06, 16:37:35	SetSensorTemp	admin (NetworkAdmin)	view record

30. Verify that `setsensorTemp` updated the `sensorTemp` value in your asset. Click **Sensor**.

PARTICIPANTS	Date, Time	Entry Type	Participant	
ASSETS				
Sensor	2017-12-06, 16:37:35	SetSensorTemp	admin (NetworkAdmin)	view record

TRANSACTIONS

All Transactions				
2017-12-06, 16:37:35				view record

31. Check the `sensorTemp` value. Does it have the new value from the `SetSensorTemp` transaction?

Asset registry for org.acme.sample.Sensor

+ Create New Asset

ID	Data
01	<pre>{ "\$class": "org.acme.sample.Sensor", "teamID": "01", "teamName": "Linux", "sensorTemp": 18, "thermostatTemp": 20, "recommendation": "Id reprehenderit in culpa amet." }</pre>

Collapse

Submit Transaction

32. Let's do another transaction. Select **Submit Transaction**.

Asset registry for org.acme.sample.Sensor

+ Create New Asset

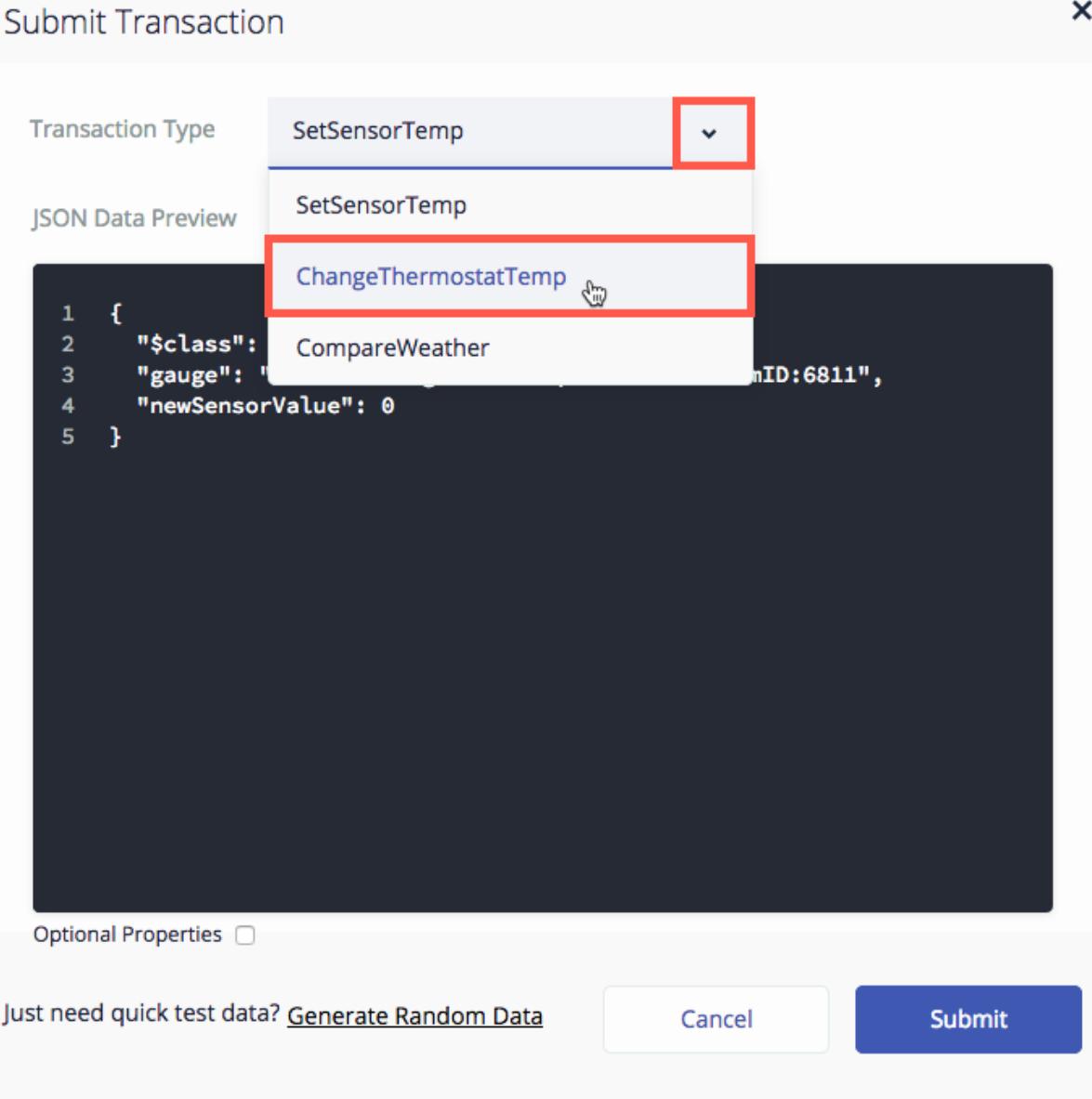
ID	Data
01	<pre>{ "\$class": "org.acme.sample.Sensor", "teamID": "01", "teamName": "Linux", "sensorTemp": 18, "thermostatTemp": 20, "recommendation": "Id reprehenderit in culpa amet." }</pre>

Collapse

Submit Transaction

33. This time let's run, `ChangeThermostatTemp`.

- In the **Transaction Type** drop down, select `ChangeThermostatTemp`.



- Edit the sample JSON for the transaction `"thermostat": "resource:org.acme.sample.Sensor#xxxx"` — change **xxxx** to your team ID value.
- Edit the sample JSON for the transaction `"newThermostatValue": 0` — Replace **0** with a value to which you would like to see if you can adjust the thermostat.
- Click **Submit**.

Submit Transaction



Transaction Type

ChangeThermostatTemp



JSON Data Preview

```
1  {
2    "$class": "org.acme.sample.ChangeThermostatTemp"
3    "thermostat": "resource:org.acme.sample.Sensor#01",
4    "newThermostatValue": 16
5 }
```



Optional Properties

Just need quick test data? [Generate Random Data](#)

[Cancel](#)

[Submit](#)

- If you select a temperature for the thermostat that is not within 3 degrees of the `sensorTemp` value, then you will get an error message like the one below. If you get this message, enter another value and click submit.

Optional Properties

Error: Too much difference! Current sensor reading 17

- If you do have permission to adjust the thermostat, you will be returned back to the transaction registry where you can see the data you just submitted.

Date, Time	Entry Type	Participant	
2017-12-06, 16:47:00	ChangeThermostatTemp	admin (NetworkAdmin)	view record
2017-12-06, 16:37:35	SetSensorTemp	admin (NetworkAdmin)	view record

- If for some reason you forget to modify your teamID value or update it to the wrong value, you will see an error like the one shown below. Check your value for teamID and try again.

Submit Transaction

Transaction Type **ChangeThermostatTemp**

JSON Data Preview

```

1  {
2    "$class": "org.acme.sample.ChangeThermostatTemp",
3    "thermostat": "resource:org.acme.sample.Sensor#teamID:2760",
4    "newThermostatValue": 0
5  }

```

Optional Properties

Error: Object with ID 'teamID:2760' in collection with ID 'Asset:org.acme.sample.Sensor' does not exist

34. Verify that the last transaction updated your asset. Click **Sensor**.

PARTICIPANTS	Date, Time	Entry Type	Participant	
ASSETS	2017-12-06, 16:47:00	ChangeThermostatTemp	admin (NetworkAdmin)	view record
TRANSACTIONS	2017-12-06, 16:37:35	SetSensorTemp	admin (NetworkAdmin)	view record
	2017-12-06, 16:31:41	AddAsset	admin (NetworkAdmin)	view record

35. Verify that the `thermostatTemp` attribute for your Team has been updated to the value you

gave successfully in the `ChangeThermostatTemp` transaction.

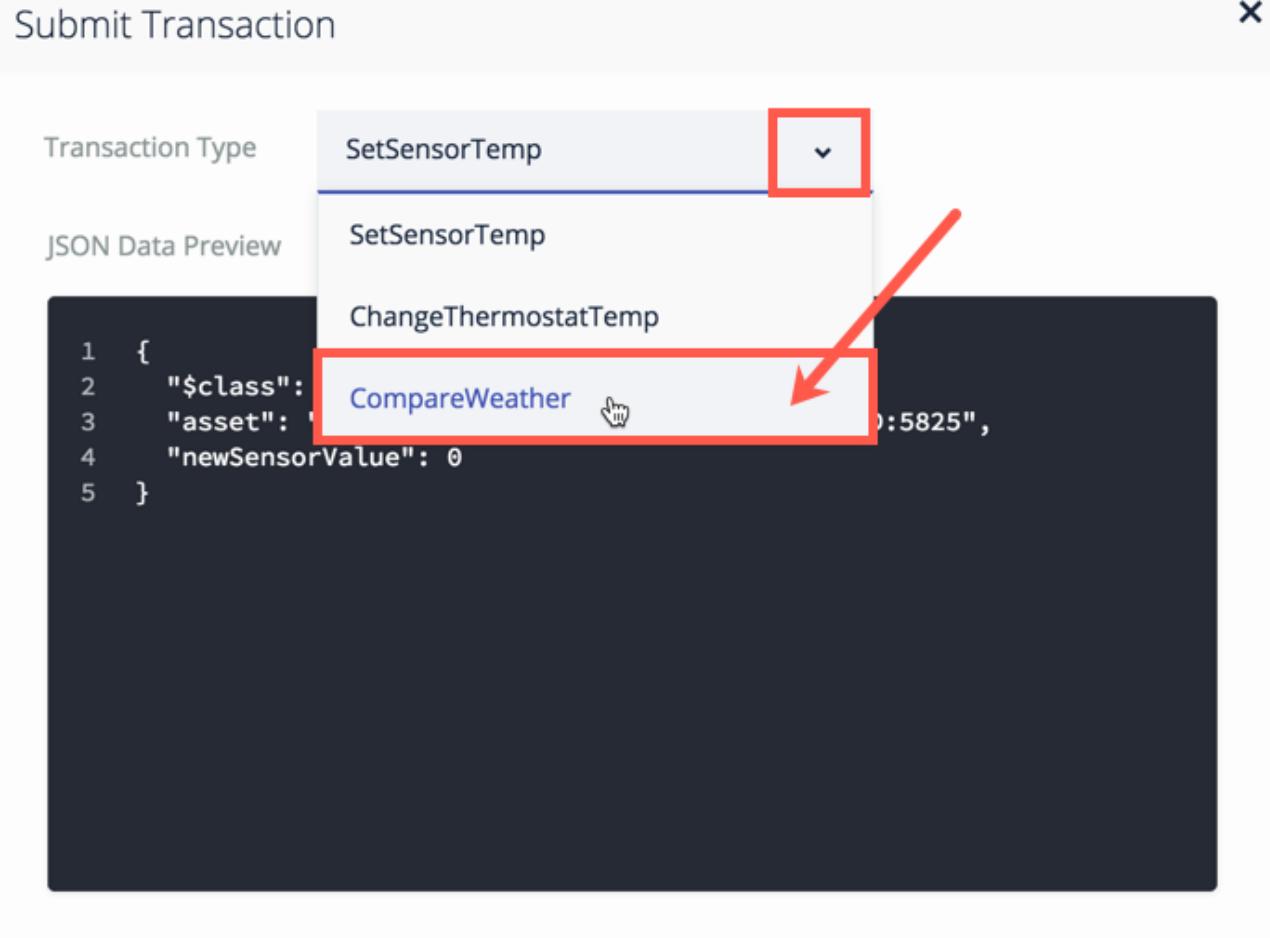
- **Note:** In step 40, you can verify that the thermostat was originally set to 20 and is now set to 16.

The screenshot shows the Asset registry interface for the `org.acme.sample.Sensor` class. On the left, there are navigation tabs for **PARTICIPANTS**, **ASSETS** (selected), and **TRANSACTIONS**. Under **ASSETS**, a single asset named `Sensor` is listed with ID `01`. The **Data** pane displays the JSON object for this asset, which includes fields like `teamID`, `teamName`, `sensorTemp`, and `thermostatTemp`. The `thermostatTemp` field is highlighted with a red box and contains the value `16`. A blue button at the bottom left labeled `Submit Transaction` is visible.

36. Time to work with the `compareWeather` transaction. Click **Submit Transaction**.

This screenshot is similar to the previous one, showing the Asset registry for `org.acme.sample.Sensor`. The `thermostatTemp` field in the JSON data is now set to `16`, indicating a change has been made. A red arrow points from the text "36. Select CompareWeather from the Transaction Type drop down." to the `Submit Transaction` button, which is also highlighted with a red box.

37. Select **CompareWeather** from the *Transaction Type* drop down.



Just need quick test data? [Generate Random Data](#)

[Cancel](#)

[Submit](#)

38. Complete the **CompareWeather** transaction.

- Modify the JSON, `"recommend": "resource:org.acme.sample.Sensor#xxxx"` — Replace **xxxx** with your team ID.
- Modify the JSON for `"outsideTemp": 0` — Enter a value for an outside temperature.
- Edit the JSON for `"feelsLike": 0` — Enter a value for what temperature it could feel like outside.
- Click **Submit**.

Submit Transaction



Transaction Type

CompareWeather



JSON Data Preview

```
1  {
2    "$class": "org.acme.sample.CompareWeather",
3    "recommend": "resource:org.acme.sample.Sensor#01",
4    "outsideTemp": 22,
5    "feelsLike": 23
6 }
```



Optional Properties

Just need quick test data? [Generate Random Data](#)

[Cancel](#) [Submit !\[\]\(9a3e5920fed70eeec050cd56697d1948_img.jpg\)](#)

39. Verify that your transaction is showing in the Transaction Registry.

PARTICIPANTS	Date, Time	Entry Type	Participant	
ASSETS				
Sensor	2017-12-06, 16:54:48	CompareWeather	admin (NetworkAdmin)	view record
TRANSACTIONS				
All Transactions	2017-12-06, 16:47:00	ChangeThermostaTemp	admin (NetworkAdmin)	view record
	2017-12-06, 16:37:35	SetSensorTemp	admin (NetworkAdmin)	view record
				view record

40. Click on **Sensor**.

PARTICIPANTS					
ASSETS		DATE, TIME		ENTRY TYPE	
Sensor		2017-12-06, 16:54:48		CompareWeather	admin (NetworkAdmin) view record
All Transactions		2017-12-06, 16:47:00		ChangeThermostatTemp	admin (NetworkAdmin) view record
		2017-12-06, 16:37:35		SetSensorTemp	admin (NetworkAdmin) view record
Submit Transaction		View Assets		View Transactions	

41. Verify there is now a message in the `recommendation` variable in your Team asset and that the `thermostatValue` has been updated to the recommended value.

PARTICIPANTS					
ASSETS		ID		DATA	
Sensor		01		<pre>{ "class": "org.acme.sample.Sensor", "teamID": "01", "teamName": "Linux", "sensorTemp": 16, "thermostatTemp": 20, "recommendation": "it feels quite nice! The recommended thermostat setting is 20 C. The thermostat is being adjusted from 16." }</pre>	Edit Delete
All Transactions				Collapse	
Submit Transaction		View Assets		View Transactions	

42. Continue testing your code for all scenarios to understand what your contract(s) can do. The hints to the remaining scenarios are as follows: (Yes, you'll have to look at the Script File under the Define Tab to figure out the criteria.)

- ChangeThermostatTemp:
 - A successful transaction where the `thermostatValue` is updated in the Sensor asset.
 - An error message in the *Submit Transaction* window advising you do not have permission to adjust the thermostat.
- CompareWeather:
 - A transaction based on `outsideTemp` values where it is really hot.
 - A transaction based on `outsideTemp` values where it is quite nice.
 - A transaction based on `outsideTemp` values where it is cold.
 - A transaction based on `feelsLike` values where it is hot.
 - A transaction based on `feelsLike` values where it is quite nice.
 - A transaction based on `feelsLike` values where it is cold.
- **Note:** You should verify that your asset values have been updated appropriately after each transaction like you did in prior steps.

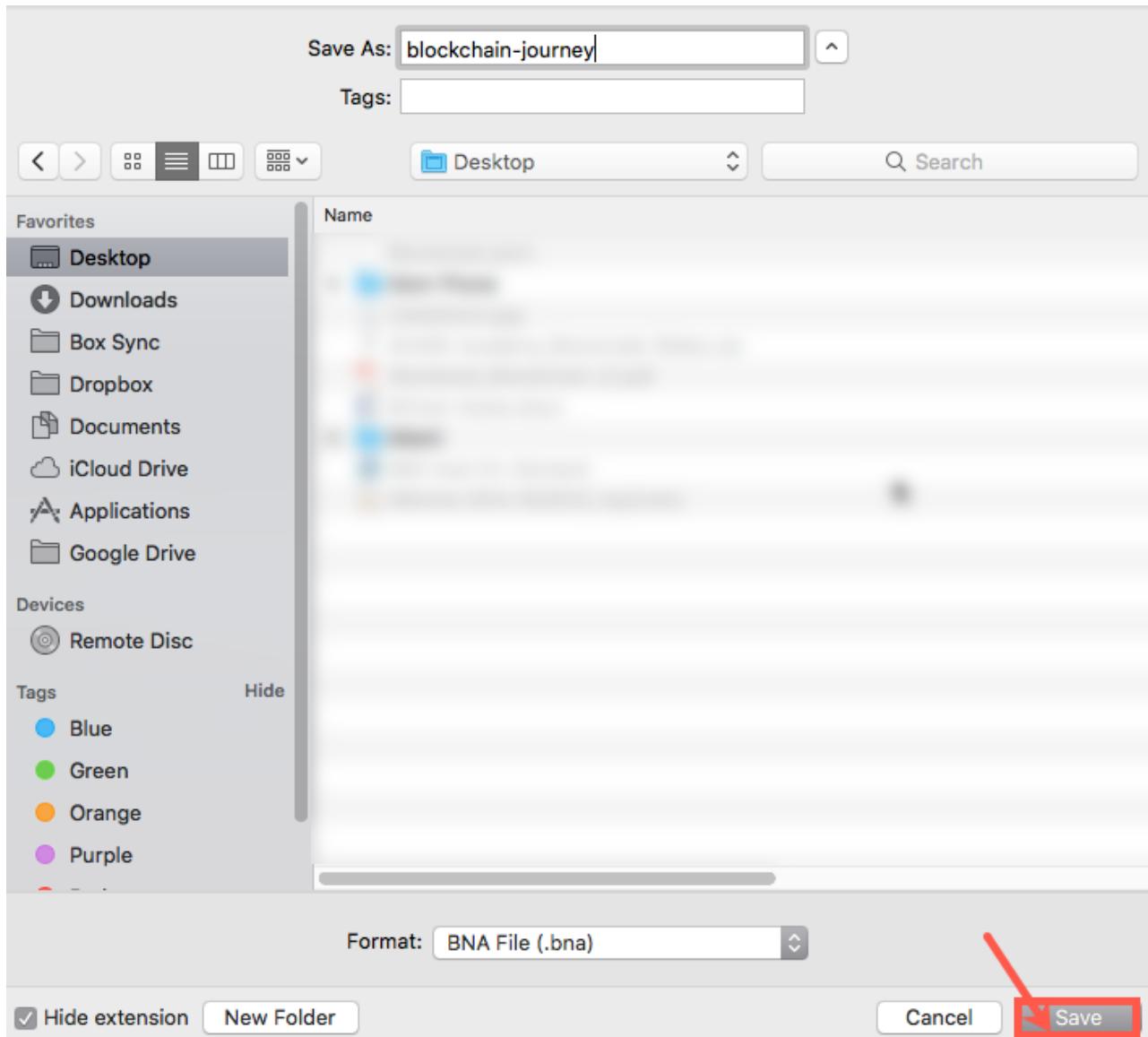
Deploy application to Hyperledger Fabric

43. Back in your browser where Hyperledger Composer Playground is running, **click** the *Define* tab and then **click** *Export* to save your code to your desktop. This is a safety measure. Export saves all of the individual files we imported at the beginning of Part 2 into a compressed file called a business network archive (.bna).

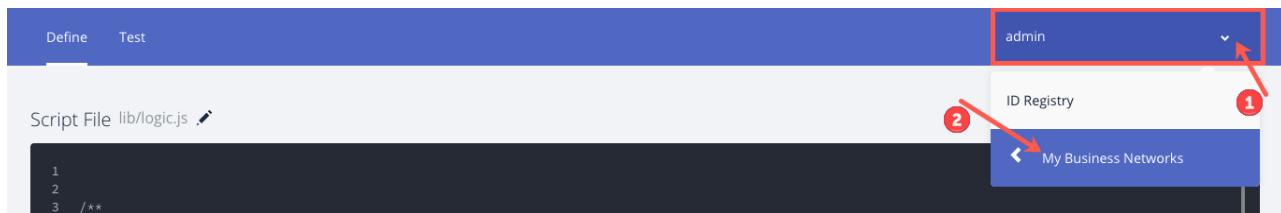
The screenshot shows the Hyperledger Composer Playground web interface. At the top, there's a navigation bar with 'Web blockchain-journey' on the left, 'admin' on the right, and a dropdown menu. Below the navigation bar is a sidebar on the left containing links for 'About', 'README.md', 'Model File', 'Script File', and 'Access Control'. In the center, there's a main workspace titled 'Script File lib/logic.js' with a code editor containing JavaScript logic for a smart contract. The code includes functions like 'onSetSensorTemp', 'onChangeThermostatTemp', and 'onCompareWeather'. Below the code editor, a message says 'Everything looks good!' with a note about no problems detected. At the bottom left of the workspace, there are buttons for 'Import/Replace' and 'Export', with 'Export' highlighted by a red box and a red circle labeled '2'. The bottom right of the workspace shows links for 'Playground v0.16.0', 'Tutorial', 'Docs', and 'Community'.

```
1
2
3
4
5
6
7
8
9 function onSetSensorTemp(setSensorTemp) {
10     setSensorTemp.gauge.sensorTemp = setSensorTemp.newSensorValue;
11     return getAssetRegistry('org.acme.sample.Sensor')
12         .then(function (assetRegistry) {
13             return assetRegistry.update(setSensorTemp.gauge);
14         });
15 }
16
17 function onChangeThermostatTemp(changeThermostat) {
18     var diff = Math.abs(changeThermostat.thermostat.sensorTemp - changeThermostat.newThermostatValue);
19     if (diff <= 3) {
20         changeThermostat.thermostat.thermostatTemp = changeThermostat.newThermostatValue;
21         return getAssetRegistry('org.acme.sample.Sensor')
22             .then(function (assetRegistry) {
23                 return assetRegistry.update(changeThermostat.thermostat);
24             });
25     } else {
26         //reject transaction
27         throw new Error("Too much difference! Current sensor reading " + changeThermostat.thermostat.sensorTemp);
28     }
29 }
30
31 function onCompareWeather(compareWeather) {
32 }
```

44. In the pop-up dialog, **choose** your directory location and **click Save**.



45. In the upper right corner of your browser, **select admin** and **click My Business Networks**.



46. In the middle of the page, **click Deploy a new business network** under the **Connection: hlfv1** business network.

Web blockchain-journey

My Business Networks

Connection: hlfv1

PeerAdmin@hlfv1

User ID: PeerAdmin

Business Network: none

Connect now →

Import Business Network Card

Create Business Network Card

47. Complete the fields under *Basic Information* and then click *Drop here to upload or browse*.

- Business Network: `hlfv1-blockchain-journey`
- Description: "Blockchain Journey network deployed to hlfv1"
- Network Admin Card: `admin@hlfv1-blockchain-journey`

Hyperledger Composer Playground

Deploy New Business Network

1. BASIC INFORMATION

Give your new Business Network a name: ① `hlfv1-blockchain-journey`

Describe what your Business Network will be used for: ② `Blockchain Journey network deployed to hlfv1`

Give the network admin card that will be created name: ③ `admin@hlfv1-blockchain-journey`

2. MODEL NETWORK STARTER TEMPLATE

Choose a Business Network Definition to start with:

Choose a sample to play with, start a new project, or import your previous work

basic-sample-network

empty-business-network

Samples on npm

Drop here to upload or browse ④

hlfv1-blockchain-journey

Blockchain Journey network deployed to hlfv1

CONNECTION PROFILE

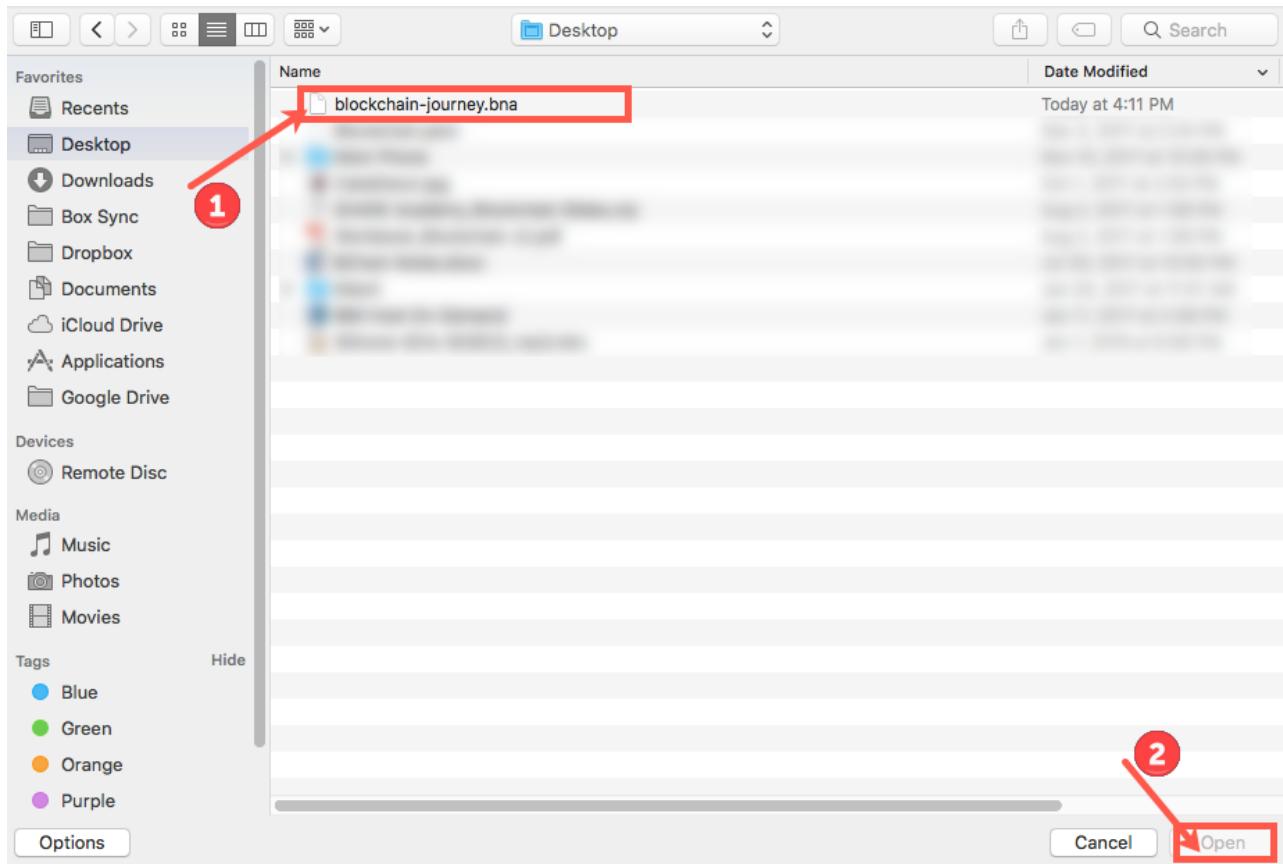
BASED ON basic-sample-network

The Hello World of Hyperledger Composer samples

Contains: 1 Participant Type, 1 Asset Type, and 1 Transaction Type

Deploy

48. Navigate to where you saved your blockchain-journey.bna in step 49. Select blockchain-journey.bna from its directory and click Open.



49. Scroll to the bottom of the page and complete the following:

- **Select ID and Secret.**
- **Create an Enrollment ID of** `admin`.
- **Create an Enrollment Secret of** `adminpw`.
- Note: If you create a different *Enrollment Id* and *Enrollment Secret* then you will need to create and import a network card for that ID. See [Hyperledger Composer documentation for more information.](#)

3. CREDENTIALS FOR NETWORK ADMINISTRATOR

You must provide credentials in one of the following formats before you deploy this business network

The credentials will be used to access the business network once it is deployed

- ① Certificates
Required here are certificate and private key files.
- ID and Secret
These can be created when accessing a business network.

An Enrollment ID and Secret must be created by someone who already has access to the Business Network you are connecting to.

Enrollment ID admin

Enrollment Secret

50. Scroll up to the top and **click Deploy** on the right side.

The screenshot shows the '2. MODEL NETWORK STARTER TEMPLATE' section. It includes fields for 'Give the network admin card that will be created a name' (set to 'admin@hlfv1-blockchain-journey') and 'Choose a Business Network Definition to start with:' (with three options: 'basic-sample-network', 'empty-business-network', and 'blockchain-journey'). A red arrow points to the 'Deploy' button on the right, which is highlighted with a red box. To the right, a 'CONNECTION PROFILE' panel shows the 'Blockchain Journey network deployed to hlfv1' and its details.

Give the network admin card that will be created a name admin@hlfv1-blockchain-journey

2. MODEL NETWORK STARTER TEMPLATE

Choose a Business Network Definition to start with:

Choose a sample to play with, start a new project, or import your previous work

basic-sample-network

empty-business-network

blockchain-journey

Samples on npm

Deploy

CONNECTION PROFILE

Blockchain Journey network deployed to hlfv1

BASED ON blockchain-journey

Creating my first blockchain network.

Contains: 0 Participant Types, 1 Asset Type, and 3 Transaction Types

51. Under *Connection: hlfv1*, find your newly deployed network *hlfv1-blockchain-journey*. **Click Connect now.**

52. Back in your terminal, enter `docker ps -a`. You can see there is now a new container running where Composer Playgroun has deployed code to the Hyperledger Fabric.

```
bcuser@ubuntu16043:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
5237c322cc51519...  dev-peer0.org.example.com-hlfv1-blockchain-journey-0.17.4-cd48c21bc7ba8e1c4db9288ea379ebef7931568023febe00a7a905aca7f0e
ae0e0b151e8d3      hyperledger/fabric-peer:1.0.0-alpha
863b9f919dbd      hyperledger/fabric-ca:390a-1.1.0-alpha
ca909ac009518      hyperledger/fabric-orderer:390a-1.1.0-alpha
b10fb5a5d9d12      hyperledger/fabric-couchdb:390a-0.4.5

```

53. Congratulations! You've deployed your first blockchain application to Hyperledger Fabric.

Generating API

54. In your terminal, issue the following command to start the API rest server:

```
○ nohup composer-rest-server -c admin@hlfv1-blockchain-journey -n always -w
true >/home/bcuser/playground/rest.stdout
2>/home/bcuser/playground/rest.stderr & disown
```

```
bcuser@ubuntu16043:~$ nohup composer-rest-server -c admin@hlfv1-blockchain-journey -n always -w true >/home/bcuser/playground/rest.stdout 2>/home/bcuser/playground/rest.stderr & disown
[1] 12673
```

55. Verify the rest server process is running. `ps -ef|grep rest`

```
bcuser@ubuntu16043:~$ ps -ef|grep rest
bcuser 12673 12628 1 17:53 pts/2 00:00:02 node /home/bcuser/bin/composer-rest-server -c admin@hlfv1-blockchain-journey -n always -w true
```

56. To see your API, go back to your browser and open a new tab or window. In the address bar, enter `http://xxx.xxx.x.x:3000/explorer` where the x's are the IP address for your Linux guest. You should see a page like the one shown.

Method	Path	Description	Show/Hide	List Operations	Expand Operations
POST	/api/ChangeThermostatTemp	A transaction named ChangeThermostatTemp	Show/Hide	List Operations	Expand Operations
POST	/api/CompareWeather	A transaction named CompareWeather	Show/Hide	List Operations	Expand Operations
POST	/api/Sensor	An asset named Sensor	Show/Hide	List Operations	Expand Operations
POST	/api/SetSensorTemp	A transaction named SetSensorTemp	Show/Hide	List Operations	Expand Operations
GET	/api	System : General business network methods	Show/Hide	List Operations	Expand Operation

[BASE URL: /api , API VERSION: 1.0.0]

57. Expand the different methods to see the various calls and parameters you can make through REST API. You can also test the API in this browser to learn how to form the API and see the responses.

The screenshot shows the Hyperledger Composer REST interface. At the top, it displays the URL `148.100.5.9:3000/explorer/#/org5Sample5/org.acme.sample.Sensor_create`. Below the header, there's a navigation bar with links like 'Getting Started', 'Add to Wish List', 'Overview - WW z ...', 'Blockchain - IBM ...', 'Ubuntu Docker Ho...', 'GitHub - IBMHyper...', and a search bar. The main content area is titled 'Hyperledger Composer REST server' and shows the 'org.acme.sample.Sensor' asset. It includes a 'GET /org.acme.sample.Sensor' button and a 'POST /org.acme.sample.Sensor' button. A message box says 'Response Class (Status 200) Request was successful'. Below this, there's a 'Model Example Value' section with JSON code:

```
{
  "class": "org.acme.sample.Sensor",
  "teamID": "string",
  "teamName": "string",
  "sensorTemp": 0,
  "thermostatTemp": 0,
  "recommendation": "string"
}
```

Below the example value, there's a 'Response Content Type: application/json' dropdown set to 'application/json'. Under 'Parameters', there's a table with columns 'Parameter', 'Value', 'Description', 'Parameter Type', and 'Data Type'. One row shows 'data' with a value of a JSON object containing sensor details. To the right, there's a 'Model Example Value' panel with similar JSON. Further down, there's a 'curl' command example:

```
curl -X POST -H "Content-Type: application/json" --header "Accept: application/json" -d '{"class": "org.acme.sample.Sensor", "teamID": "string", "teamName": "CommunityCloud", "sensorTemp": 10, "thermostatTemp": 10, "recommendation": "string"}' http://148.100.5.9:3000/api/org.acme.sample.Sensor'
```

At the bottom, there's a 'Request URL' field with the value `http://148.100.5.9:3000/api/org.acme.sample.Sensor` and a 'Response Body' section showing the expected JSON response.

58. Congratulations! You now have a working blockchain application and have created APIs to call your blockchain application.

Part 3 — Utilizing blockchain API through NodeRED

Importing your flow into NodeRED

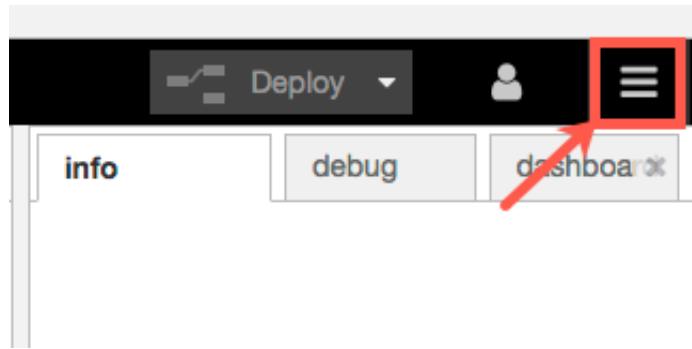
1. Open NodeRED in your browser in a new tab or window. Enter `http://xxx.xxx.x.x:1880` in the address bar where the x's correspond to your Linux guest IP address.
 - o Note: You may need to use a browser other than Chrome for Node-RED. There have been intermittent issues with Node-RED loading properly in Chrome.

The screenshot shows the NodeRED interface. At the top, the browser address bar has the URL `127.0.0.1:1880/#` with a red arrow pointing to it. The NodeRED interface itself has a title bar 'Node-RED'. The left sidebar contains a tree view of nodes under 'input' categories: inject, catch, status, link, mqtt, http, websocket, and tcp. The main workspace is titled 'Flow 1' and contains a single node: an 'inject' node. On the right side, there are three tabs: 'info', 'debug', and 'Flow'. The 'Flow' tab shows the following details:

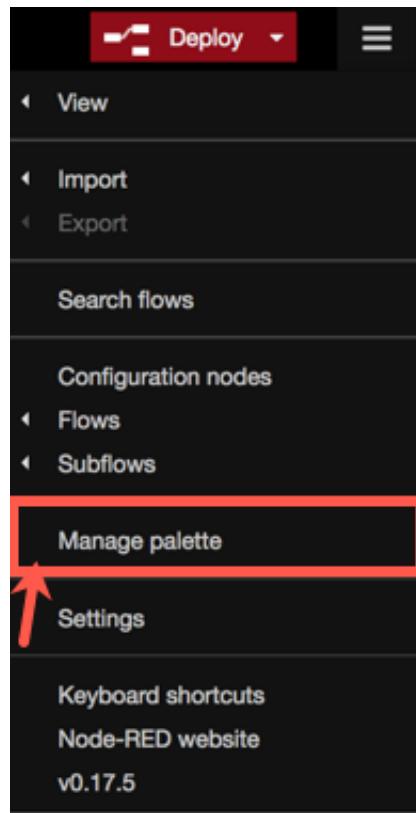
Name	Flow 1
ID	"f3ea0c00.1744b"
Status	Enabled

Below the Flow tab, there's an 'Information' section.

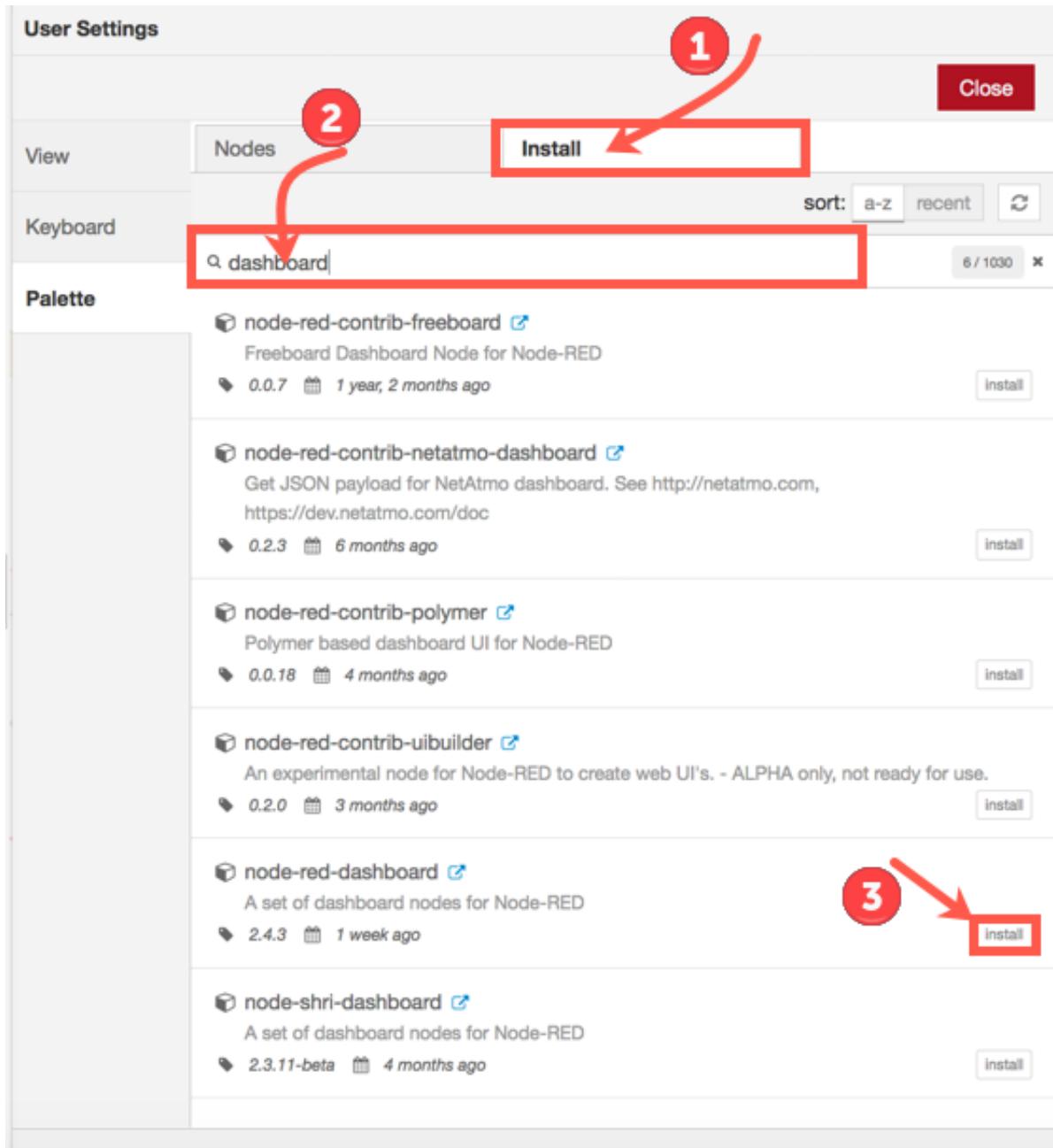
2. You'll need to add a few more nodes to your NodeRED palette to have complete working flows. To do this, **select** the *menu* button in the upper right corner.



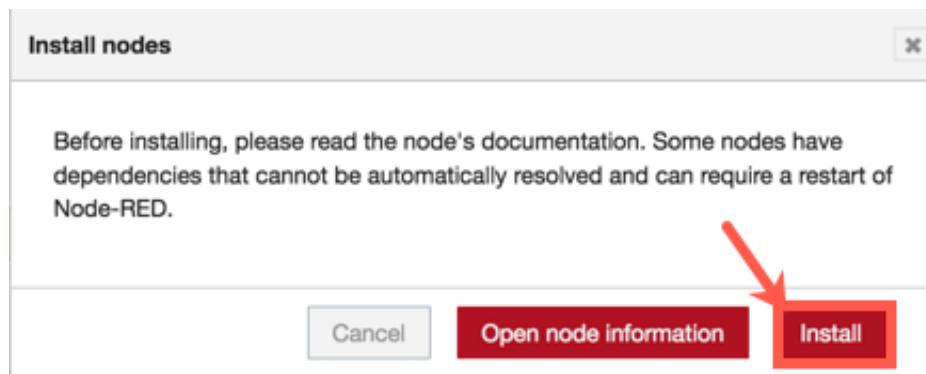
3. **Select** *Manage Palette* from the drop down menu.



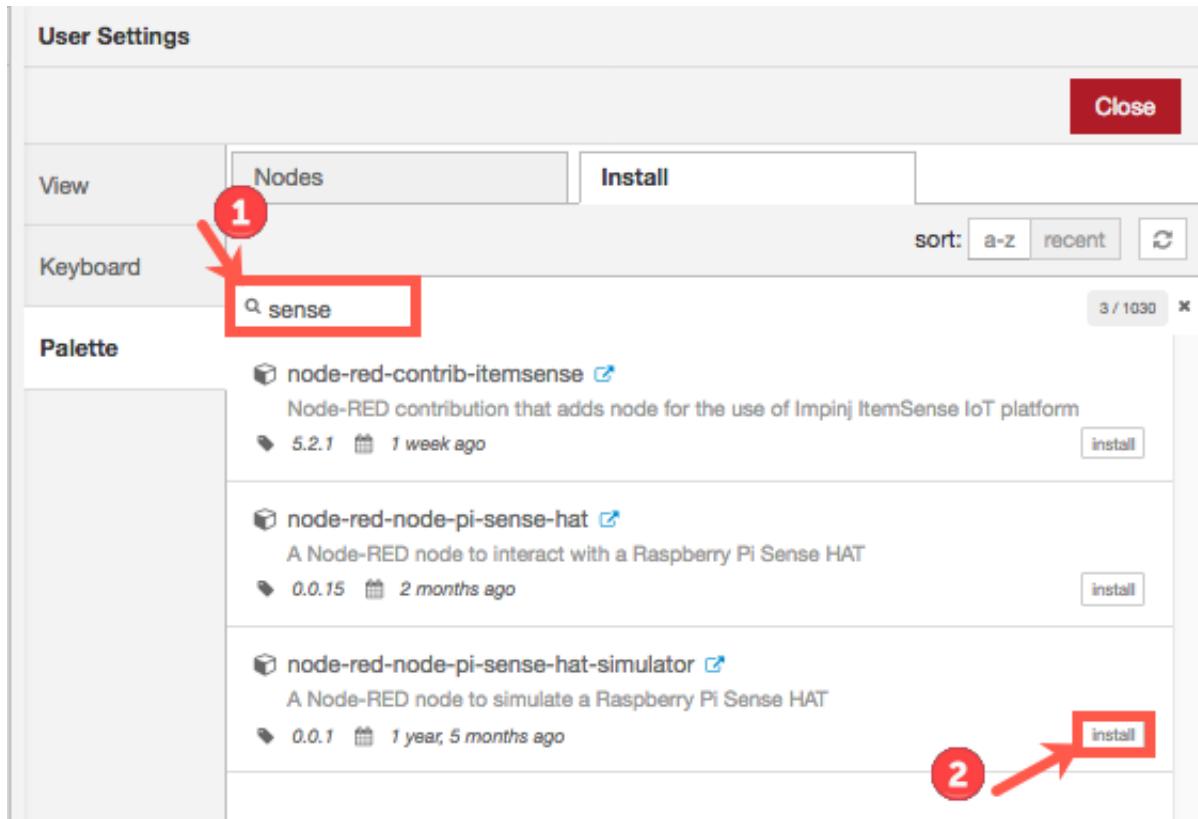
4. In the *User Settings* window, **click** *Install*, **type** `dashboard` in the search bar and select **install** next to *node-red-dashboard*.



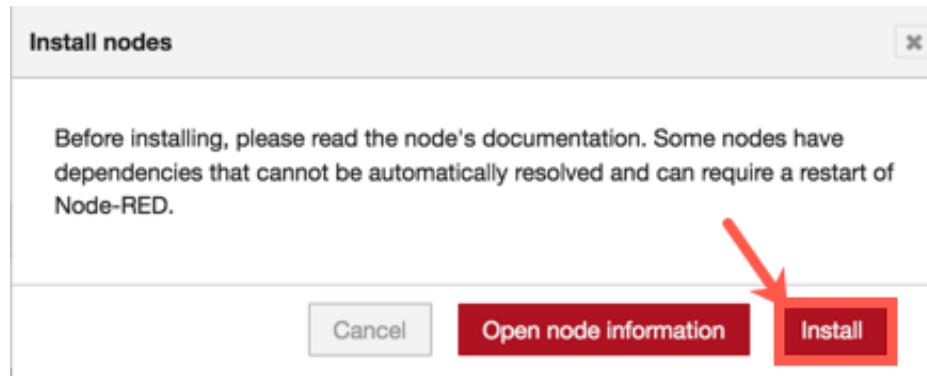
5. In the *Install nodes* pop-up, click *Install*.



6. Now it's time to install a RaspberryPi Sense Hat simulator. In the *User Settings* window under *Install*, type `sense` in the search bar and select **install** next to `node-red-pi-sense-hat-simulator`.



7. In the *Install nodes* pop-up, click *Install*.



8. One more node to add, type **weather** in the search dialog and **select install** next to *node-red-node-weather-underground*.

User Settings

Close

View Nodes Install

sort: a-z recent

Keyboard weather 8 / 1331

Palette

node-red-contrib-netatmo-dashboard Get JSON payload for NetAtmo dashboard. See <http://netatmo.com>, <https://dev.netatmo.com/doc>

0.2.3 1 year, 1 month ago

node-red-contrib-yr A collection of Node-RED nodes for accessing yr.no weather info and data.

0.1.11 2 years, 6 months ago

node-red-node-darksky A Node-RED node that gets the weather forecast from the DarkSky API

0.1.17 3 months ago

node-red-node-forecastio A Node-RED node that gets the weather forecast from Forecast.io

0.1.12 1 year, 6 months ago

node-red-node-openweathermap A Node-RED node that gets the weather report from openweathermap

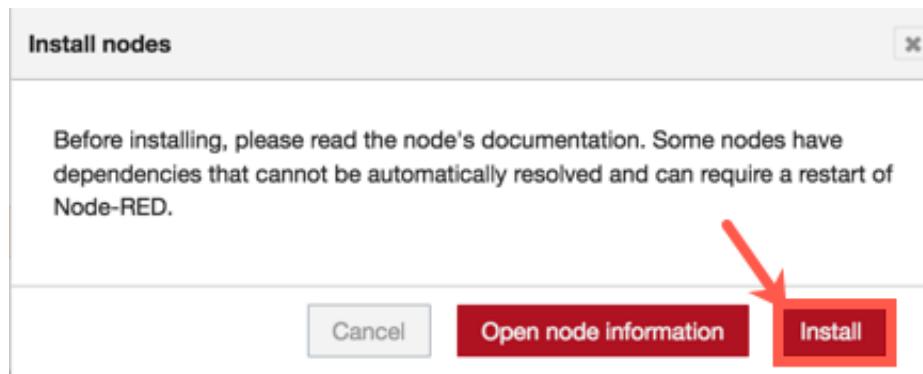
0.2.0 1 week ago

node-red-node-weather-underground A Node-RED node that gets the weather report and forecast from The Weather Underground

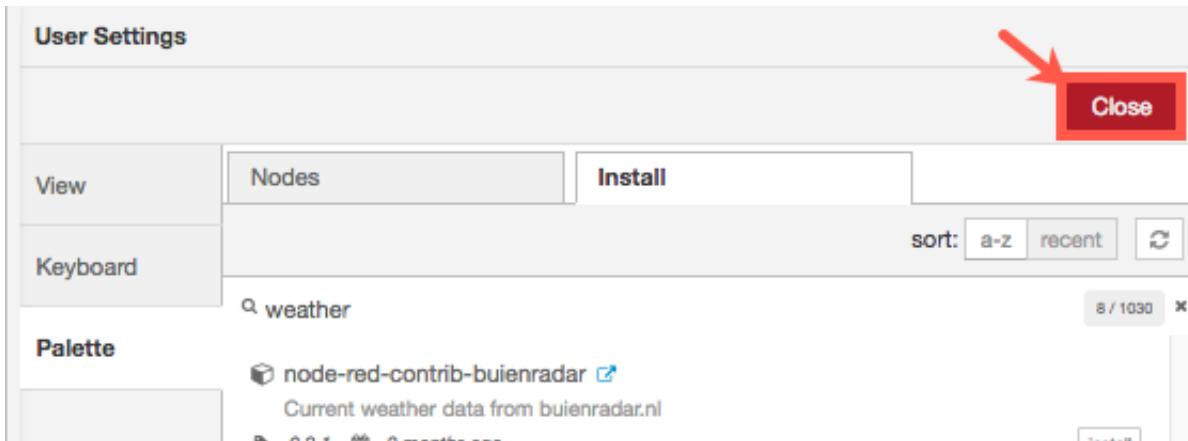
0.1.12 1 month ago 2

 1

9. In the *Install nodes* pop-up, click **Install**.



10. Click **Close** to leave the User Settings dialog.

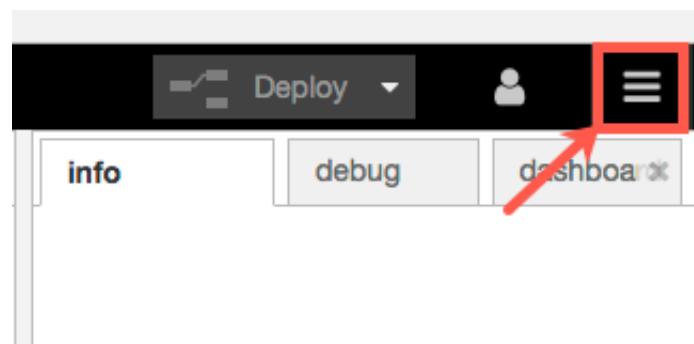


9. Copy all of the JSON from the [GitHub repository](#) to import into the flow.

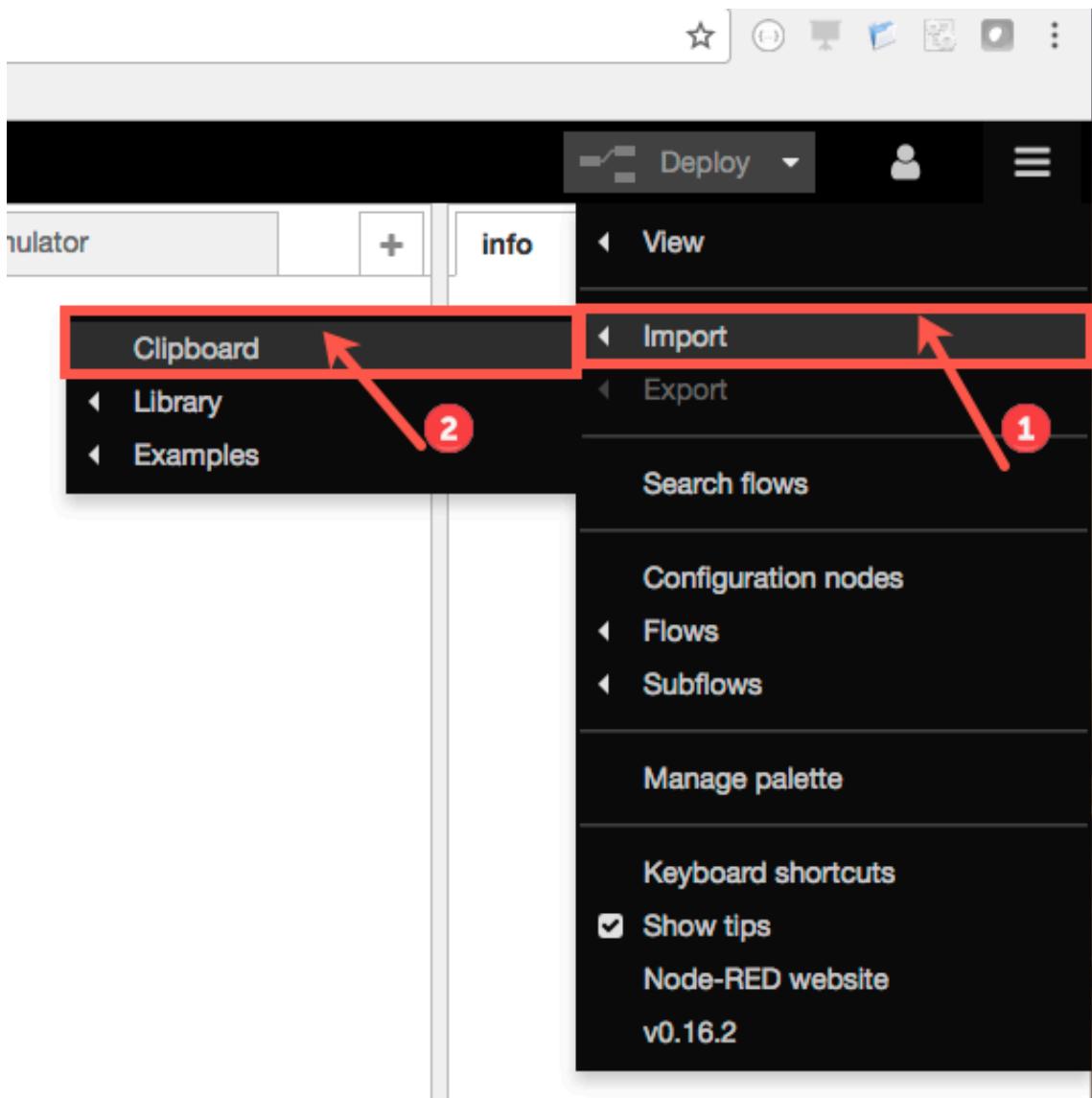
- **Note:** The easiest way to do this is clicking the hyperlink above. You can also find it in the GitHub repository in the code folder as `node-red.json`. To view it in GitHub, **click** on `node-red.json`, **select raw** and then copy it.

```
{
  "id": "bd033c75.c97bd",
  "type": "tab",
  "label": "Blockchain",
  "disabled": false,
  "info": ""
},
{
  "id": "b496f71c.546088",
  "type": "tab",
  "label": "Dashboard",
  "disabled": false,
  "info": ""
},
{
  "id": "6b9de9e3.32ee28",
  "type": "tab",
  "label": "Gauge Simulator",
  "disabled": false,
  "info": ""
},
{
  "id": "cbc61383.cc0f5",
  "type": "ui_group",
  "z": "",
  "name": "Blockchain",
  "tab": "70b558f5.711d68",
  "order": 2,
  "disp": true,
  "width": "6"
},
{
  "id": "fa648df3.380e48",
  "type": "ui_group",
  "z": "",
  "name": "Thermostat",
  "tab": "70b558f5.711d68",
  "order": 4,
  "disp": true,
  "width": "6"
},
{
  "id": "70b558f5.711d68",
  "type": "ui_tab",
  "z": "",
  "name": "Home"
}
```

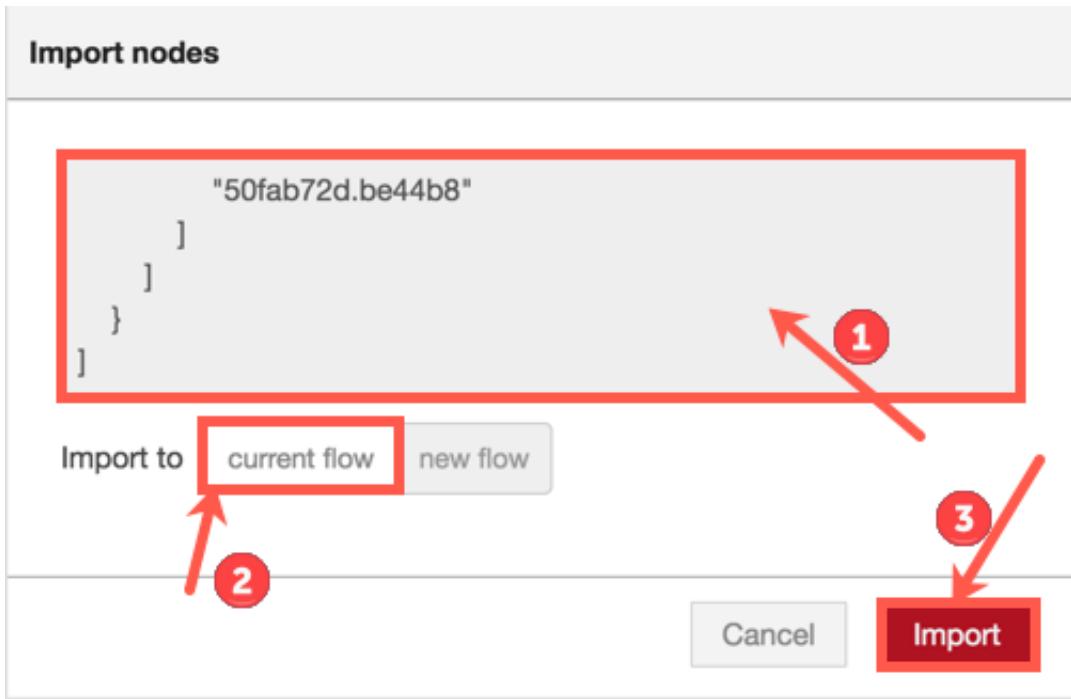
10. Paste it into NodeRed, by **clicking** on the *menu icon* in the upper right corner.



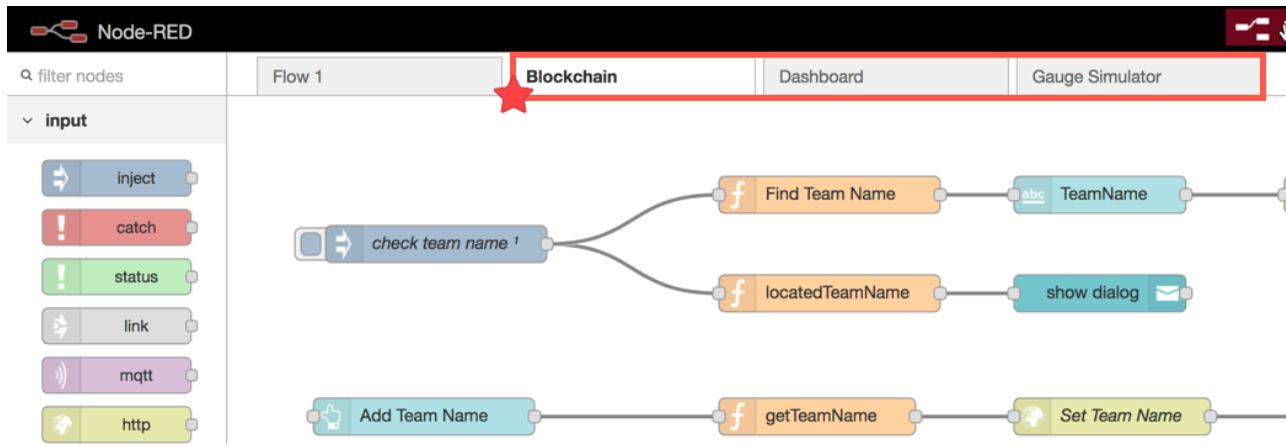
11. Select **Import -> Clipboard**.



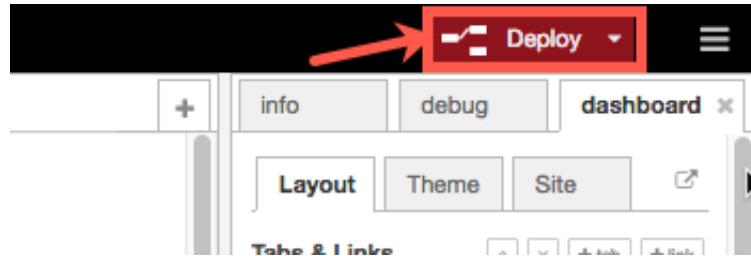
12. Paste the code in the editor. Make sure to select "current flow" button. **Select Import.**



13. You should now have three flows in NodeRED — Blockchain, Dashboard and Gauge.

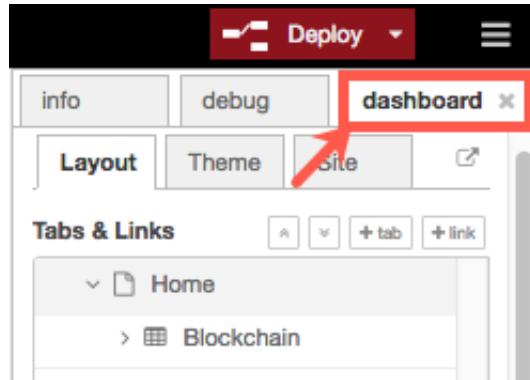


14. On the right side of the Node-RED browser, click Deploy to save the work you've imported.

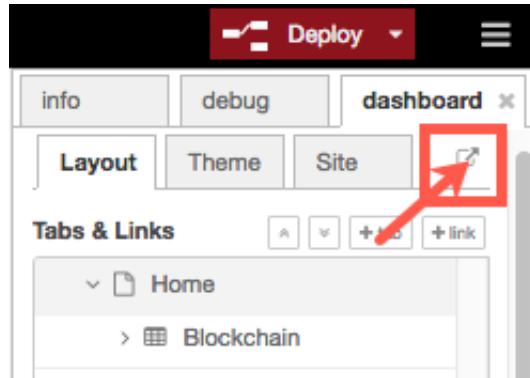


Interacting with blockchain through a dashboard

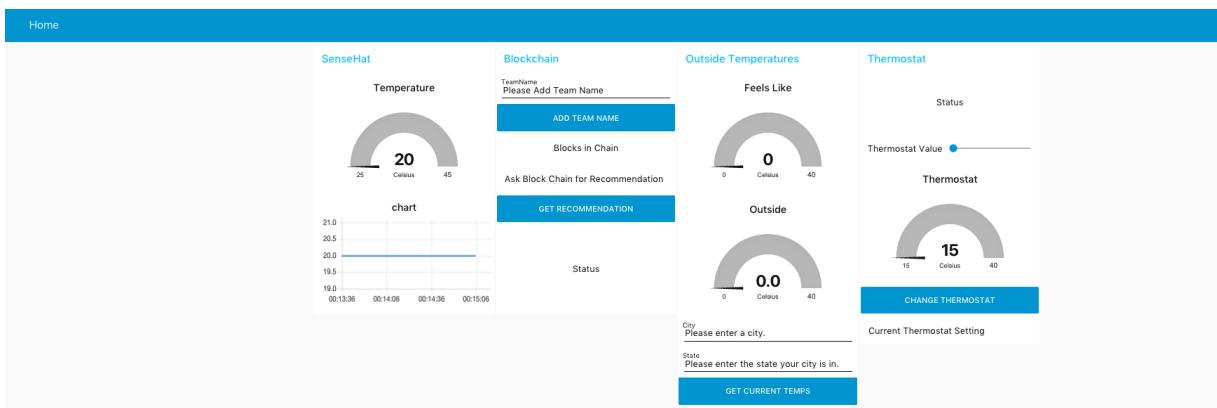
15. Click the *dashboard* tab in the upper right corner under *Deploy*.



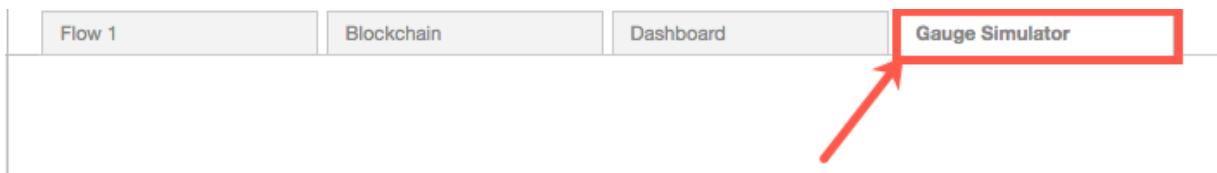
16. Click the *pop out* button to open the dashboard in a browser.



17. A new tab should open. Your dashboard should look like the following image.



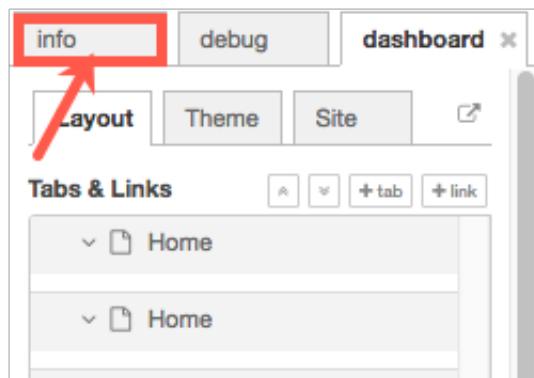
18. To interact with your simulated RaspberryPi, go back to your Node-RED tab and select the *Gauge Simulator* tab.



19. Click on the *Sensor Gauge Simulator* node.



20. On the right side of the browser, **click** on the *Info* tab.



21. In the third paragraph under Information there is a hyperlink for the word *here*. That hyperlink opens the simulator in a new tab. **Click** *here*.

info debug dashboard

Node

Name	Sensor Gauge Simulator
Type	rpi-sensehatsim in
ID	"c99759a9.968038"

show more ▾

Information

Raspberry Pi Sense HAT Simulator input node.

This node simulates readings from the various sensors on the Sense HAT, grouped into three sets; motion events, environment events and joystick events.

Once deployed, the simulator can be accessed [here.](#)

Motion events - not currently supported by the simulator

Motion events include readings from the accelerometer, gyroscope and magnetometer, as well as the current compass heading. They are sent at a rate of approximately 10 per second. The `topic` is set to `motion` and the `payload` is an object with the following values:

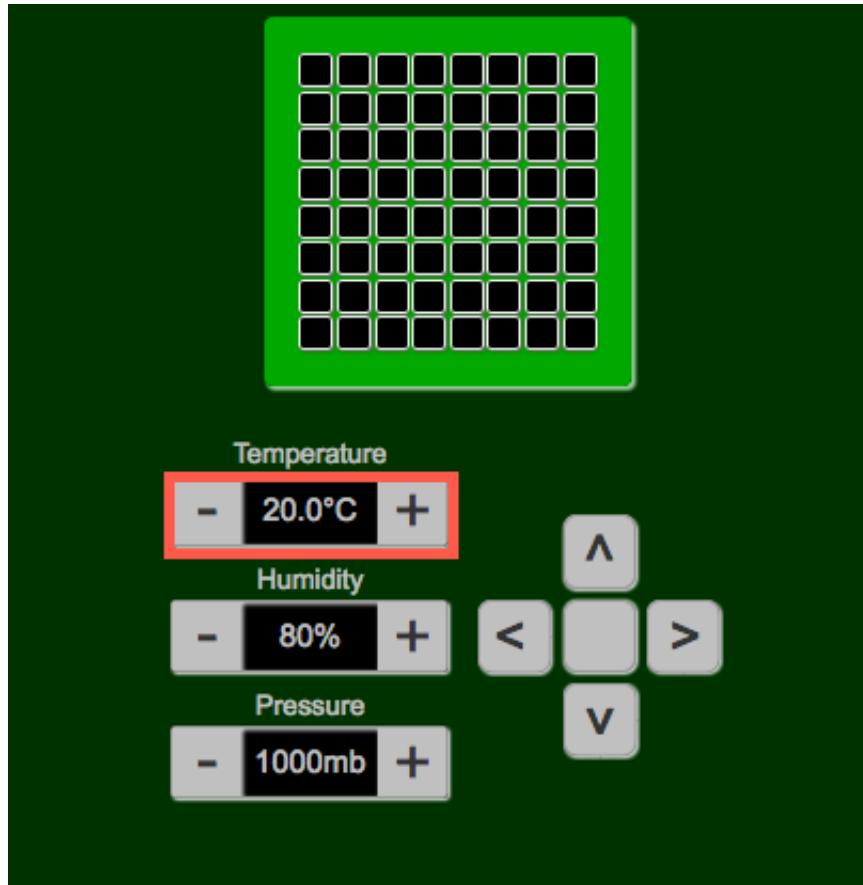
- `acceleration.x/y/z` : the acceleration intensity in Gs
- `gyroscope.x/y/z` : the rotational intensity in radians/s
- `orientation.roll/pitch/yaw` : the angle of the axis in degrees
- `compass` : the direction of North in degrees

Environment events

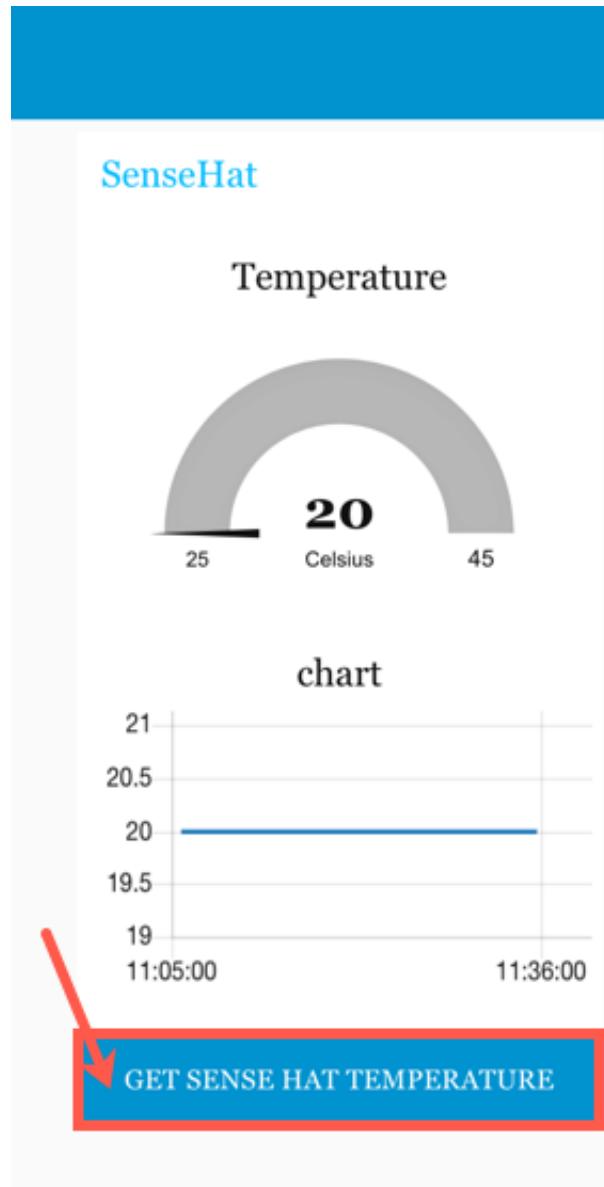
Environment events include readings from the temperature, humidity and pressure sensors. They are sent at a rate of approximately 1 per second. The `topic` is set to `environment` and the `payload` is an object with the following values:

- `temperature` : degrees Celsius
- `humidity` : percentage of relative

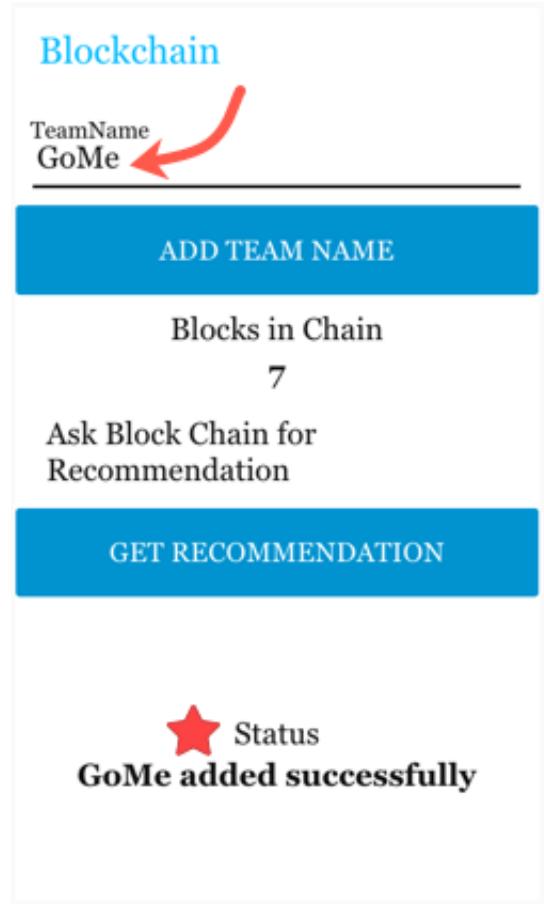
22. Adjust the temperature of the sensor in the simulator from 20 C.



23. Switch to the tab for your dashboard, **click** *Get Sense Hat Temperature* to see the change to the sensor temperature and graph.



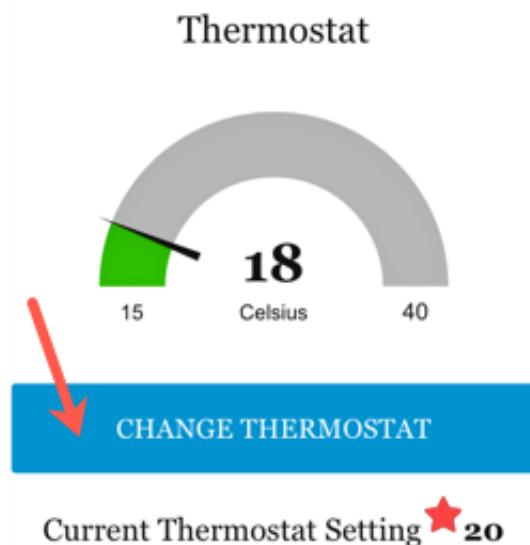
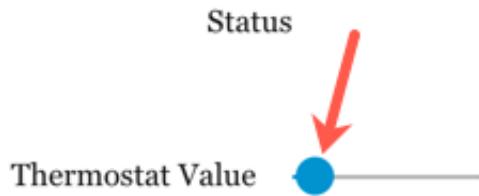
24. Continue to play with your dashboard to interact with blockchain via API. You can try the following:
 - Type in a team name & click *Add Team Name*. You should see a successful message afterward.



- Change the thermostat by moving the slider next to *Thermostat Value*. Click *Change Thermostat* to send the value to blockchain.'

Thermostat Value 

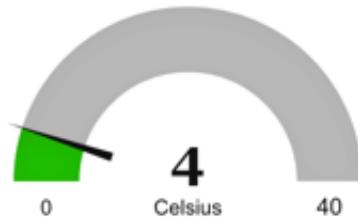
Thermostat



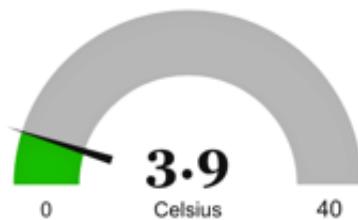
- Type in a City and State to find outside temperatures. Click *Get Current Temps* to get the values.

Outside Temperatures

Feels Like



Outside



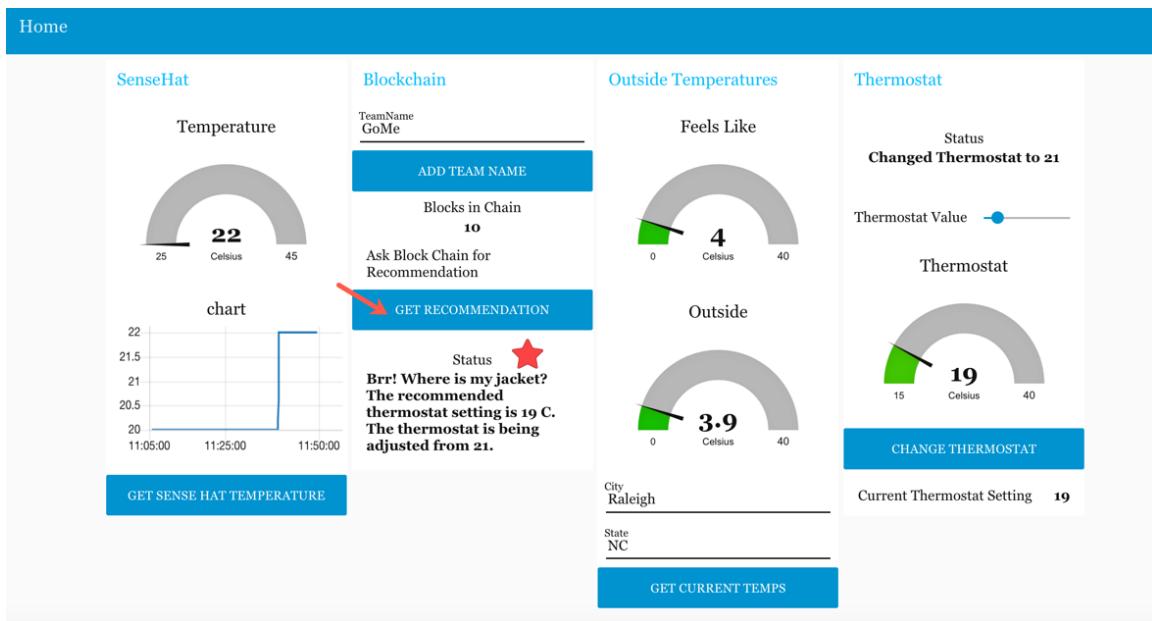
City
Raleigh

State
NC

 GET CURRENT TEMPS

- After you have outside temperatures, you can click *Get Recommendation* to find the ideal temperatures for the thermostat. Notice that this will adjust your thermostat in the dashboard.

○



25. Congratulations! You've completed this lab!